A New Approach to Tagging in Indian Languages

Kavi Narayana Murthy and **Badugu Srinivasu**
Department of Computer and Information Sciences
University of Hyderabad
knmuh@yahoo.com, srinivasucse@gmail.com

Abstract

Tagging is the process of assigning short labels to words in a text for the purpose of indicating lexical, morphological, syntactic, semantic or other such information associated with these words. When the focus is mainly on syntactic categories and/or sub-categories, this is also known as part-of-speech or POS tagging. It may be noted that the term tagging is broader than the term POS tagging. One of the main reasons for incorporating a tagging level between lexical and morphological levels on the one side and syntactic parsing on the other side, is to reduce ambiguities. Tag ambiguities multiply at an exponential rate making syntactic parsing so much more difficult.

The design of the tag set is critically dependent on the purpose and the approach taken for tagging. The beaten path is to develop a manually tagged database of sentences and then use this for training a machine learning algorithm. The machine learning algorithm is expected to generalize from these training examples so that it can then tag any new sentence. Manual tagging is difficult, time consuming and prone to human errors. Consistency is difficult to achieve especially if the tag set is elaborate. Also, given the limited amount of training data that is practically possible to develop, a large and detailed tag set will lead to sparsity of training data and machine learning algorithms will fail to learn effectively. From these considerations, researchers tend to restrict themselves to small, shallow or flat tag sets which are least confusing to human annotators. When this idea is taken to the extreme, useful sub-categorizations may be lost. In this paper we propose an alternative view and a novel approach to tagging with focus on Indian languages. We demonstrate our system for Telugu and Kannada languages.

We believe that a lot of avoidable confusion arises because of a heavy emphasis given to the orthographic rendering of texts. We define words as meaningful sequences of phonemes and we try not to get influenced by the written form. Whether and where spaces appear, for example, are irrelevant to us. When viewed from this stand point, we find that the degree of lexical ambiguity is far less than normally seen in other approaches. A vast majority of words are not ambiguous at all. Most of the remaining cases of ambiguity at root word level get resolved automatically once we consider the morphology of the inflected forms in which these words usually occur in running texts. Therefore, we believe that the main task of tagging is not one

of tag assignment but only of tag disambiguation. Tags can be assigned by the dictionary and morphology components quite effectively. This way, we can develop large scale annotated corpora with high quality of tagging, without any need for manual tagging or any machine learning algorithm. We demonstrate our approach for Telugu and Kannada and argue for the merits of our approach compared to other competing approaches.

1 Introduction

1.1 Language, Grammar and Computation

We human beings are capable of producing a number of different kinds of sounds. We are capable of making interesting patterns by stringing together these basic sound units, called phonemes, into larger structures. And, most importantly, we are capable of systematically associating meanings with these patterns of sounds. Further, we are capable of learning such associations, we are capable of effectively communicating these association rules to others. We are also capable of communicating our ideas, thoughts, feelings and emotions to others by expressing them in patterns of sounds according to these mutually agreed upon mapping rules. This faculty of the human mind is called language. Language is the capacity to map sounds to meanings and use this for speech and thought. Language is, by far, a unique gift of nature to mankind.

Computers are capable of storing and manipulating symbol structures. They are not capable of understanding meanings. Computers do not understand the meaning of any single word in any human language. How then can we put computers to good use in meaningful processing of language? The answer to this question comes from observing the fact that there is structure in language and there is a systematic relationship between structure and meaning. This relationship between structure and meaning can be observed, learned, taught and used. Therefore, if only we take care to store only meaningful structures and if only we take care to allow only meaningful manipulations of such structures, we can ensure that everything remains meaningful (to us) throughout, although the machine itself does not understand a word. This systematic relationship between structure and meaning is what we shall call grammar. Grammar is thus the key to language processing in machines.

Grammar is the key to language processing even in humans. We do not simply store all possible linguistic units and their corresponding meanings. The number of possible sentences, for example, is infinite and we are capable of understanding the meaning of sentences we have never heard before in our life. This is possible only because we have a grammar in our head and we use this grammar to construct new sentences or to understand sentences spoken by others. This is true not only of sentences but also of all levels of linguistic analysis.

To summarize, we must develop appropriate representations of basic linguistic units, appropriate representations of their structures and appropriate formalisms for manipulating these structures at all levels of linguistic analysis. What is appropriate and what is not is dictated mainly by meaning. The written form has no role at all in this. This is a brief summary of the theory we have been working on. See [1] for more details.

1.2 Language and Script

Language is a powerful means of communication. Of course we can also communicate certain ideas and feelings through body language, gestures etc. Simply getting up or walking out can also convey some message to others. We can even communicate at times through silence. Nonetheless, by and large the most effective and most widely used means of communication among humans is by making sounds. We have therefore defined language as a mental faculty of human beings that enables us to systematically map sound patterns to meanings. Language is speech, it has nothing to

do with writing. We believe that an enormous amount of confusion has been created both within NLP and in Linguistics by giving too much of importance to the written form. We all learned our first language only by listening and speaking. Reading and writing are learned later, that too only upon being taught. Literacy is not as important as people think today, one can be a highly knowledgeable scholar without being literate. Many languages of the world do not have a script of their own, the need for writing was never felt all through the history of many cultures. Writing is a technology, that has been invented by us as an after-thought, while language is a natural gift of nature to mankind. Do not confuse language for writing or script. Language can exist without a script but not vice versa. It is unfortunate, therefore, that we have started defining everything based on the written form. Words are not sequences of letters or characters separated by spaces. It does not matter if there are zero, one or more spaces within or between words. In fact inserting spaces is also a newly developed idea - stone inscriptions do not have spaces between words, for example. There are no spaces between words in speech. A word cannot be defined in terms of written symbols separated by white space. This is just not right.

