SCHOOL OF COMPUTER AND COMMUNICATION ENGINEERING

Formal Language and Computational Theory Course Report

TOPIC: A Review of Computational Complexity of Natural Language

NAME: ZITA LIFELO 泽塔
STUDENT ID: B20200691
MAJOR: COMPUTER SCIENCE AND TECHNOLOGY
PROFESSOR: DUAN SHIHONG

Date: 29 November, 2020

Table of Contents

1. Introduction

Natural language processing (NLP) is the study of mathematical and computational modeling of various aspects of language and the development of a wide range of systems. These include spoken language systems that integrate speech and natural language; cooperative interfaces to databases and knowledge bases that model aspects of human-human interaction; multilingual interfaces; machine translation; and message-understanding systems, among others. Research in NLP is highly interdisciplinary, involving concepts in computer science, linguistics, logic, and psychology. NLP has a special role in computer science because many aspects of the field deal with linguistic features of computation and NLP seeks to model language computational (Joshi, 1991).

The complexity of natural language became a specific topic of scientific inquiry and progress when it was addressed from the perspective of its computational processing. The study of the computational complexity of natural language was initiated by Noam Chomsky in the late 1950's and has advanced since then with a growing body of established results. One of the main reasons researchers are interested in knowing the complexity of natural language is because, being able to compute natural language processing would be an important tool to better understand the human brain and to create new artificial intelligence technology. Hence, the motivation for this report.

Examples of this interest include a review on Computational Complexity of Natural Languages: A Reasoned Overview (Branco, 2018). In his review, he noted that many studies appear to be misled by misunderstandings of relevant mathematical notions and proofs, thus inducing misinterpretations of empirically gathered evidence.

This report aims at providing a review on the computational complexity of natural language with emphasis on parsing and the recognition problem. The next section gives a brief overview of parsing and recognition. Then the details of natural language processing will be highlighted when it comes to address its computational complexity, and in section 3, then the criteria to ascertain different levels of computational complexity is presented. The key evidence that supports the discussion around the level of computational complexity of natural language parsing is presented in section 5, and how this evidence has received different interpretations and supported different research is discussed in section 6. Finally, the conclusion is explained in section 7.

1. Brief Introduction on Parsing and Recognition

In the context of formal language theory, a *language* is a set of strings over some alphabet Σ. Some languages are specified by *grammars*, which are themselves finite objects whose semantics is defined by a *grammar framework*. Familiar grammar frameworks are: context-sensitive grammars, definite clause grammars, tree adjoining grammars, context-free grammars, and nondeterministic finite state automata. Within a given grammar framework *F*, any grammar *G recognizes* a unique

language *L(G)*, namely, the set of strings *accepted* by *G*. Thus, the apparatus of the multi-tape Turing machine also constitutes a grammar framework in this sense. Each grammar in that framework, that is, each specific Turing machine *M* over signature Σ, recognizes the language *L(M)* comprising the set of strings over Σ accepted by *M*. If *F* is a grammar framework, we understand the *universal recognition problem* for *F* to be the following problem:

given a grammar G in F and a string σ over the alphabet of G, determine whether $\sigma \in L(G)$.

This problem is to be distinguished from the *fixed-language recognition problem* for any *G* in *F*:

given a string σ over the alphabet of G, determine whether $\sigma \in L(G)$.

The complexity of the universal recognition problem for a framework *F* is in general higher than that of the fixed language recognition problem for any grammar in *F* (Pratt-hartmann, 2010). For the framework of Turing machines, the universal recognition problem for Turing machines is undecidable; and it is an immediate consequence of the existence of a universal Turing machine that there exist Turing machines whose fixed-language recognition problem is undecidable. In this report, only the fixed-language recognition problem is considered.

Parsing, is an essential feature of a compiler, which translates from one computer language (the source) to another (the target). Typically, the source is a high-level language, while the target is machine language. Parsing is also a feature of natural language recognition systems. More precisely, if the grammar *G* is in Chomsky normal form, we can parse an arbitrary string $w \in L(G)$ of length *n* in $O(n^3)$ time. While a running time of $O(n^3)$ is often considered tractable in computer science, as programs get bigger and bigger, it becomes more and more essential that parsing be performed in linear time (Shallit, 2008).

