CSME 206A – Natural Language & Speech Processing **Spring Semester**

Lecture 19: NLP for Indian Languages

Lecturer: K.R. Chowdhary *: Professor of CS*

19.1 Introduction

High Complexity and diversity of human languages makes automated translation one of the hardest problems in computer science. Yet the job is becoming more important as written text and speech are becoming increasingly digitized, and other driving force is that traditional separations between societies is getting fast dissolved.

Few parts of the globe have as much need to translate from one language to another as does India. According to India's 2001 census, the country has 122 languages, 22 of which are designated as official languages by the government. The top six – Hindi, Bengali, Telugu, Marathi, Tamil, and Urdu – are spoken by 850 million people worldwide.

Now a decades-long effort by researchers is about to bear fruit. A multipart machine translation architecture, *Sampark*, is nearing completion as the combined effort of 11 institutions led by the Language Technologies Research Center at the International Institute of Information Technology in Hyderabad (IIIT-H). The Sampark combines both traditional rules- and dictionary-based algorithms with *statistical machine learning*, and will be rolled out to the public at http://sampark.iiit.ac.in/.

Many Indian languages are derived from *Sanskrit*, which is based on rules set down by *Paanini*, the 4th century B.C. grammarian. Even those Indian languages that are not derived from Sanskrit are structurally similar to others in India. This common underpinning makes the translation from one Indian language to another easier than from, say, German to Chinese. Nevertheless, there are 462 pair-wise translations (counting each direction for a pair) possible among the 22 official Indian languages, so clearly the researchers had to find a generalized approach that could be easily adapted from one language to another.

The chosen method, a transfer-based approach, consists of three major parts: *analyze*, *transfer*, and *generate*. First, the source sentence is analyzed, then the results are transferred in a standard format to a set of modules that turn it into the target language. Each step consists of multiple translation "modules."

An advantage of the three-step approach is that a particular language analyzer, one for Telugu, for example, can be developed once, independent of other languages, and then paired with generators in various other languages, such as Hindi [anthes2010automated].

The 13 major translation modules together form a hybrid system that combines rules-based approaches – where grammar and usage conventions are codified – with statistical-based methods in which the software in essence discovers its own rules through "training" on text tagged in various ways by human language experts, as shown in Fig. 19.1.

19.2 Transfer-Based Approach for Translation

Translation systems for major languages today – from companies like Google and Microsoft, for example – often use statistical approaches based on parallel corpora, huge databases of corresponding sentences in two languages. These systems use probability and statistics to learn by example for which translation of a word or phrase is most likely to be correct. And, they move directly from source language to target language with no intermediate transfer step. "The statistical direct translation approach is, in a sense, the lazy man's approach, because all it requires is that you go and hunt for parallel corpora and you turn the "crank" and you get what you want.

The *transfer-based* approach is much more linguistically motivated, because you try to analyze the sentence and try to arrive at something that is close to a representation of its meaning.

Parallel Corpora The *parallel corpora* are specialized databases consisting of sentences very carefully translated and then mapped one-for-one to their translations. More over, to do a good job of training translation systems, the parallel corpora must be very large –in the billions of sentences. "People are coming to grips with the fact that parallel data are not easy to come by. This is a very specialized kind of data. Indeed, parallel corpora for many Indian language pairs do not exist and cannot easily be built, in part because not much Indian language text has been digitized.

Paaninian Approach Nevertheless, developers at the *Language Technologies Research Center* were able to apply statistical machine learning in a limited way by annotating small *monolingual* corpora and analyzing the tagged text with statistical techniques. So, although machine learning techniques were employed in some of the modules, developers painstakingly developed multi-language dictionaries and codified rules in the *Computational Paaninian Grammar* framework. They also held workshops of experts of all these languages to develop a standard tag set, and then used those tags to annotate the monolingual corpora.

19.3 Process of Machine Translation

In fact most machine translation is not inspiration, but the perspiration. The hard part is building all the required resources, like dictionaries, morphological analyzers, parsers, and generators. The effort that Sampark developers put into language analysis could have a broad impact beyond translating Indian languages. Even the best purely statistical systems can be made more accurate by first doing the types of detailed language analysis employed in Sampark. What one can do in the future is to first do monolingual analysis of one or both sides in paralleled corpora, and then use that to improve the quality of machine learning from the parallel corpora. So what we have done would also be useful if larger parallel corpora became available in future.