1.3 Words, Word Classes and Tagging

A dictionary stores words and their meanings. Morphology deals with the internal structure of words. A tagger attaches tags to words. Sentences are built up from words. Words form very important and fundamental units of language. What exactly is a word then?

Words are minimal sequences of phonemes with meaning. Words that indicate things are called nouns. Words that indicate action or state are called verbs. Words that describe things are called adjectives. Nouns, verbs, adjectives are examples of word classes. Conjunctions, articles, prepositions etc. are not word classes, they do not correspond to words, they do not correspond to phoneme sequences with a clear meaning of their own. The so called function words are not words at all. Of course we have various theories of meaning and we may also have border line cases where it is difficult to decide if a word has any independent meaning of its own or not. We will have to take a carefully considered, yet practicable and consistent stand point. A sentence is not merely a sequence of words, a sentence may contain words, feature bundles that can be associated with words or other larger linguistic structures, connectives etc. The sentence 'John has been jumping' does not have four words, it has only two. John is the person who is doing something and what he is doing is expressed by the three tokens 'has been jumping'. John is performing only one activity, not three. There can thus be only one verb here, after all, a verb is a word that indicates an action and there is only one action here. There are no such things as auxiliary verbs, we are not indicating any auxiliary, constituent or related activities here, we are talking of one atomic activity and that is all. From these examples, it should be clear that our theory has far reaching implications at all levels of linguistics and NLP. Interested readers may see [1] for more details.

Some may argue that definition words in terms of meanings is not practicable since computers do not understand meaning. If the right thing is difficult to do that is not a valid excuse for doing the wrong thing. Do pre-processing, do post-processing, have manual intervention, do whatever you wish but do not stray away from the truth too far. We believe that it is practically possible to work with meaning-defined words in all languages of the world.

Word classes such as noun, verb and adjective are also called 'Parts of Speech' (POS) by tradition. For the sake of convenience, we may use short labels, called tags, for these. For example, nouns may be indicated by N and verbs by V. POS Tagging is the process of attaching such short labels to indicate the Parts of Speech for words.

Tags need not indicate purely syntactic categories. There is need for sub-categorization in syntax and a tagging scheme may include not only the major grammatical categories but also sub-categories. For example, one may talk of common nouns and proper nouns, of intransitive verbs and transitive verbs. One can actually go beyond purely syntactic properties and include lexical, morphological or even semantic information in the tags. It all depends upon what we need and what we can. In this paper we use the terms Tag and Tagging in this broader sense, not restricting ourselves to POS tags or POS tagging.

Tagging is only for convenience. However, tagging is usually intended to reduce, if not eliminate, ambiguities at word level. It is well known that syntactic parsing is at least cubic in computational complexity and having to consider several alternative interpretations for each word can exponentially increase parsing complexity. Tagging has been invented in NLP as an independent layer of analysis, sitting between morphology and syntax, mainly to help the syntactic parser to do better in terms of speed. However, if we take the definition of word we have given here, we will find that sentences are not as long as they appear to be (in terms of number of words) and words are not as ambiguous as they appear to be either. Therefore, syntactic parsing is actually orders of magnitude simpler than what we usually think it is. To this extent, the importance of tagging is reduced. It is worth noting that linguistic theories never posited tagging or chunking as separate layers of analysis sitting between morphology and syntax.

1.4 Grammar

Words are finite, the mapping from words (that is phoneme sequences) to meanings can be stored in our brain. This is the mental lexicon. But there are infinitely many possible sentences and we can understand all of them. Our mental capacity is finite and so we must necessarily be using a finite device to handle the infinitely many sentences. This mental device we have that enables us to construct and analyze infinitely many valid sentences using the finite vocabulary we have is called grammar.

Consider the sentences 'Rama saw the running deer' and 'Rama saw the deer running'. Sentences having the same set of words can thus vary in meaning and the difference can only be accounted for by the structure. Grammar is the finite device that maps an infinite variety of structures to their corresponding meanings. Grammar is the central core of the human language faculty - without grammar, language is impossible.

The lexicon maps phoneme sequences to meanings. Sequence is also one kind of structure, although a simple one. Therefore, a lexicon is also a grammar. The lexicon is not an adjunct to grammar or an independent module, it is itself a part of grammar.

Words of a language are finite and hence listable. If all words of a language can be simply listed in the lexicon, there is no need for morphology. However, there is structure inside words, there are systematic relationships between these structures and meanings and these systematic relations can be observed, learned, taught and used by the human mind. The human mind has a natural tendency to observe systematic relationships and make generalizations. Thus, morphology, which deals with the internal structure of words in relation to their meanings, also become a component of grammar. The lexicon, the morphology and the syntax constitute the three main components of grammar up to the sentence level.