2. Processing Problems

Human language is an entity of the natural world and to know within which boundaries its computational complexity lies it is necessary to understand how its processing takes place. (Branco, 2018), stated that there are various empirical evidence upon which to draw hypotheses about the processing of natural language. Which ranges from latency times obtained in experimental settings from a population of subjects to individual linguistic judgments, and includes quantitative data collected from corpora or images and recordings of neurological activity in the brain, amongst others. He noted that the processing of natural language is unlikely to constitute a single monolithic procedure. For instance, taking into account perception which permits the mapping of a linguistic form into the linguistic meaning it conveys, several procedures are likely to be involved and interacting among each other (e.g. the detection of the different phonemes, their grouping into individual lexemes, the grouping of lexemes into phrases, the compositional calculation of their meaning from the meaning of their parts,) All such different dimensions and sub-problems of language processing do not have necessarily to be addressed by a single

computational method or procedure, or by different solutions of the same level of computational complexity.

One of the many sub processes that people do in natural language processing can be described as the recognition problem. Here, a computer is given a sentence and asked if the sentence is or is not in a specific natural language. We want to know what the complexity of this problem is. A more formal way of stating this is below, where *s* is a sentence and *L* is a language:

"Given a string s of lexical forms of a natural language L, how complex is the procedure to determine whether or not s is a sentence of L?"

Addressing the computational complexity of natural language from this perspective has the advantage that the empirical evidence needed for its investigation is quite clear and framework-independent, as it requires taking into account just strings of lexemes forming sentences. Recognizing a string of lexical forms as a sentence is actually a simple sub-procedure. It is also worth noting that the overall level of complexity of human language processing is not lower than the level of complexity of its more complex sub-procedures.

3. Levels of Complexity

The recognition problem is rendered as a set membership problem. When the empirical evidence to be taken into account is confined to strings of lexemes, a language *L* lends itself to be regarded as the set *SL* whose elements are precisely those strings of lexemes that are its sentences. Seeking a computational solution for the problem whether a string of lexemes *s* is recognized as being a sentence of language L is thus seeking a solution for the decision whether the string *s* is a member of the set *SL*.

To try to find the complexity of the problem, one must look for some type of equation, algorithm, or machine that would solve the recognition problem and then find the complexity of that tool. A machine that has a language input and breaks apart and analyzes its input contents is called a parser. A brief introduction of the parser was given in the first section.

One way to find the parser for the recognition problem is to think of *L* (a language) as a set of acceptable sentences. We could create a parser that checks the input sentence *s* against the list of acceptable sentences in *L*. Through checking the list, the parser could determine if *s* is in *L*. However, this is the brute-force method, and there are too many sentences in a natural language to be able to make a complete list and a machine to check against that list. The resulting parser would be impractically slow.

The other way we can create a recognition parser is to determine a language *L* by a set of rules. If the input sentence *s* passes all the rules for *L*, then we can say it is a sentence in *L*. These rules are

called grammars and what we find is that depending on the grammar we use in the parser, the complexity of the parser changes. A formal way to structure this parser is written below.

"On input *s* and *GL*" (grammar of a language)

1. Process over a finite many steps
2. If *s* is in *SL* (set of sentences in *L*) based on *GL*:

   Report YES

   else

   Report NO

Now the answer of the original question, the complexity of the recognition problem, lies in the hands of which grammar we use. Different grammars are used to build different parsers with higher and lower complexities. Noam Chomsky talks about three different categories of language grammars in his works: regular, context-free, and context-sensitive (Noam, 1956). Currently, there are no practical parsers using context-sensitive grammars. However, the best parsers for any regular or context-free grammar are practical solutions for the membership problem, with the best parser for regular grammars being a reasonably very efficient one. In particular, the most efficient parsing algorithm for context-free grammars has polynomial (cubic) complexity, while best parsers for regular grammars have linear complexity with time for obtaining a solution for a problem instance of size n (i.e. sentences with n lexemes) being around a value proportional to n^3 and n, respectively, in the worst case (Pratt-hartmann, 2010).

