*i*Journals: International Journal of Software & Hardware Research in Engineering (IJSHRE)
ISSN-2347-4890
Volume 9 Issue 6 June 2021

A Brief Concept of Natural Language Processing: The Chomsky Hierarchy and Hidden Markov Models

Authors: Shin-Jye Lee[1]; Ching-Hsun Tseng[2]; Ying-Yi Chou[3]

Affiliation: Institute of Management of Technology, National Chiao Tung University, Taiwan[13];

Department of Computer Science, University of Manchester, United Kingdom[2]

E-mail: camhero@gmail.com[1];hank131415go61@gmail.com[2];shiaopooh1811@gmail.com[3]

DOI: 10.26821/IJSHRE.9.6.2021.9614

ABSTRACT

Nowadays Natural Language Processing (NLP) has been popularly applied in the real world, and a lot of NLP applications can be seen in our daily life. Basically, NLP plays an important role in promoting an advanced development to Artificial Intelligence, and the primary methods contributing a high-performance NLP comprise Chomsky Hierarchy and Hidden Markov Models. Furthermore, these methods become an increasing issue to carry out a more advanced application of NLP.

Keywords: Natural Language Processing

1. INTRODUCTION

Nowadays Natural Language Processing (NLP) has been popularly applied in the real world, and a lot of NLP applications can be seen in our daily life. Basically, NLP plays an important role in promoting an advanced development to Artificial Intelligence, and the primary methods contributing a high-performance NLP comprise Chomsky Hierarchy and Hidden Markov Models. Furthermore, these methods become an increasing issue to carry out a more advanced application of NLP. The Chomsky Hierarchy, a series of four classes of formal languages, a hierarchy of grammars, ismost commonly used in computational linguistics.Hidden Markov Model (HMM) is a component of a modern speech recognizer, and it is an advanced version of Markov Model.

2. THE CHOMSKY HIERARCHY

The Chomsky Hierarchy, a series of four classes of formal languages, a hierarchy of grammars, ismost commonly used in computational linguistics. It includes four kinds of grammars, and the set of languages describable by grammars of greater power subsuming the set of languages describable by grammars of lesser power. This is the form of The Chomsky Hierarchy:

Type	Common Name	Rule Skeleton	Linguistic Example
0	Turing Equivalent	$\alpha \rightarrow \beta$, s.t. $\alpha \neq \varepsilon$	ATNs
1	Context Sensitive	α A $\beta \rightarrow \alpha\ \gamma\ \beta$, s.t. $\gamma \neq \varepsilon$	Tree-Adjoining Grammars
2	Context Free	A$\rightarrow \gamma$	Phrase Structure Grammars
3	Regular	A→xB or A→x	Finite State Automata
Where A is a single non-terminal α , β , and γ are arbitrary strings of terminal and non-terminal symbols x is an arbitrary string of terminal symbols.			

1) **Type 0, Unrestricted Grammars,** have no restrictions on the form of their rules, except that the left-hand side cannot be the empty string ε. Any (non-null) string can be written as any other string (or as ε). Type 0 grammars consist of all recursively enumerable languages, and a Turing Machine can enumerate those strings.

2) **Type 1, Context-Sensitive Grammars,** have rules that rewrite a non-terminal symbol A in the context αAB→αγβ or in the form A→γ/α__β, where A is a non-terminal and α,β, and γ are arbitrary words with γ nonempty. If γ was allowed to be empty then any Type 0 languages of the Chomsky hierarchy could be generated. To derive the empty word, a production S→Λ must also be included, with S not occurring in the right-hand side of any production. The term context-sensitive refers to the fact that A can be rewritten to γ only in the context α…β. In a length-increasing grammar each production has a right-hand sides at least as long as its left-hand side. Clearly any context-sensitive grammar is length-increasing, but it can also be shown that any length-increasing grammar is equivalent to a context-sensitive one. Context-sensitive grammars are a class of phrase-structure grammar. By the way, a linguistic model that is known to be context-sensitive is Tree-Adjoining Grammar.