1.5 Computational Grammar

A complete grammar of any given language must include the complete lexicon, the complete morphology and the complete syntax. Samples will not do, we need to be comprehensive and exhaustive. Only a complete grammar can define the language fully. The lexicon, morphology and syntax should be mutually exclusive and complementary. For example, what is

handled by morphology need not be, in fact, should not be listed in the lexicon. Because of such interactions, it is not possible to have a dictionary without morphology and syntax, nor can we have a syntactic grammar without a dictionary. The dictionary, morphology and syntactic grammar always go together and must always be viewed as a whole.

We cannot open up the human brain and see what kind a grammar is sitting there but a good grammar needs to be psychologically plausible as also simple, neat and elegant. It must capture generalizations adequately and satisfactorily. It must have predictive and explanatory power. It must be universal. A child born in any language community anywhere in the world picks up its mother tongue with equal ease and in more or less the same amount of time. Hence there must be universal principles underlying and governing all human languages. The main goal of modern linguistics is to discover such a universal grammar.

The main goal of computational linguistics is to discover such a comprehensive yet simple, neat, elegant, and universal grammar. The computer is only a powerful tool in our hand in this grand project. A computational grammar is not a different kind of grammar, it is only a comprehensive, universal, yet simple and elegant grammar, the only difference is that it has been implemented, tested and validated on real data by actually building computational models.

A complete grammar must be capable of handling each and every valid structure and map it to appropriate meanings. How can we be sure? The only way to test and ascertain this is to test on large scale real life data. A computational grammar is simply a grammar that has actually been implemented as a computer program and subjected to extensive testing and validation on real life linguistic data. Even to build such an exhaustive grammar, we invariably need large scale data. A computational grammar is designed, developed, tested, validated on large scale real data. In order to do this, the grammar itself needs to be defined at a very minute level and in a very precise way. A whole lot of definitions and treatments you will find in grammar books are very superficial, cursory, exemplary and merely illustrative, not exhaustive. These have never been tested and validated.

Linguists often fail to understand the importance of the size of data used for building and testing grammatical systems. They think there are only a few types of structures and there is no point in looking at a thousand examples of each kind. There is more to it than meets the eye. There are a large number of significant linguistic phenomena and not all of them occur equally frequently. Those who are used to looking at only a few important phenomena will not understand the importance of large scale data. Rare phenomena are more likely to occur in a large corpus than in a small corpus. Therefore, when the aim to develop wide coverage, comprehensive, if not exhaustive grammars, the importance of a large and representative corpus cannot be undermined.

We believe that we can go much closer to the dream of a universal grammar if we take the definition of word we have given here seriously. Meaning is central to language and linguistics and any process that causes loss or distortion of meaning is simply not acceptable. We have started off on our journey in this direction and this paper we shall give you a glimpse of our first few steps.

2 Designing a Tag Set

Tagging approaches based on machine learning require manually tagged training data. Manual tagging becomes difficult and error prone as the tag set becomes large and elaborate and so there is a strong tendency to go for small, flat tag sets. Such tag sets may not capture all the required and/or useful bits of information for carrying out various tasks in NLP. Flat tag sets are also rigid and resist changes. Hierarchical tag sets are more flexible. In our case, we do not depend

upon statistical or machine learning techniques and we do not need any training data. No manual tagging work is involved and so we can afford to have large, elaborate, hierarchical tag set that carries as much of lexical, morphological, syntactic and semantic fields as we wish.

One of the biggest difficulties that researchers face while designing tag sets, while performing manual tagging and while building and evaluating tagging systems is the very definition of tags. Tags involve lexical, morphological, syntactic and semantic considerations and often there are conflicts. One cannot go purely by intuitive definitions such as 'nouns are things, pronouns stand in place of nouns and adjectives modify nouns'. We will need to give precise definitions and criteria to decide which tag label should be given to which word. Let us illustrate with some examples. In Kannada, one may argue that 'obba' is a pronoun because it stands for some one person. Another may argue that it should be treated as an adjective since it indicates number, (apart from indicating that the following noun should be human) as in 'obba huDuga' (cf. 'oMdu mara'). A third person may counter this by saying it cannot be an adjective since it can take nominal inflections such as number and case. The word 'obba' takes nominal inflections and can be subject of a sentence and so it can only be a noun or a pronoun. But pronouns cannot modify nouns, they can only stand in place of a noun, and so this word can only be taken as a noun. In order to avoid such confusions, we will need to delineate the characteristics of each of the categories and sub-categories very precisely so that even a machine can decide on the right tag, given the necessary properties. We shall illustrate this with just a few examples below.

