NLP Assignment

Name:- Sumit Porwal

Enrollment No.:-EN18CS301275

Class:-CS-E Section

Explain

1. Entropy :-

Entropy or self-information is the average uncertainty of a single random variable.Entropy is a slippery concept in physics, but is quite straightforward in information theory. Suppose you have a process (like a language *L* that generates words). At each step in the process, there is some probability *p* that the thing that happened (the event) was going to happen. The amount of **surprisal** is –log(*p*) where the logarithm is taken in any base you want (equivalent to changing units). Low probability events have high surprisal. Events that were certain to happen (*p*=1) have 0 surprisals. Events that are impossible (*p*=0) have infinity surprisal.

The entropy is the expected value of the surprisal across all possible events indexed by *i*:

$$H(p) = -\sum_i p_i \log p_i$$

So, the entropy is the average amount of surprise when something happens.

Entropy in base 2 is also optimal number of bits it takes to store the information about what happened, by Claude Shannon's source coding theorem. For example if I told you that a full-length tweet of 280 characters had an entropy of 1 bit per character, that means that, by the laws of mathematics, no matter what Twitter does, they will always have to

have 280 bits (35 bytes) of storage for that tweet in their database. (In practice of course, they have to have quite a bit more).

In the context of our language model, we'll have to make one tweak. Given that we are interested in sentences *s* (sequences of events) of length *n*, we'll define the entropy rate per word (event) as:

$$H_n(L) = -\frac{1}{n}\sum_{s \in L} L(s) \log L(s)$$

where the sum is over all sentences of length *n* and L(s) is the probability of the sentence. Finally, a technical point: we want to define the entropy of the language L (or language model M) regardless of sentence length *n*. So finally we define

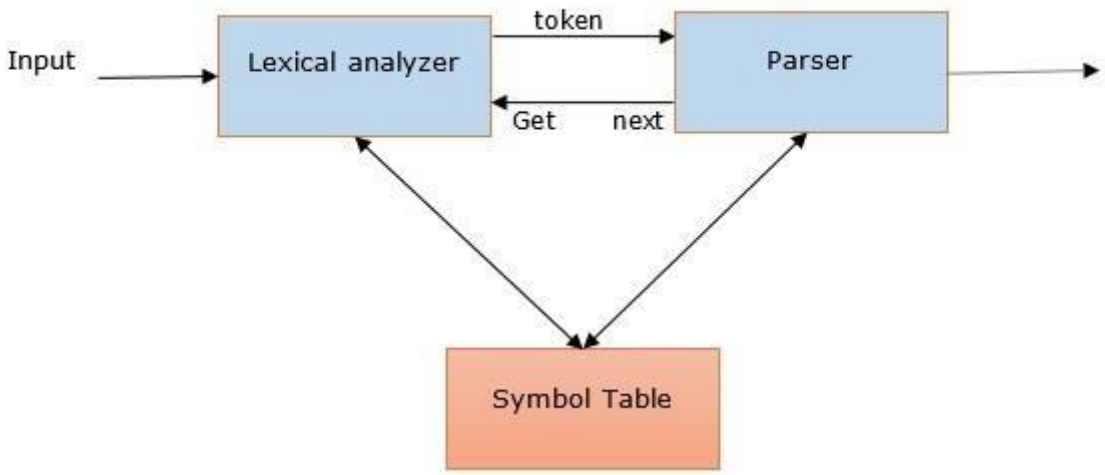
$$H(L) = \lim_{n \to \infty} -\frac{1}{n}\sum_{s \in L} L(s) \log L(s)$$

2. Parsing :-

The word 'Parsing' whose origin is from Latin word **'pars'** (which means **'part'**), is used to draw exact meaning or dictionary meaning from the text. It is also called Syntactic analysis or syntax analysis. Comparing the rules of formal grammar, syntax analysis checks the text for meaningfulness. The sentence like "Give me hot ice-cream", for example, would be rejected by parser or syntactic analyzer.

In this sense, we can define parsing or syntactic analysis or syntax analysis as follows −

It may be defined as the process of analyzing the strings of symbols in natural language conforming to the rules of formal grammar.



We can understand the relevance of parsing in NLP with the help of following points −

- Parser is used to report any syntax error.
- It helps to recover from commonly occurring error so that the processing of the remainder of program can be continued.
- Parse tree is created with the help of a parser.
- Parser is used to create symbol table, which plays an important role in NLP.
- Parser is also used to produce intermediate representations (IR).