Other advantage of the transfer approach, is its generalizability. For example, given a parallel corpus dealing with financial news, and it is trained with millions of sentences of that sort. And later there is need to translate a sports article, it is not going to perform well. But that kind of application domain change has been explicitly anticipated by Sampark's developers. The first version, is general purpose and optimized for tourism-related uses, but it will be made available to large users who wish to customize it for other domains. That would involve building a new domain dictionary, incorporating rules that handle domain-specific grammatical structures, and perhaps retraining some modules such as Part of Speech Tagger and Named Entity Recognizer.

The effort required to make those changes is minimized by building on the existing multilingual dictionary. It is sense- or meaning-based, so that for one domain or language, "bank," for example, would most likely represent a financial institution, but for another it might refer to the edge of a river, Sharma says. The dictionary currently allows translation among nine languages.

The language-translation system has two especially noteworthy attributes. First, the linguistic analysis based on Paanini is "extremely good," he says. It was initially chosen for Indian languages, but we find it is also suitable for other languages. Initially, hard work is needed, he says, in setting it up by developing standards for parts-of-speech tags and dependency tree labels and for figuring out ways to handle unique language constructs.

The second attribute of special note is the system's software architecture. It is an open architecture in which all modules produce output in Shakti Standard Format (SSF). The architecture allows modules written in different programming languages to be plugged-in. Readability of SSF helps in development and debugging because the input and output of any module can be easily seen. "If a module fails to perform a proper analysis, the next module will still work, albeit in a degraded mode. So the system never gives up; it always tries to produce something."

An automated system for translating one Indian language to another, sampark (Fig. 19.1), is a hybrid system consisting of traditional rules-based algorithms and dictionaries and newer statistical machine-learning techniques. It consists of three major parts and 13 modules arranged in a pipeline.

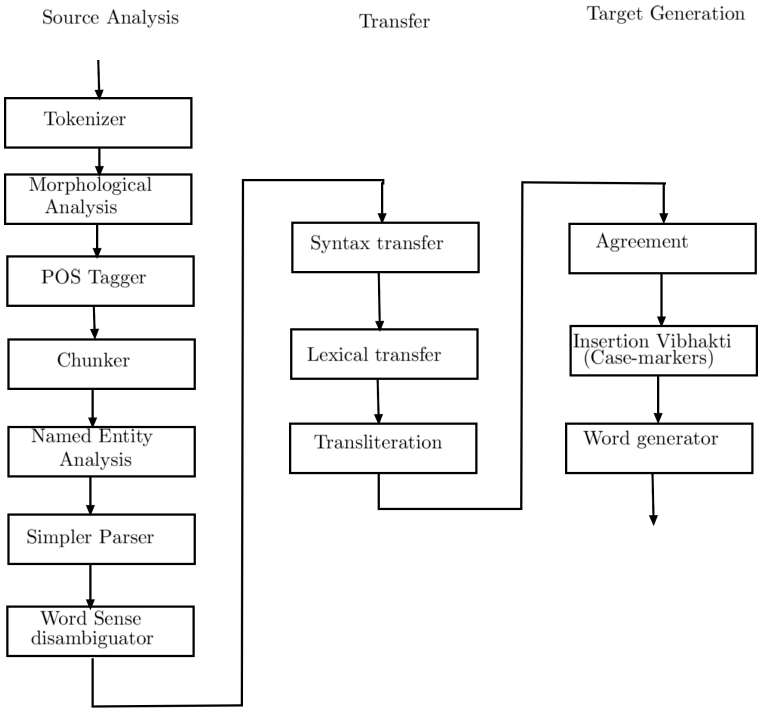

Figure 19.1: Working of Smpark module

Following is the explanation of various blocks of Sampark.

Source Analysis:

- *Tokenizer:* Converts text into a sequence of tokens (words, punctuation marks, etc.) in shakti standard Format.
- *Morphological analyzer:* Uses rules to identify the root and grammatical features of a word. splits the word into its root and grammatical suffixes.
- *Part of speech tagger:* Based on statistical techniques, assigns a part of speech, such as noun, verb or adjective, to each word.
- *Chunker:* Uses statistical methods to identify and tag parts of a sentence, such as noun phrases, verb groups, and adjectival phrases, and a rule base to give it a suitable chunk tag.
- *Named entity recognizer:* Identifies and tags entities such as names of persons and organizations.
- *Simple parser:* Identifies and names relations between a verb and its participants in the sentence, based on the Computational paninian Grammar framework.
- *Word sense disambiguation:* Identifies the correct sense of a word, such as whether "bank" refers to a financial institution or a part of a river.