Since both nouns and pronouns can take nominal inflections and both can function as subjects and objects of sentences, why should we even make a distinction between them? After all, there is no real difference in morphology either. Upon careful examination, we find that pronouns are neither modified by adjectives nor do they function like adjectives, modifying other nouns. Nouns can. A noun can be modified by a demonstrative adjective, a pronoun cannot be. Nouns can be modified by quantifying adjectives, pronouns cannot be. As such, there are significant differences at the level of syntax and that is why we must distinguish between these two categories. Similarly, common nouns differ from proper nouns in significant ways. Demonstrative adjectives, quantifying adjectives, ordinals, and descriptive adjectives need to be treated differently since they have different roles in chunking. Not making such distinctions will lead to unnecessary explosion of possible parses, making the parsing process slow and less accurate. Even within proper names, names of persons, places etc. vary in their grammatical properties. Place names appear more in nominative, dative and locative cases, and not so much in accusative case. A place name can modify a person name but not vice versa. Place names can modify common nouns, person names usually will not. Of course some of these may be hard constraints and some may be more of a matter of degree but these do matter in parsing.

Keeping such considerations in mind, we have developed a fairly elaborate hierarchical tag set starting from Kannada and Telugu data. Tagging is an intermediate level and appropriateness of tag set and tag assignments can only be verified by placing it in between morphology and syntax and building and testing actual systems. We are in the process of developing an end-to-end system and this gives us strong basis to argue why our design is superior and why exactly other possible alternatives will not do. Let us simply list our main tags here. See [2] for definition of each tag and comparisons with other tag sets.

It must be made clear that conjunctions, interjections, symbols etc. are not really words at all and they really have no place in the dictionary. Nevertheless, they are parts of sentences and we need to deal with them at the level of syntax. For the sake of practical convenience,

we have included them within the dictionary instead of storing them elsewhere.

3 A Novel Approach to Tagging

There is only one critical question that we need to ask when it comes to tagging - where can we find the crucial bits of information required to assign the correct tag to a given word in a given sentence? Statistical approaches assume that the necessary information comes from the other words in the sentence. In many cases, only the words that come before the current word are taken into direct consideration. We believe, in sharp contrast, that the crucial information required for assigning the correct tag comes from within the word. It is the internal structure of a word that determines its grammatical category as also sub-categorization and other features. True, there will be instances where the internal structure alone is not sufficient. Firstly we find that such cases are not as frequent as you may be thinking. A vast majority of the words can be tagged correctly by looking at the internal structure of the word. The crux of tagging lies in morphology. This is clearly true in the case of so called morphologically rich languages but this is actually true of all human languages if only we define words in terms of meanings rather than in terms of the written form and spacing considerations. Secondly, in those cases where morphology assigns more than one possible tag, information required for disambiguation comes mainly from syntax. Syntax implies complex inter-relationships between words and this cannot be reduced to a mere sequence of entities. In Kannada, for example, the plural/honorific imperative form of a verb and a past conjunctive verbal participle form are the same. Hence morphology cannot resolve this ambiguity. This ambiguity can only be resolved by looking at the sentence structure. If this word is functioning as a finite verb, it must be the imperative. If it is followed by another finite verb later in the sentence, this could a conjunctive participle. Statistical techniques are perhaps not the best means to capture and utilize such complex functional dependencies. Instead, chunking and parsing will automatically remove most of the tag ambiguities. Given this observation, we use a simple pipe-line architecture as depicted in the figure below. We keep going forward and we do not need to come back again and again to preceding modules. We carry with us all the necessary/useful information in the form of tags, each module adding or refining the information as we move on. The lexicon assigns tags to words that appear without any overt morphological inflection. Morphology handles all the derived and inflected words, including many forms of saMdhi. The bridge module combines the tags given by the dictionary and the additional information given by the morph, making suitable changes to reflect the correct structure and meaning where required. The chunker takes these tag sequences to produce chunks. The parser analyzes these chunk sequences and produces a dependency structure. The overall tag structure remains the same throughout, making it so much simpler and easier to build, test and use.

3.1 Architecture

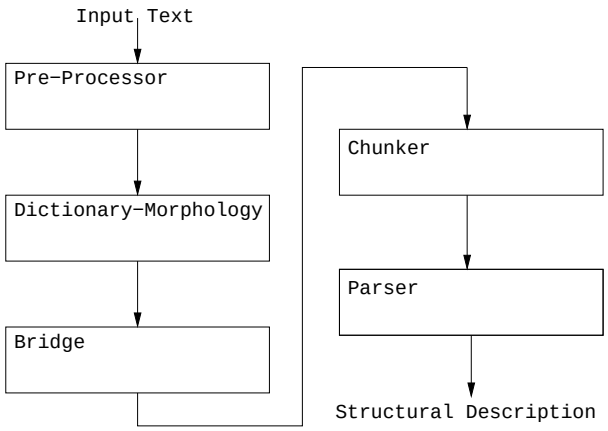

FIG 1 The System Architecture

3.2 Design of the System

3.2.1 Layered Approach to Grammar

We have two contradicting requirements. Firstly we need to develop a grammar that is simple, neat and natural. Secondly, we need to develop a grammar that can actually handle all

linguistic data as we see in a corpus, including all the variations and deviations. A real corpus will include loan words, code mixing and code switching (that is, mixing of the grammars of two different languages, or switching from one to the other in the same discourse), mixture of various dialects, mixing of colloquial forms and standard written forms, mixture of old and modern versions, spelling variations and spelling errors, incomplete and ungrammatical sentences, sentence fragments such as headings and titles, tables, lists, everything. How do we satisfy both of these contradictory requirements?