Deep Vs Shallow Parsing

Deep Parsing	Shallow Parsing
In deep parsing, the search strategy will give a complete syntactic structure to a sentence.	It is the task of parsing a limited part of the syntactic information from the given task.
It is suitable for complex NLP applications.	It can be used for less complex NLP applications.
Dialogue systems and summarization are the examples of NLP applications where deep parsing is used.	Information extraction and text mining are the examples of NLP applications where deep parsing is used.

It is also called full parsing.	It is also called chunking.

Various types of parsers

As discussed, a parser is basically a procedural interpretation of grammar. It finds an optimal tree for the given sentence after searching through the space of a variety of trees. Let us see some of the available parsers below
:-

Recursive descent parser

Recursive descent parsing is one of the most straightforward forms of parsing. Following are some important points about recursive descent parser −

- It follows a top down process.
- It attempts to verify that the syntax of the input stream is correct or not.
- It reads the input sentence from left to right.
- One necessary operation for recursive descent parser is to read characters from the input stream and matching them with the terminals from the grammar.

Chart parser

Following are some important points about chart parser −

- It is mainly useful or suitable for ambiguous grammars, including grammars of natural languages.
- It applies dynamic programing to the parsing problems.
- Because of dynamic programing, partial hypothesized results are stored in a structure called a ‘chart’.
- The ‘chart’ can also be re-used.

Shift-reduce parser

Following are some important points about shift-reduce parser − □ It follows

a simple bottom-up process.

- It tries to find a sequence of words and phrases that correspond to the right-hand side of a grammar production and replaces them with the left-hand side of the production.
- The above attempt to find a sequence of word continues until the whole sentence is reduced.
- In other simple words, shift-reduce parser starts with the input symbol and tries to construct the parser tree up to the start symbol.

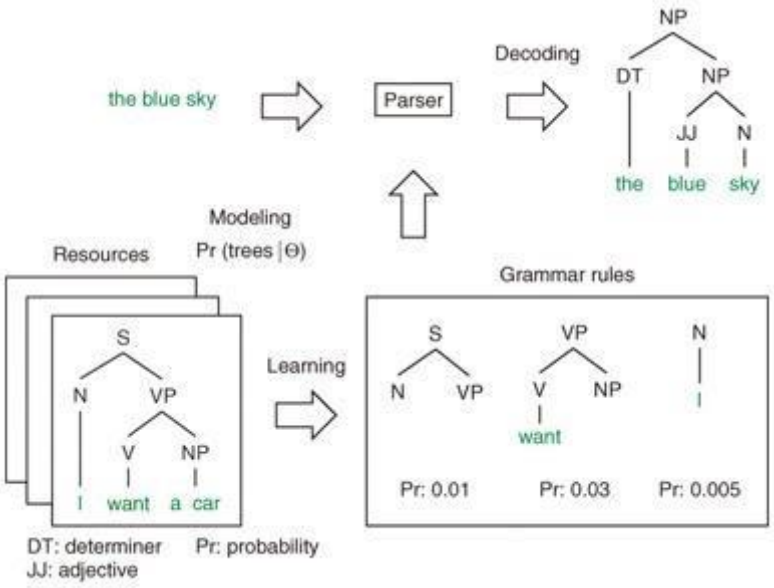
Regexp parser

Regexp parsing is one of the mostly used parsing technique. Following are some important points about Regexp parser −

- As the name implies, it uses a regular expression defined in the form of grammar on top of a POS-tagged string.
- It basically uses these regular expressions to parse the input sentences and generate a parse tree out of this.

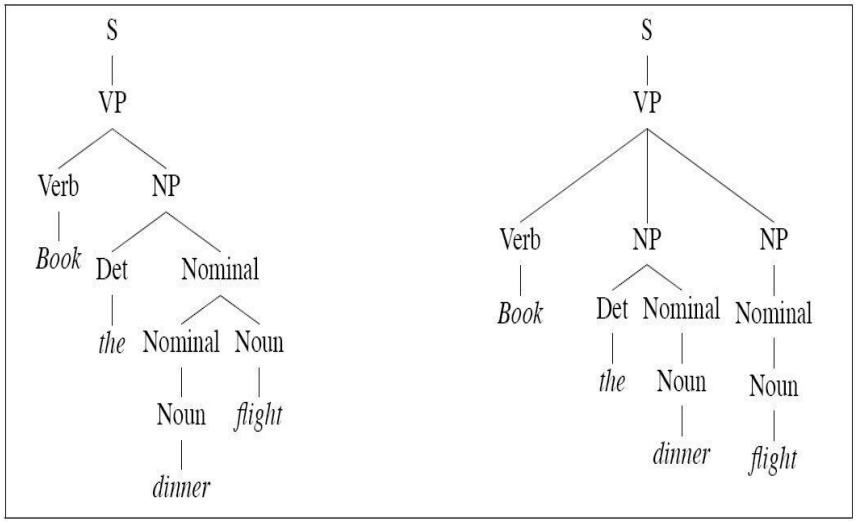
3. Statistical parsing :- Statistical parsing is a group of parsing methods within natural language processing. The methods have in common that they associate grammar rules with a probability. Grammar rules are traditionally viewed in computational linguistics as defining the valid sentences in a language. Within this mindset, the idea of associating each rule with a probability then provides the relative frequency of any given grammar rule and, by deduction, the probability of a complete parse for a sentence. (The probability associated with a grammar rule may be induced, but the application of that grammar rule within a parse tree and the computation of the probability of the parse tree based on its component rules is a form of deduction.) Using this concept, statistical parsers make use of a procedure to search over a space of all candidate parses, and the computation of each candidate's probability, to derive the most probable parse of a sentence. The Viterbi algorithm is one popular method of searching for the most probable parse.

