Formal Language and Computational Theory

Presenter: Zita Lifelo (泽塔)

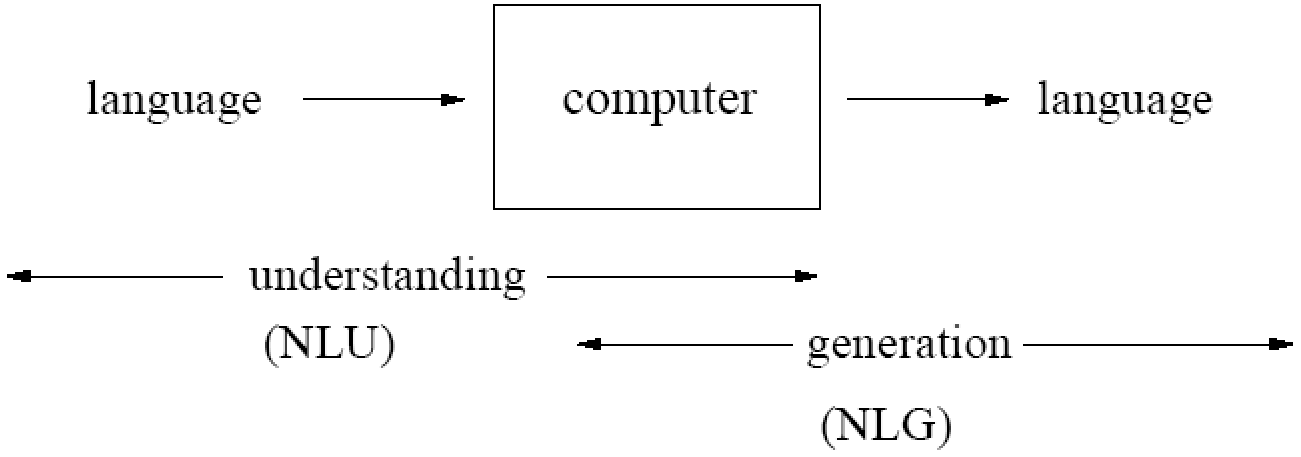
Student ID: B20200691

Report Topic: A Review of Computational Complexity of Natural Language

Content

What is NLP?

- Natural Language Processing (NLP) is a field in Artificial Intelligence (AI) devoted to creating computers that use natural language as input and/or output.



Why NLP?

- To interact with computing devices using human (natural) languages. For example,
  - Building intelligent robots (AI).
  - Enabling voice-controlled operation.
- To access (large amount of) information and knowledge stored in the form of human languages quickly.

Introduction

- 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.

Parsing and Recognition

- *Given a grammar G in F and a string σ over the alphabet of G, determine whether σ ∈ L(G).*
- *Given a string σ over the alphabet of G, determine whether σ ∈ L(G).*

Parsing and Recognition

- 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.

Processing Problems

- One of the many sub processes that people do in natural language processing can be described as the recognition problem.
- *"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?"*

Levels of Complexity

- To find the complexity of the problem, one must look for some type of equation, algorithm, or machine that would solve the recognition problem
- then find the complexity of that tool.

Levels of Complexity

1. 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*.

Levels of Complexity

1. 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*.

- "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
```

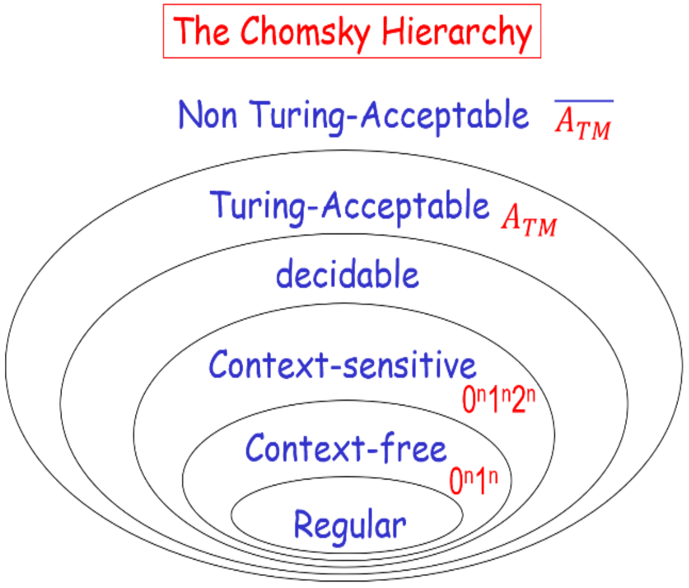
Levels of Complexity

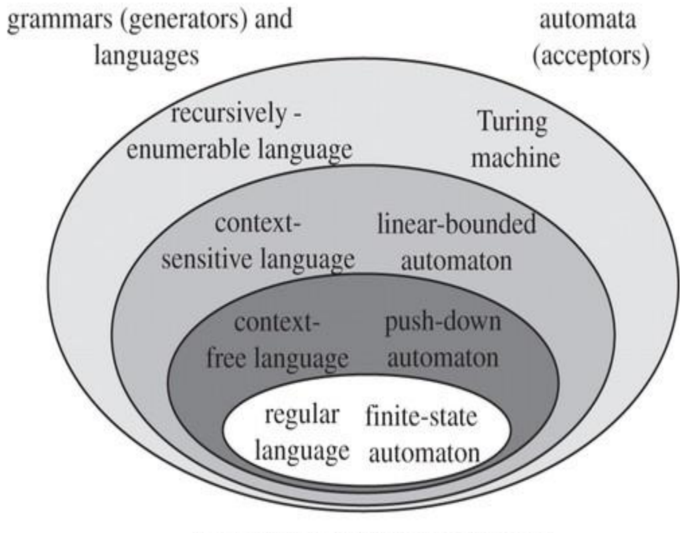
1. The other way we can create a recognition parser is to determine a language *L* by a set of rules. If the input sentence *s* passes all the rules for *L*, then we can say it is a sentence in *L*.

- "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
```

Levels of Complexity





the traditional Chomsky hierarchy

Types of Grammar

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.*

Types of Grammar

1. Supra-regular

- 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

Types of Grammar

1. Supra-regular

- 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.

Types of Grammar

1. Supra-regular

- 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.

Types of Grammar

- 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:

Types of Grammar

- 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*

Types of Grammar

2. Supra-context-free

- *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.*

Types of Grammar

- 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

Types of Grammar

- 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}

Types of Grammar

- 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.

Types of Grammar

- 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:

Types of Grammar

- 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:
  - the length of vwx is at most c;
  - vx is not an empty string;
  - for all $i > 0$, uv^iwx^iy is in L

Outcomes

- 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?

Outcomes

1. 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).

Outcomes

2. The second thought is that context-free grammars should be used for natural language parsers.

- The lowest complexity parser that uses a context free grammar has a computation complexity of n^3

Outcomes

3. The third line of thought 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.

Outcomes

- Researchers believe that the complexity of the grammar itself is more important than looking at the grammars based on the complexity of their parsers.
- 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.

Outcomes

- 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.

- THANK YOU!