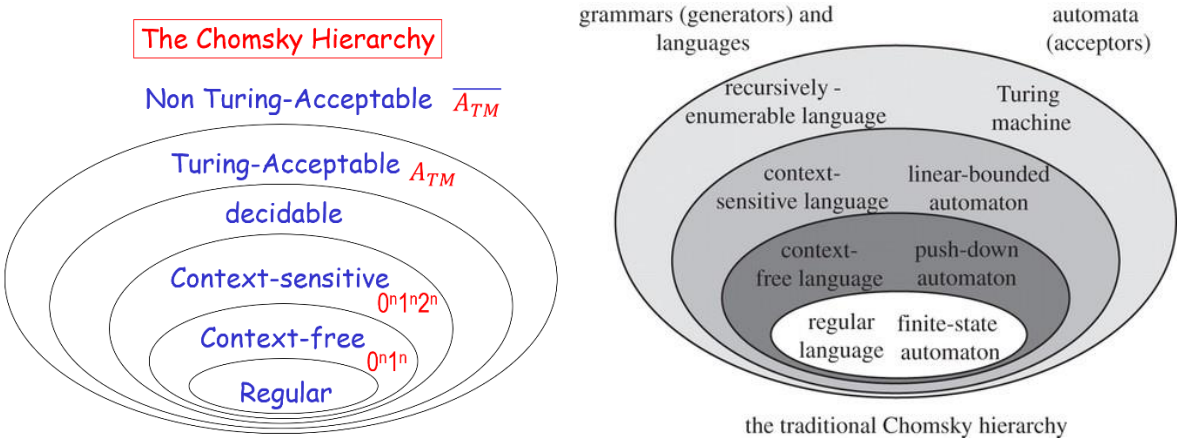

Figure 1: Chomsky Hierarchy Diagram

Figure 1 shows the Chomsky hierarchy for grammars, languages and automata. The Chomsky hierarchy is an inclusion hierarchy that aligns particular forms of grammar, and the languages they generate, with particular classes of automata, abstract computational 'machines' that can be

constructed to accept such languages. All of the grey circles beyond the innermost white circle represent the supra-regular grammars and languages, which require computational power above the level of a finite-state automaton.

This complexity hierarchy has been a yardstick used to help determine the complexity of the solution for the recognition problem in natural language. Assessing the level of complexity for this solution turns out to consist of empirically clarifying what type of grammar is suited to cope with this problem.

4. Types of Grammar

The claim that natural languages are not strictly regular, that is, that they are supra-regular, was put forward in (Noam, 1956). (Branco, 2018), in his review presented the argument as follows:

5.1 Supra-regular

Consider the following sequence of English example sentences built by successively embedding into each other direct object relative clauses modifying subjects:
The cat escaped.
The cat [the dog bit] escaped.
The cat [the dog [the elephant stepped over] bit] escaped.
The cat [the dog [the elephant [the mouse frightened] stepped over] bit] escaped.

Based on these examples, and letting
A = {the dog , the elephant , the mouse , the fly ,...}
B = {bit , stepped over , frightened , chased ,...}

be finite sets of simple noun phrases and transitive verbs, respectively, the following infinite sub set of English can be defined

E' = {*the cat* $a^n b^n$ *escaped* | $n \geq 0$}

where a^n and b^n are any finite sequences of size *n* of concatenated members of A and B. Notice that E' is the intersection of the set E, containing all sentences of English, with the following regular language

R = {*the cat* $a^* b^*$ *escaped*}

where a^* and b^* are finite sequences of any size of concatenated members of A and B, respectively. Given that regular sets are closed under the operation of intersection, that E' results from the intersection between R and E, and that E' is not regular, hence set E, with English sentences, is not regular.

The proof that $a^n b^n$ is not regular resorts to the following Pumping Lemma for Regular Languages:

Let L be a regular language. Then there exists a constant c (which depends on L) such that for every string w in L of length $l \geq c$, we can break w into three subsequences $w = xyz$, such that y is not an empty string, the length of xy is less than $c + 1$, and for all $k \geq c$, the string xy^kz is also in L (J. Hopcroft, 2001). Intuition for the proof: however the members of E' of length longer than c are broken, no subsequences of them can be found that consistently match a pattern xy^kz (Sipser, 2013).The intended proof that E' (and hence E, that is, the English language) is not regular has its grip in case E' is considered to be infinite.

While it is not practically feasible to check this result for every one of the over 7 000 existing languages in the world (M. Paul Lewis, 2015), it is worth noting that this argument has been easily replicated with other types of syntactic constructions besides the center-embedded relative clauses above, and also for natural languages other than English.

(Hauser, 2004), seconded by (Timothy Gentner, 2006), proposed that the divide between regular and supra-regular computational process is the key to tell the difference between non-human and human-like cognitive capacities. This claim was based on arguments of the sort just described.

In the search for the possible place of natural languages in the Chomsky hierarchy of computational complexity, the above argument leads to the next compelling question, whether natural languages are not context-free, that is, whether they are supra-context-free (besides being supra-regular).

5.2 Supra-context-free

Different attempts were made to support the claim that natural languages are supra-context-free. According to (Branco, 2018), one of those that were retained as the best arguments are based on reduplication in noun formation on Swiss German embedded infinitival verb phrases.