The solution is to go for a layered approach to grammar development. There will be a core and several layers or wrappers around it. The core will be designed for a restricted subset of the whole of language. These restrictions should not come in the way of expressive power, the restricted set should also be a complete language, not a sub-language. The core only removes those dimensions of variability that do not restrict the expressive power of the language. For example, in English both 'grey' and 'gray' are accepted spellings and if we choose any one of these to build a grammar of English, such a grammar could still be a complete grammar of English. Dialectal variations, colloquial forms, loan words, code mixing, spelling variations, even grammar variations can be abstracted out to define the core grammar. In our work here, the standard written form of modern Kannada and Telugu are taken to define the core.

The dictionary includes the standard forms as also the variants with a pointer back to the standard forms. The pre-processor handles dialectal and colloquial forms and spelling variations. A separate module is posited for handling spelling errors, named entities, etc.

This layered approach to grammar has the potential to give us simple, elegant, efficient grammars which can also handle all the jaggy edges we see in actual usage.

3.2.2 The Dictionary

The dictionary is supposed to contain words and the associated meanings. Meanings are only for use by humans, the machine cannot do anything with meanings directly, it does not understand the meanings of words whatever language we may choose to use. In the current version of our system, our focus has been only on the automatic processing aspects and so we have left out meanings. We will need to add meanings later. Right now, the dictionary only contains words and the associated grammatical information. We include a variety of kinds of grammatical information including lexical, morphological, syntactic and even semantic aspects, depending upon the need and usefulness. For example, gender is given for nouns in Kannada since the rules of Kannada morphology are conditioned on gender. Proximal/Distal distinction is shown for pronouns as this is useful for human users of the system, although the computational grammar itself does not make use of this information. Transitivity information is indicated for verbs since this helps in syntactic analysis. Whether the final 'u' vowel in nouns is real or enunciative is indicated since this is required for correct analysis and generation of Kannada noun forms. Our use of the terms tag and tagging is thus much more broad based than mere syntactic or POS tags and tagging. What words are listed or not listed in the dictionary is directly related to what forms are handled by morphology. All exceptions, where we do not have a generic and productive-enough rule that can be implemented within morphology, are listed in the dictionary.

We find that about 40% of all words found in any text are found directly in the dictionary. Hence avoiding morphology in these cases would make the system faster. More importantly, this can also avoid unintended and unexpected analyses. This choice implies that whenever a word is listed in the dictionary, complete information as would have come from morph should be made readily available in the dictionary itself. We will need to give

number and case for nouns, for example, since in a majority of cases, nouns occurring in bare stem form indicate singular number and nominative case in Kannada, both of which involve zero morphology. More importantly, if a word has two or more analyses, all must be given in the dictionary. Thus the Kannada word 'kaaDige' is listed in the dictionary saying it is a noun in singular nominative form as also a singular dative form of a different noun.

At times, there will be clashes between what morphology demands and what syntax and semantics demand. For example, words such as 'guru', 'praje' are human nouns in Kannada but they undergo plural formation as per the rules of non-human nouns. One option is to sub-categorize such nouns and let the grammar use this for correct analysis and generation. Another solution is to let the morphology treat these words are neuter nouns and let syntax look at them as human nouns. In our system, whenever there is a clash of requirements, we generally favour the earlier module since failures early on in a pipe-line architecture can be more devastating than failures at later stages. We make sure that suitable corrections are made as we move on, so that we get the right results at the end.

3.2.3 Morphology

It is unlikely that the human mind uses two different grammars, one for analysis and one for generation. That would be a bad choice even from the point of view of simplicity and efficiency. It is not an intelligent choice at all. Therefore, the same grammar must be usable both for analysis and for generation. In our system the computational grammar at the level of morphology is designed as a Finite State Machine. We use this FSM bidirectionally. The standard Finite State Machine can be and needs to be extended in several ways to facilitate morphological analysis and generation. Let us look at the various extensions have done for our work on Telugu and Kannada.

- Forward and Reverse Processing: The written form of a word is a sequence of morphemes, not a sequence of characters. The dictionary head words are also morphemes. By adding other morphemes as suffixes or prefixes to these, we build full word forms. Kannada and Telugu are basically suffixing languages, prefixes are not used productively. In order to generate a word form, we start with the given root taken from the dictionary and we keep adding suffixes as we pass through the FSM from the start state to the terminal state. The resulting string will be the fully inflected and/or derived word form. For analysis, we can use the same FSM grammar in reverse. We can start with the end state and work backwards towards the start state. This is always possible. Every FSM has a single start state and we can make any FSM to have a single terminal state, by connecting all the terminal states to a newly added state through epsilon arcs. Having a single terminal state makes the algorithm simpler but this is not theoretically essential. We start with the full word form and keep removing suffixes one by one, until we reach the start state by which time we should be left with only the bare stem, which can then be checked with the dictionary. Using the same grammar for both analysis and generation has several advantages. Firstly, the effort needed to develop an analyzer and a generator is reduced significantly. Secondly, this facilitates testing and evaluation. It is always easy to test and evaluate an analyzer, we can simply run it on a large test data set and observe the results. Testing a generator is harder, it requires more of manual effort. If we use the same grammar for both, we can test the analyzer be sure that the generator will also be correct. Thirdly, since there is no evidence to show that we use different grammars for analysis and generation in our minds, this is the natural, normal, intelligent way. Computational efficiency is not affected either.

