Faculty of Computer Science, Dalhousie University *12-Sep-2023*

CSCI 4152/6509 — Natural Language Processing

Lecture 3: Finite Automata Review

Location: Rowe 1011 Instructor: Vlado Keselj
Time: 16:05 – 17:25

Previous Lecture

- Why is NLP hard?
  - ambiguous, vague, universal
- Ambiguities at different levels of NLP
- About course project
  - Deliverables: P0, P1, P, R
  - Project report structure
  - Choosing project topic

Part II

Stream-based Text Processing

Slide notes:

- Considering text as a stream of characters, words, and lines of text
- Review of Finite Automata and Regular Expressions
- Review of Unix-style text processing
- Introduction to Perl
- Morphology fundamentals
- N-grams
- Reading: Chapter 2, Jurafsky and Martin

In this part, we will consider language and text as a stream of characters, words, and lines of text, and look into some processing models that are applicable in this environment. We will first refresh out knowledge about finite automata and regular expressions, and some common Unix-based tools that can be used for basic text processing. We will then introduce the Perl programming language as an extension of these tools. Following this, we will introduce elements of morphology, and introduce character and word n-grams.

4 Review of Automata and Regular Expressions

4.1 Finite-State Automata

Finite-State Automata

- Regular Expressions and Regular Languages
- Regular Languages can be described using
  - Regular Expressions

  - Regular Grammars
  - Finite-State Automata (DFA and NFA)
- DFA = Deterministic Finite Automaton
- NFA = Non-deterministic Finite Automaton
- also referred to as Finite-State Machines

Deterministic Finite Automaton

- Formally defined as a 5-tuple: $(Q, \Sigma, \delta, q_0, F)$
  - Q is a set of states
  - Σ is an input alphabet
  - $\delta : Q \times \Sigma \to Q$ is a transition function
  - $q_0 \in Q$ is the start state
  - $F \subset Q$ is a set of final or accepting states
- Graph representation is frequently used
- Consider finite automata for sets of strings:
  `baaa...a!  ha-ha-...-ha  up-up-down-up-down-up-up-...down`

You should notice that we define a DFA (Deterministic Finite Automaton) in such way that for each state and each character from the input alphabet, there is exactly one transition to another state. In some definitions of DFA, such as the one given in the book, this is relaxed by having at most one transition, and if a transition does not exist, it is assumed to lead to a non-listed “dead” state. We are going to explicitly show this “dead” state in our examples.

Representing DFA

- Formally, as sets and functions (mappings)
- As a transition table
- As a graph
- Consider the DFA for the language: `baaa...a!`

Non-deterministic Finite Automaton

- Formally: $(Q, \Sigma, \delta, q_0, F)$
- However, the transition function is different: $\delta : Q \times \Sigma_\varepsilon \to P(Q)$
  where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$, and $P(Q)$ is the set of all subsets of Q (powerset)
- A string is accepted if there is at least one path leading to an accepting state
- Consider: `/.*ing/` or `/jan|jun|jul/`

Another NFA and DFA Example

- Write a DFA that accepts any sequence over alphabet $\Sigma = \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ that ends with ‘`eses`’, like ‘theses’ or ‘parentheses’.
- Write an NFA that accepts the same language.

You can consider more examples, such as HTML tags and comments.

Implementing NFAs

- DFA — easy to implement, NFA — not straightforward
- Two approaches for NFA: backtracking and translation to DFA
- Using backtracking — usually inefficient solution

- Translating into a DFA
  - Sets of reachable NFA states become states of new DFA

NFA to DFA Translation

- Start with NFA and create new equivalent DFA
- DFA states are sets of NFA states
- If q_0 is the start NFA state, then the start DFA state is **Closure**(q_0)
- **Closure**(A) of a set of NFA states A is a set A with all states reachable via ε-transitions from A
- Fill DFA transition table by keeping track of all states reachable after reading next input character
- Final states in DFA are all sets that contain at least one final state from NFA