The argument based on Swiss German data is as follows:

Consider the following sequence of example sentences built by successively embedding verb phrases in subordinate clauses (-DAT and -ACC signal dative and accusative case, respectively):
Jan s¨ait das mer em Hans es huus haend wele h¨alfe aastriiche.
Jan said that we the Hans-DAT the house-ACC have wanted help paint
Jan said that we have wanted to help Hans paint the house.
Jan sait das mer d'chind em Hans es huus haend wele laa h¨alfe aastriiche.
Jan said that we the children-ACC the Hans-DAT the house-ACC have wanted let help paint
Jan said that we have wanted to let the children help Hans paint the house.

Based on these examples, and letting
A = {*d'chind , ...*}
B = {*em Hans, ...*}
C = {*laa , ...*}
D = {*halfe, ...*}

be finite sets of accusative noun phrases (A), dative noun phrases (B), accusative object taking transitive verbs (C), dative object taking transitive verbs (D), respectively, the following subset of Swiss German can be defined:

G' = {*Jan s¨ait das mer* $a^n b^m$ *es huus haend wele* $c^n d^m$ *aastriiche* | $n, m \geq 0$}

Notice that G' is the intersection of the set G, with all sentences of Swiss German, with the following regular language R

R = {*Jan s¨ait das mer* $a^* b^*$ *es huus haend wele* $c^* d^*$ *aastriiche*}

Given that context-free sets are closed under intersection with regular sets, that G' results from the intersection between R and G, and that G' is not context-free, hence the set G, with Swiss German sentences, is not context-free.

The proof that $a^n b^m c^n d^m$ is not context-free resorts to the following Pumping Lemma for Context-free Languages:

Let *L* be a context-free language. Then there exists a constant *c* (which depends on *L*) such that if *z* is any string in *L* such that its length is at least *c*, then we can write *z=uvwxy*, subject to the following conditions:

i. the length of *vwx* is at most *c*;
ii. *vx* is not an empty string;
iii. for all *i > 0*, $uv^i wx^i y$ is in *L* (Hopcroft et al., 2001, p.275).

Intuition for the proof: however the members of G' of length longer than *c* are broken, no subsequences of them can be found that consistently match the pattern $uv^i wx^i y$. (Sipser, 2013)). The intended proof that G' (and hence G) is not context-free has its full grip in case G' is considered to be an infinite set.

5. Outcomes

For the purpose of gaining insight into the computational complexity of natural language processing, the inquiry reported above focused on the complexity of recognizing a string of lexemes as a sentence. Its outcome turns out to be methodologically productive as it helps to uncover what appear as interesting constraints concerning the nature and processing of natural languages. The way these constraints have been addressed and accounted for has been a key factor on how different types of grammatical research frameworks for natural language have been shaped.

So now, the question is between the two grammars, regular and context-free, which one should we choose to describe the complexity of natural language? Which grammar would create the lowest

complexity recognition parser for natural languages? There are three different parties of thought for solving the problem of choosing the grammar.

The first party of thought is that one should use a regular grammar to create the natural language parser. The lowest complexity regular grammar parser is of *n* complexity (has linear computational power). This bold argument was made first by Noam Chomsky, however many other researchers have found new evidence supporting this claim since then. Various studies have shown that the amount of time it takes humans to process a sentence and the length of that sentence have a linear correlation. Another reason to make this claim is that after embedding more than two phrases within each other, a sentence becomes hard to comprehend from a native speaker's perspective. From a linguistic and natural language processing prospective, this means that those sentences are considered ungrammatical. In addition to this, adding phrases to the right of each other has been shown to be easier to comprehend than adding phrases within each other. This shows that there might be a finite upper bound to sentence comprehension, which would make a linear computational complexity parser (a regular grammar parser) a plausible solution for natural languages.

The second thought is that context-free grammars should be used for natural language parsers, which is a view that is held by (Shieber, 1985) and (Culy, 1985). The lowest complexity parser that uses a context free grammar has a computation complexity of n^3. One reason for this claim, is because, some phrase embedding is so complex that it is not understood by native speakers. If this is the case that processing a natural language is of linear complexity, simple phrase embeddings would be comprehensible. The fact that some phrase embeddings are not easy to decipher shows that they require a higher complexity parser. The obvious more computationally complex parser to choose would then use a context free grammar.

It is important to note that neither of the two arguments are actually contradicting each other. They are both just taking the same set of evidence and interpreting it differently to make an argument for one grammar or the other.