- Finite State Transducer: Instead of just accepting or rejecting a given string, we can produce some output. Depending upon whether the output bits are attached to the arcs or to the states we have what are called Moore and Mealy machines. In the case of analysis, this facilitates the production of a structural description of the given word form. In the case of generation, this facilitates such a structural description to be given as input, asking for the corresponding word form. This is especially useful in Machine Translation. The source language words need to be analyzed and the target language word forms need to be generated, as per given grammatical features. In our work, we attach output strings to the arcs. All that we have to do is to add another column to the FSM table, indicating the output for each arc. The computational efficiency of FSMs is not affected.

- Adding a Category Field: Inflectional morphology is mainly limited to nouns and verbs in Telugu and Kannada. The suffixes are almost completely exclusive, noun forms and verb forms rarely clash, unlike in English. In English there are very common clashes such as between the plural forms of nouns and third person, singular, present tense forms of verbs. In our languages, the noun and verb inflection grammars are disjoint. However, once we add derivation, it becomes difficult to maintain separate FSMs. Derivation involves adjectives and adverbs too. Therefore, it is better to merge all the aspects of morphological grammar into one FSM. Particular arcs apply only to particular categories and this can be indicated by adding a new field for category. For clitics etc., where any category will do, we can indicate this by using a label such as 'any'. In principle, these FSMs can be broken into separate FSMs, and hence there is no change in the computational efficiency or other properties as long as the categories do not change. The category fields merely helps in selecting the suitable sub-part of the combined FSM.

- Derivation: Derivation involves change in category. To facilitate derivation, we allow two categories on an arc, the initial category and the category after derivation. A word starts off with a particular category and changes colour, so to say, taking a different category and continues to move on the same combined FSM. Thus handling derivation requires very little change to the whole system. Inflection and derivation are not water-tight compartments, there is a seem-less joint and the simplicity of this approach is actually stands testimony to this view. Computational efficiency does not change whether we use on big FSM or several smaller ones, addition of linear time algorithms will still give us a linear time algorithm in the worst case. Hence derivation does not change the overall computational efficiency of the system.

- saMdhi: The biggest change we will need is to introduce a saMdhi process while adding suffixes. Telugu and Kannada are highly inflectional as also agglutinative, there are significant morph-phonemic changes at the juncture when morphemes combine. In general saMdhi may involve insertion, substitution, as also deletion of characters, hence it is tricky to establish the computational efficiency bounds.

- Loop Control: There are loops in the FSM and this causes no problem at all in analysis, as analysis is guided by the morphemes actually found in a given string. For generation, however, this can pose a problem, the program can get into infinite loops. This does not happen when we ask for a particular word form to be generated from a given set of features, this happens only when we try to generate all forms of a given stem or root. Although a FSM does not have memory, here we may add a counter to prevent infinite looping. Generating all forms of a given stem is not a natural process for human beings, this is simply a technological

feasibility. Therefore, this is not a serious problem as far as the theory is concerned. Further, after a closer study, we can eliminate loops by expanding out on the various possibilities.

- External saMdhi and Compounds: Compounds also involve saMdhi. When there is saMdhi between two or more words, one thing that needs to be done is to locate the place where we can cut and separate the components. Blindly trying at each byte position is no good. We cannot even work with akShara-s, saMdhi is basically a morpho-phonemic process, while akShara-s are purely units of writing. In our approach, we start analyzing the compound word or a sequence of words conflated by external saMdhi as if it is a single word. We work leftwards from the right end. Once we reach the start state of the network, we check the remaining string against the dictionary. In the case of external saMdhi or compound, we will not find a match. At this stage, we will need to use heuristics to hypothesize possible places to cut. An essential requirement of compounding is that there be a modifier-modified relationship between the components. saMdhi usually avoids this, saMdhi usually occurs between categories that are grammatically unrelated, for example, a pronoun and an adverb. Certain kinds of compounds occur more regularly. For example, n-v compounds are very frequent, here the verb should be one of the few known verbalizers, not any arbitrary verb. Similarly n-n compounds are more common.

We give below a sample of the Finite State Grammar for Kannada. The current FSM has 405 transitions.

```
0 3 n gaLu PL
0 3 n epsilon SL

3 95.5 n annu ACC
3 95.5 n iMda ABL

17 20 v id PAST
17 21 v utt PRES
17 22 v uv FUT

20 95 v enu P1.MFN.SL
20 95 v evu P1.MFN.PL
20 95 v e P12.MFN.SL
20 95 v ir(i) P2.MFN.PL
20 95 v anu P3.M.SL

17 100 v ali OPT
17 95 v ooNa HORT.P1.PL

17 25 v al(u) INF

25 26 v ee CLIT.ee
25 26 v uu CLIT.uu
25 100 v uu CONJ
25 95 v illa PAST.NEG
25 26 v epsilon NULL

22 4 v>n udu GRND

20 50      v>adj   (a) RP

94 97 any     uu CLIT.uu
94 97 any     oo CLIT.oo
94 97 any     epsilon NULL
94 100 any     yaa CLIT.INTG

100 0 n>v ennu +V.ennu
100 0 v ennu +V.ennu

100 0 n>v aagu +V.aagu
100 0 n>v iru +V.iru
```