3) **Type 2, Context-Free Grammars**, the most commonly used mathematical system for modeling constituent structure in English and other languages, are also called Phrase-Structure Grammars. Also, the formalism is equivalent to what is also called Backus-Naur Form or BNF. A context-free grammar consists of a set of rules or productions, and each of which expresses the ways that symbols of the language can be grouped and ordered together. The symbols that are used in context-free grammars are divided into two classes. The symbols corresponding to words in the language are called terminal symbols, and the symbols expressing clusters or generalizations of these are called non-terminal. Context-free rules allow any single non-terminal to be rewritten as any string of terminals and non-terminals. A non-terminal may also be rewritten as ε. In each context-free rule, the item to the right of the arrow (→) is an ordered list of one or more terminals and non-terminals, while to the list of arrow is a single non-terminal symbol expressing some cluster or

generalization. Briefly, context-free grammars are a class of phrase-structure grammar.

4) **Types 3, Regular Grammars,** are equivalent to regular expressions. A given regular language can be characterized either by a regular expression or by a regular grammar. Regular grammars can either be right-linear or left-linear. A rule in a right-linear grammar has a single non-terminal on the left, and at most one non-terminal on the right-hand side. If there is a non-terminal on the right-hand side, it must be the last symbol in the string. All regular languages have both a left-linear and a right-linear grammar. Briefly speaking, regular grammar, a grammar in which each production has one of the forms: A→b and A→bC, where b is a terminal and A, C are nonterminals. Like the right-linear and left-linear grammars, regular grammars generate precisely the regular languages.

3. HIDDEN MARKOV MODELS

Due to the lack of sample data, only the model architecture is described. It is expected that the experimental environment uses the Anaconda of Python 3.6, mainly using the well-developed machine learning framework sklearn module. The machine learning model set in this paper will be described in detail below, which can be divided into three steps: data collection, feature engineering, model building and training.

Hidden Markov Model (HMM) is a component of a modern speech recognizer. Markov Model has to be introduced before the introduction of Hidden Markov Model. Markov Model is specified by the set of states Q, the set of transition probabilities A, a defined start state and end state(s), and a set of observation likelihoods B. For weighted automata, we defined the probabilities bi(ot) as 1.0 if the state i matched the observation ot and 0. However, Hidden Markov Model has additional two requirements more than Markov Model. First, it has a separate set of observation symbols O, which is not drawn from the same alphabet as the state set Q. Second, the observation likelihood function B is not limited to the values 1.0 and 0; in an HMM the probability bi(ot) can take on any value from 0 to 1.0. By the way, these are the parameters to define the Hidden Markov Model:

- **States**: a set of states Q = q1q2…qN
- **Transition probabilities**: a set of probabilities A = a01a02…anl…ann. Each aij represents the probability of transitioning from state i to state j. The set of these is the transition probability matrix.
- **Observation likelihoods**: a set of observation likelihoods B = bi(ot), each expressing the probability of an observation ot being generated from a state i.

In a Hidden Markov Model, it uses two " special" states (non-emitting states) as the start and end state. However, it's also possible to avoid the use of these states by specifying two more things:

- **Initial distribution**: an initial probability distribution over states, π, such that πi is the probability that the HMM will start in state i. Of course some states j may haveπj = 0, meaning that they cannot be initial states.
- **Accepting states**: a set of legal accepting states.

4. CONCLUSION

Natural Language Processingis one of popular fields of Artificial Intelligence with a splendid future, and it playsa critical role in Artificial Intelligence. Although

it has been highly developed around the world, there are still a lot of fields that need probing and exploring.

5. REFERENCE

[1] Daniel Jurafsky and James Martin, Speech and Language Processing, Prentice Hall, 2000.

[2] Xuedong Huang, Alex Acero and Hsiao-Wuen Hon, Spoken Language Processing, Prentice Hall, 2001.

[3] Christopher Manning and Hinrich Schutze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.

[4] http://www.cse.unsw.edu.au/~billw/nlpdict.html