The third party takes into account that sentences tend to be fairly short and so the hypothetical computational complexity in worst case scenario, will not be how this parser is used if tested with real, written, practical sentences. Because of this, (Weinberg, 1982) believe that the complexity of the grammar itself is more important than looking at the grammars based on the complexity of their parsers. Grammars have a memory size, number of rules, and other aspects that can make the individual grammars themselves more or less complex. The more concise a grammar is, the faster the resulting parser will be. So, instead of focusing on parser complexity, we should be focusing on grammars' complexity. This means that a parser can be more complex than a context-free parser, but still be a faster parser, depending on the size and shape of the grammar used. Grammars that go beyond context-freeness might create faster parsers than the other context-free and regular grammars (Weinberg, 1982). These researchers believe that the solution to the problem lies in the

balance between the complexity of the sentence parsing procedure (the parser algorithm) and the size and shape of the grammar used.

Because all three of these views are possible, the answer to the recognition problem is not presently known. This problem is so interesting precisely because it is unsolved. The recognition problem is really important to consider for advancement in natural language processing.

6. Conclusion

The research on natural language grammar described above adopt different ways to accommodate results from research on the computational complexity of the recognition problem. Given the Chomsky complexity hierarchy for computable solutions, they fill the whole spectrum of hypothesis ranging from the position that the grammars of natural languages are regular to the positions that they are context-sensitive, also including the claim that they are basically context-free.

What these outcome bring to light is that, importantly, it is by no means sufficient that a linguistic construction instantiates, a language includes, or an agent handles sequences of items under a pattern $a^n b^n$ or under a pattern $a^n b^m c^n d^m$ to ascertain that these patterns are the result or empirical evidence of at least, respectively, an underlying context-free grammar or an underlying context-sensitive grammar. Likewise, by themselves alone, they are not sufficient to ascertain cognitive skills of higher computational complexity.

To interpret the relevant empirical evidence here, it is not only the shape of the patterns that matter; the possible length of the stretch made of iterated items and the size range of the input also matter. Of course, these observations also hold for artificial languages that happen to be mastered by humans and non-humans alike under experimental settings.

This should not, however, dispute that restricting the focus of inquiry to the recognition procedure has been a productive methodological move, one that has permitted new insights into the computational complexity of natural language. Yet, as noted at the outset, this is certainly just one of the possible sub procedures involved in the wider task of natural language processing, helping to advance research on the lower bound of natural language complexity.

At the moment, since the answer is uncertain and the problem is unsolved, it is more of a thought experiment than a usable piece of knowledge. However, this can be useful in AI and natural language processing once this question is solved. It could definitely help researchers make machines that are able to parse and understand natural language in a similar cognitive way as a human would, which could be applied to multiple different uses. In the future, the third-party path should be explored more. In many ways, if the third party is correct, choosing the specific type of grammar becomes less important and nullifies the claims of the first two parties.

References

Branco, A. (2018). Computational Complexity of Natural Languages: A Reasoned Overview. Lisbon: Proceedings of the Workshop on Linguistic Complexity and Natural Language Processing.

Culy, C. (1985). The complexity of the vocabulary of bambara. *Linguistics and Philosophy*, 8:345–351.

Hauser, W. T. (2004). Computational constraints on syntactic processing in a nonhuman primitives. *Science*, 303:377–380.

J. Hopcroft, R. M. (2001). Introduction to Automata Theory, Languages, and Computation.

Joshi, A. (1991, November 10). *Natural Language Processing*. Retrieved November 10, 2020, from Science, 253(5025), 1242-1249: http://www.jstor.org/stable/2879169

M. Paul Lewis, G. F. (2015). *Ethnologue, Languages of the World. SIL.*

Noam, C. (1956). Three models for the description of language. *IRE Transactions on Information Theory.*

Pratt-hartmann, I. (2010). Computational Complexity in Natural Language. In C. F. Alexander Clark (Ed.), *The Handbook of Computational Linguistics and Natural Language Processing* (pp. 43-73).

Shallit, J. (2008). Parsing and recognition,in A Second Course in Formal Languages and Automata Theory. *Cambridge: Cambridge University Press*(doi: 10.1017/CBO9780511808876.006), 140–173.

Shieber, S. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Sipser, M. (2013). *Introduction to the Theory of Computation. Cengage Learning* (3rd edition. ed.).

Timothy Gentner, K. F. (2006). Recursive syntactic pattern learning by songbirds. *Nature*, 440:1204–1207.

Weinberg, R. B. (1982). Parsing efficiency, computational complexity, and the evaluation of grammatical theories. *Linguistic Inquiry*, 13:165–191.