As an example, think about the sentence "The can can hold water". A reader would instantly see that there is an object called "the can" and that this object is performing the action 'can' (i.e. is able to); and the thing the object is able to do is "hold"; and the thing the object is able to hold is "water". Using more linguistic terminology, "The can" is a noun phrase composed of a determiner followed by a noun, and "can hold water" is a verb phrase which is itself composed of a verb followed by a verb phrase. But is this the only interpretation of the sentence? Certainly "The can can" is a perfectly valid noun-phrase referring to a type of dance, and "hold water" is also a valid verb-phrase, although the coerced meaning of the combined sentence is non-obvious. This lack of meaning is not seen as a problem by most linguists (for a discussion on this point, see Colorless green ideas sleep furiously) but from a pragmatic point of view it is desirable to obtain the first interpretation rather than the second and statistical parsers achieve this by ranking the interpretations based on their probability.



4. Probabilistic parsing :-

The higher the weight of a parse or subparse, the more confident we are that it is correct. This is called weighted parsing. If the weights are chosen to define a probability distribution over parses or strings, this may also be called probabilistic parsing. Disambiguation is achieved by computing the parse with the highest



weight or, where appropriate, highest probability. The simplest form of probabilistic parsing relies on an assignment of probabilities to individual rules from a context-free grammar.

The success of probabilistic parsing is due its flexibility and scalability, in contrast to approaches to disambiguation that rely on much deep knowledge of language.

5. Treebank :- A treebank is a collection of texts in which sentences have been exhaustively annotated with syntactic analyses. The term itself, pioneered by the Penn Treebank for English, draws from the traditional representation of sentences as upside-down trees, whose leaves are the words in the sentence.Treebanks have become valuable resources in natural language processing (NLP) in recent years (Abeillé, 2003). A treebank is a collection of syntactically annotated sentences in which the annotation has been manually checked. The name derives from the fact that syntactic descriptions of sentences often come in the form of tree structures, in particular constituent trees. But treebank annotation has also been done in the framework of dependency grammar and recent annotation has also exceeded syntax towards semantic features such as predicate-argument structures or word senses. For example, annotated treebank data has been crucial in syntactic research to test linguistic theories of sentence structure against large quantities of naturally occurring examples.

Use Of Treebanks

In theoretical linguistics and psycholinguistics is interaction evidence. A completed treebank can help linguists carry out experiments as to how the decision to use one grammatical construction tends to influence the decision to form others, and to try to understand how speakers and writers make decisions as they form sentences. Interaction research is particularly fruitful as further layers of annotation, e.g. semantic, pragmatic, are added to a corpus. It is then possible to evaluate the impact of non-syntactic phenomena on grammatical choices.

Types of TreeBank Corpus

Semantic and Syntactic Treebanks are the two most common types of Treebanks in linguistics. Let us now learn more about these types −

Semantic Treebanks

These Treebanks use a formal representation of sentence's semantic structure. They vary in the depth of their semantic representation. Robot Commands Treebank, Geoquery, Groningen Meaning Bank, RoboCup Corpus are some of the examples of Semantic Treebanks.

Syntactic Treebanks

Opposite to the semantic Treebanks, inputs to the Syntactic Treebank systems are expressions of the formal language obtained from the conversion of parsed Treebank data. The outputs of such systems are predicate logic based meaning representation. Various syntactic Treebanks in different languages have been created so far. For example, **Penn Arabic Treebank, Columbia Arabic Treebank** are syntactic Treebanks created in Arabia language. **Sininca** syntactic Treebank created in Chinese language. **Lucy, Susane** and **BLLIP WSJ** syntactic corpus created in English language.