The FSM is also stored in a single plain text file. Zero is the assumed start state and 100 is the only terminal state. Each transition shows the starting state, the ending state, the category, the suffix and the feature bundle associated with it. Note how categories can change in derivational morphology. A plus sign prefix indicates that there is external saMdhi at this point. Parenthesized parts

indicate optional dropping, this is one way to handle variations in internal saMdhi that defy the existing standard rules of saMdhi formation. Many suffixes, especially the clitics, have multiple roles with multiple meanings. For example, 'aMte' added to nouns can indicate comparative, the same clitic added to verbs may mean several different things as exemplified in sentences such as 'raamanu baradaMte taDe', 'raamanige baruvaMte heeLu', 'aLabeeDa magu, naaLe niinee modalu haaDuveyaMe'. The clitic 'oo' may indicate doubt, it can act like an interrogative too. Disambiguation is not within the scope of morphology, we need to go at least to the level of syntax. Therefore, the feature bundles simply indicate the suffix itself, without trying to go further. This is one way explicit ambiguities are cut down.

Let us now look at some examples of the output of morphological analyzer:

```
manege<mane:N-COM-COU-N.SL-NOM:
%n-SL-DAT->

maaDuttaane<maaDu:N-COM-COU-N.SL-NOM||
V-TR1:%v-PRES-P3.M.SL->

maaDidare<maaDu:N-COM-COU-N.SL-NOM||
V-TR1:%v-PAST-COND->

maaDabeekaagibaMdaaga<maaDu:
N-COM-COU-N.SL-NOM||V-TR1:
%v-INF-CMPL-AUX.aagu-CJP.PAST
-AUX.baru-PAST-RP-%adj-CLIT.aaga->

maaDibiTTaraMtaa<maaDu:
N-COM-COU-N.SL-NOM||V-TR1:
%v-CJP.PAST-AUX.biDu-PAST-P3.MF.PL
-CLIT.aMte-CLIT.INTG->
```

3.2.4 The Bridge Module and Tagging

Here the bits of information obtained from the dictionary and morphology are combined to generate final tags. For the examples shown above, the tagger will produce the following tags:

```
manege||mane||N-COM-COU-N.SL-DAT

maaDuttaane||maaDu||V-TR1-PRES-P3.M.SL

maaDidare||maaDu||V-TR1-PAST-COND

maaDabeekaagibaMdaaga||maaDu||
V-TR1-INF-CMPL-AUX.aagu-CJP.PAST
-AUX.baru-PAST-RP-adj-CLIT.aaga

maaDibiTTaraMtaa||maaDu||V-TR1
-CJP.PAST-AUX.biDu-PAST-P3.MF.PL
-CLIT.aMte-CLIT.INTG
```

It may be noted that noun verb ambiguities have been resolved by morphology. The bridge module has a challenging task since it has to iron out all differences between the lexicon and morphology as also between overt structure and meaning.

We call the complete labels such as N-COM-COU-N.SL-DAT as single tags. Tags are made up of tag elements such as NOM and N.SL. Tag elements may in turn be made up of tag atoms. N.SL is one tag element with two tag atoms. The first field is always the main category and the subsequent one or two fields may indicate sub-categories. Rest of the fields indicate grammatical features.

It is also important to observe that the same overall scheme pervades all levels of analysis, starting from the dictionary and going all the way up to the syntactic parser.

3.2.5 The Parser

We choose to perform dependency parsing as we believe a dependency structure is more revealing and more directly connected with the meaning than other possible parse structures. Expectations from the verbs form the top-down force and the suitability of noun phrases act as the bottom-up force, and together we have a

constraint satisfaction solution. Non-arguments are filled up later, as appropriate. Currently we have only small, sample systems for the purposes of demonstration alone. More work is on.

Thematic roles in a dependency parse are filled by whole noun groups and recognition of noun groups is thus a sub-task of parsing. This is the essence of what is commonly known as chunking. We use Finite State Grammars for np-chunking. See [2] for more details.

Now that we have morphological analysers and generators, as also syntactic parsing/generation system for Kannada and Telugu, we can demonstrate automatic translation between Kannada and Telugu.

4 Experiments and Results

Here we shall give details of the system we have built for Telugu. The status of Kannada is similar.

We have performed POS tagging experiments on various corpora. F1 is a randomly selected file from TDIL corpus. F2 is set of sentences extracted from the TDIL corpus containing about 15,000 most frequent words from this corpus. These most frequent words have a great significance not just because they account for more than 60% of the whole corpus but also because they include the most confusing items from the point of view of lexicon, morphology and tag assignment. The rest of the words forming the bulk of the corpus are the simplest as far as tagging is concerned. In order to be sure that there is no over-fitting for any particular data set, we have next attempted tagging files F3 and F4 from the Eenadu Telugu daily newspaper. The table below shows the performance of the system as on date. Here D indicates the number of words directly found in the dictionary, M indicates the number of words analyzed by the morph, D-AMB is the number of words found in the dictionary and having more than one tag, M-AMB is the number of words analyzed by morph and having more than one tag. UNK indicates the number of words that remain untagged.