Transfer:

- *Syntax transfer:* Converts the parse structure in the source language to the structure in the target language that gives the correct word order, as well as a change in structure, if any.
- *Lexical transfer:* Root words identified by the morphological analyzer are looked up in a bilingual dictionary for the target language equivalent.
- *Transliteration:* Allows a source word to be rendered in the script of the target language. Useful in cases where translation fails for a word or a chunk.

Target Generation:

- Performs gender-number-person agreement between related words in the target sentence.
- *Insertion of Vibhakti:* Adds post position and other markers that indicate the meanings of words in the sentence.
- *Word generator:* Takes root words and their associated grammatical features, generates the appropriate suffixes and concatenates them. Combines the generated words into a sentence.

19.4 Paninian Grammar

Pānanini's grammar (ca. 350 BC) seeks to provide a complete maximally concise, and theoretically consistent analysis of Sanskrit grammatical structure. It is the foundation of all traditional and modern analysis of Sanskrit, as well as having great historical and theoretical interest. Modern linguistics acknowledges it as the most complete generative grammar of any language yet written, and continues to adopt technical ides from it.

The grammar is based on the spoken language (*bhāsā*) of Pānini's time, and also give rules on vedic usage and on regional variants. Its optional rules distinguish between preferable and marginal forms. The grammar consists of four components:

(a) *Astādhyāyī* : a system of about 4000 grammatical rules.

(b) *Śivasūtras* : the inventory of phonological segments, partitioned by markers (anubandhas) to allows abbreviation for classes of segments to be formed by a technique described below (*prayahārās*).

(c) *Dhātupātha* : a list of about 2000 verbal roots, with subclassification and diacritic markers encoding their morphological and syntactic properties.

(d) *Ganāpātha* : an inventory of classes of lexical items idiosyncratically subject to various rules.

The rules of *Astādhyāyī* make reference to classes defined on the elements in the other three components by means of conventions spelled out in the *Astādhyāyī* itself. Thus, while none of the components is intelligible in isolation, together they constitute a complete integrated system of grammar.

There are also various peripheral adjuncts to the system. The most important of these are the *Unādisūtras*, which extend the Pāninian technique to analyze unproductive and irregularly formed derivatives from roots. Though mentioned in a few rules of *Astādhyāyī*, many of the words they derive are treated as underived there, and they are probably post-Pāninian at least in their present form.

Pānini, Kātyāyana, and Patañjali's are referred to as three sages and considered authoritative by later grammarians.

The *Astādhyāyī* is formulated in morphologically, syntactically, and lexically regimented from Sanskrit. The maximize it being concise with minimum ambiguity, rules are compressed by syntactically omitting repeated expressions from them, according to the procedure modeled on natural language syntax (*anuvrtti*). The rules can be divided into four types: (a) definitions, (*saṃjñā*), (b) metarules (*paribhāsā*), (c) headings (*adhikāra*), and (d) operational rules (*vṛtti*).

19.5 Paninian approach to Natural language Processing

It was first thought that machine translation (MT) of one language to another should be easy matter once one has compiled dictionaries and obtained mathematical representation of the grammars of the languages in question. It was believed that actual translation would proceed by replacing the words of source language by those of target language equivalents, and then rearranging and modifying these new words according to grammar of the target language. But, it was soon found that this task was not easy, as a word can have several equivalents, and the correct one can be decided only by context. The mathematical representation of the grammar of a natural language was also given up as an intractable problem.

19.5.1 Paanini's Grammar

Paanini's grammar Astadhyayi (Eight chapters) deal ostensibly with the Sanskrit language; however, it represents the framework for a universal grammar that may (and possibly does) apply to any language. His book consists a little over 4000 rules and aphorism. Paanini's grammar attempts to completely describe Sanskrit the spoken language of its time.

Panini's grammar begins with meta-rules, or rules about rules. To facilitate his description he establishes a special language, or meta-language. This is followed by several sections on how to generate words and sentences starting from roots, as well as rules for transformation of structure. The last part of the grammar is a one-directional string of rules, where a given rule in the sequence ignores all the rules that follow. Panini also uses recursion by allowing elements of earlier rules to recur in later rules. This anticipates in form

and spirit by more than 2500 years the idea of a computer program. The structure of this part of Panini's grammar should rightly be termed the Panini Machine.