It may be observed that about 40% of the word forms are directly found in the dictionary and 50-60% of the words are analyzed by morph. Unless the texts contain a large percentage of proper nouns (ex. F4), only some 5-10% of the words remain untagged. Most of the unknown words are loan words, proper nouns, or words involving external saMdhi or compounds. Further work on dictionary and morphology can reduce the number of untagged words in future. Of the tagged words, only about 10% of the words have more than one tag assigned. The maximum number of tags that can get assigned is 4 but this is a very rare case. Even getting 3 analysis is a rare phenomenon. Only a few typical kinds of ambiguities occur most of the times and preliminary studies have shown that a majority of them will automatically get resolved through chunking and parsing. We may not need a statistical approach to resolve these but if one wishes, we can easily tag the whole corpus using our system and create training data from that. One can also try a variety of heuristic rules to resolve the remaining ambiguities.

Upon careful observation of the tagged samples, we find that most words are tagged correctly. In order to be doubly sure, we again tagged 252 sentences selected from various grammar books including a wide variety of sentence structures. these sentences include 1278 tokens. All of these get tagged. Only two words were tagged incorrectly. Only 140 words are assigned more than one tag.

5 Conclusions

In this paper we have presented a new approach to tagging based on our theory of language, grammar and computation. We have demonstrated the viability and merits of our ideas through actually developed systems for Kannada and Telugu. More work is on.

6 Bibliography

1 Kavi Narayana Murthy,"Language, Grammar and Computation", forthcoming.

2 Kavi Narayana Murthy, Srinivasu Badugu,"On Tagging of Natural Language Texts", forthcoming.

3 Kavi Narayana Murthy, "A Network and Process Model for Morphological Analysis/Generation", ICOSAL-2, The Second International Conference on South Asian Languages, 9-11 January, 1999, Punjabi University, Patiala, India

4 G Bharadwaja Kumar, Kavi Narayana Murthy and B B Chaudhuri, "Statistical Analysis of Telugu Text Corpora", International Journal of Dravidian Languages, 36:2, June 2007, pp 71-99

5 CH. Narsinga Rao and Kavi Narayana Murthy, "On the Design of a Hierarchical POS Tagset for Telugu ", MTech thesis, Department of Computer and Information Sciences, University of Hyserabad, 2008

6 K. Anil Kumar, "Morphological Analysis of Telugu Words", MTech thesis, Department of Computer and Information Sciences, University of Hyderabad, 2003

7 Sankaran Baskaran, "Hindi Part of Speech Tagging and Chunking",Proceedings of NLPAI Machine Learning Workshop on Part of Speech Tagging and Chunking for Indian languages, IIIT Hyderabad, Hyderabad, India, 2006

8 Rama Sree R.J, Umamaheshwar Rao G and Madhu Murthy K.V, "Assessment and Development of POS Tagset for Telugu", The 6th Workshop on Asian Language Resources, IJCNLP, IIIT Hyderabad, Hyderabad, India, 11-12 January, 2008

9 Umamaheshwar Rao G, "Compound Verb Formation in Telugu", National Workshop-cum-Seminar on Lexical Typololgy, Telugu University, Hyderabad, 1996

N (NOUN)	
	COM(Common)
	PRP(Proper)
	-PER(Personal)
	-LOC(Location)
	-ORG(Orgzn.)
	-OTH(Others)
	LOC(Locative)
	CARD(Cardinal)
PRO (Pronoun)	
	PER(Personal)
	INTG(Interrogative)
	REF(Reflexive)
	INDF(Indefinite)
ADJ (Adjective)	
	DEM(Demonstrative)
	QNTF(Quantifying)
	ORD(Ordinal)
	ABS(Absolute)
ADV (Adverbs)	
	MAN(Manner)
	CONJ(Conjunctive)
	PLA(Place)
	TIM(Time)
	NEG(Negative)
	QW(Question Word)
	INTF(Intensifier)
	POSN(Post-Nominal Modifier)
	ABS (Absolute)
CONJ (Conjunction)	
	SUB(Subordinating)
	COOR(Coordinating)
V (Verb)	
	IN(Intransitive)
	TR(Transitive)
	BI(Bitransitive)
	DEFE(defective)
INTJ (Interjection)	
SYMB (Symbol)	

Table 1: LERC-UoH Tag Set

File	#Sent	#Tok	Dict	Morph	M-AMB	D-AMB	UNK	Ambiguity
F1	365	4910	2186 (45%)	2389 (49%)	158 (3%)	170 (4%)	313 (6%)	328 (7%)
F2	15100	76004	45225 (59%)	31103 (41%)	4917 (6%)	3194 (4%)	20 (0%)	8111 (10%)
F3	33	282	107 (38%)	173 (61%)	15 (5%)	10 (3%)	2 (1%)	25 (9%)
F4	27	237	88 (37%)	99 (42%)	8 (3%)	7 (3%)	50 (21%)	15 (6%)

Table 2: Tagging Performance