The Panini's system a finite set of rules is enough to geenerate an inifinity of sentences. The algebraic character of Panini's rules was not appreciated in the West until late 1950s when a similar generative structure as discussed by Noam Chomsky and others. Before this, in the nineteenth century, Panini's analysis of root and suffixes and his recognition of ablaut had led to the founding of the subjects of comparative and historical linguistics.

Despite similarities between Paninian and modern generative grammars, there exits striking difference as well. Some of these differences are related to the nature of language under study: sanskrit and modern Europian languages. Furthermore, the contemporary evaluation of Panini is still going one and has been slowed by the fact that the original of this grammar is inaccessible to mots linguists and computer scientists.

We define an approach to language processing as being Paninian if it uses the following:

1. Root and affix analysis.
2. Linear strings of rules and analysis by rule sequence.
3. Analysis by functional structure.
4. An exhaustive description.

Because the current grammar for English does not satisfy condition (3), we would constrain the allowed structures so as to fit the capabilities of the grammars.

19.5.2 language Ambiguities

Several ambiguities make machine translation of languages a difficult task. These ambiguities need to be resolved, if at all possible, in various steps to obtain a translation. Each kind of ambiguity is addressed separately in a sequence of steps that constitute the usual form of a computer-based understanding system.

Lexical Ambiguity This type ambiguity arises when a single word has tow or more different meanings, all of which are potentially valid. Consider "Saty away from the range," which could be advice to keep away from either the stove or the meadow. "The Court was packed" is a more complex eample: "the court" may refer to a judicial or royal court, or a rectangular or open space, and "packed" might refer to a biased composition or a crowding by people.

Structural Ambiguity One source of structural ambiguity is the ways in which words in a sentence may be combined into phrase and then interpreted. Thus, in "He saw the crane fly outside." one might be referring to the crane fly (a long-legge, two-winged fly) or to a bird known as a crane, flying. Other examples of ambiguity are: "My friend came home late last night" and "Flying kites can be tricky." yet another kind of structural ambiguity occurs when the sentence has a unique grammatical structure but still allows different meanings because of different underlying "deep structures." For example, "The policeman's arrest was illegal" does not tell us who was arrested – the policeman or some one else. Another example is "The leopard was spotted."

Semantic Ambiguity An example of this kind of ambiguity is the sentence "I like to eat brown grapes." This could either mean that the speaker liked a particular bunch of grapes in front of him, or that he merely expressed a preference for brown grapes.

Pragmatic Ambiguity This ambiguity is related to the context of the sentence. Thus, in "She put the brick in the washer and spoiled it," the meaning would be different depending on whether the brick was made of metal or wax! Similarly, the meaning of "John loves his wife and so does Bill" is ambiguous only of it is known that Bill is bachelor.

Such difficulties are inherent in English but are not fundamental to all natural languages. However, the *Shastric* (scientific) Sanskrit is one natural language that appears to be particularly precise.

19.6 Analysis of Sanskrit Language

A language may be described at several hierarchically organized levels. At the lowest level a spoken language may be characterized in terms of terms of elementary sounds, the study of which is called *phonology*. The natural languages are characterized by 30-80 phonemes. English is usually described with about 40 phonemes and Sanskrit is described in term sof 48 phonemes, each of them is represented by a unique symbol of IPA (international phonetic alphabet).

The study of the next level of linguistic analysis is termed as *morphology*. Simple words as well as inflectional sounds such as ploral endings or prefixes and suffixes that convey meaning are *morphemes*. Thus, a system might process "unknowlingly" by finding its root fro "know" and then determining the change in meaning affected by each of the additional morphemes "un-", "-ing", and "-ly."

The syntax deals with the manner in which the meaningful constituents are put together to form an utterance or sentence. A parse-tree represents this structure. Some sentences can be parsed in many different ways, leading to different meanings. For exaple, "traffic jams were caused by slow trucks and buses carrying heavy goods" may be parsed in at least in four different ways. The "slow" may qualify trucks only or or trucks and buses; similarly, "heavy loads" may qualify buses only, or both trucks and buses. The rules of this grammar may be:

$$
\begin{aligned}
S &\rightarrow NP \;\; VP \\
NP &\rightarrow Adj \;\; N \\
NP &\rightarrow Det \;\; N \;\; PP \\
PP &\rightarrow prep \;\; NP
\end{aligned}
$$

A collection of such rules is one way to generate the sentences.