The Word-based Regular Expressions

Aleksey Cheusov
<vle@gmx.net>
<Aleksey.Cheusov@IHS.com>
Ex. <cheusov@invention-machine.com>
Ex. <cheusov@scnsoft.com>

Minsk LUG meeting, 29 Dec. 2012
LVEE 2013 Winter, 16 Feb. 2013
Belarus

Plan for presentation

- Mathematics: RL definition and theorem
- Real World: applications of RL and some notes
- Linguistics: POS tagset, Semantic tagset, Text tagging, Tagged text corpus
- Wanted: Tasks
- Mathematics: ExpRL definition, WRL definition
- Linguistics + Real World: WRE syntax and examples
- Бурные и продолжительные аплодисменты (вы же знаете ☺)

Regular language

Definition: given a finite non-empty set of elements Σ (alphabet):

- $\emptyset$ is a regular language.
- For any element $v \in \Sigma$, $\{v\}$ is a regular language.
- If A and B are regular languages, so is $A \cup B$.
- If A and B are regular languages, so is $\{ab | a \in A, b \in B\}$, where ab means string concatenation.
- If A is a regular language, so is $A*$ where $A* = \{a_1 a_2 \ldots a_n | a_i \in A, n >= 0\}$.
- No other languages over Σ are regular.

Example:

- Let $\Sigma = \{a, b, c\}$. Then since aab and cc are members of $\Sigma*$, $\{aab\}$ and $\{cc\}$ are regular languages. So is the union of these two sets $\{aab, cc\}$, and so is the concatenation of the two $\{aabcc\}$. Likewise, $\{aab\}*$, $\{cc\}*$ and $\{aab, cc, aabcc\}*$ are regular languages.

Regular language (interesting fact)

Provable fact:

- Regular languages are closed under intersection and negation operations.

Finite state automaton

Definition: 5-tuple $(\Sigma, S, S_0, F, \delta)$ is a finite state automaton

- Σ is a finite non-empty set of elements (alphabet).
- S is a finite non-empty set of states.
- $S_0 \subseteq S$ is a set of *start* states.
- $F \subseteq S$ is a set of *finite* states.
- $\delta : S \times \Sigma \to 2^S$ is a state transition function.

Theorem: $\forall$ regular language R, $\exists$ FSA f, such that $L(f) = R$; $\forall$ FSA f, $L(f)$ is a regular language (L — language of FSA, that is a set of accepted inputs).

Consequence: **FSA** are widely used in practice for pattern matching with a help of regular languages (and **finite state machines** AKA **FSM** too, including **Mealy machines**, **Moore machines**, and **finite state transducers** AKA **FST**)

Real world. Regular expressions (hello UNIX and FOSS!).

Regular language is a foundation for so called **regular expressions**, **alphabet** Σ is mainly a set of characters (e.g. ASCII, Unicode):

- **POSIX BRE**: **grep** (except back references), **sed**, **vi** and a lot of other traditional UNIX tools
- **POSIX ERE**: **grep -E**, **sed -E** (BSD), **sed -r** (GNU), **awk**, **lex** and a lot of other traditional UNIX tools
- **Perl**, **pcre**, **Ruby**, **Python** (superset of regular language, and therefore extremely inefficient ☹)
- Google re2, Yandex PIRE
- ...

Some extensions in **regular expressions** over **regular languages**:

- submatch operation (depending on implementation may still be FSM but not FSA)
- backreference (incompatible with regular languages and FSMs at all)

Tagsets

Penn part-of-speech tag set:

- **NN** noun, singular or mass (*apple, computer, fruit* etc.)
- **NNS** noun plural (*apples, computers, fruits* etc.)
- **CC** coordinating conjunction (*and, or*)
- **VB** verb, base form (*give, book, destroy* etc.)
- **VBD** verb, past tense form (*gave, booked, destroyed* etc.)
- **VBN** verb, past participle form (*given, booked, destroyed* etc.)
- **VBG** verb, gerund/present participle (*giving, booking, destroying* etc.)
- etc.

Semantic tag set:

- **LinkVerb** a verb that connects the subject to the complement(*seem, feel, look* etc.)
- **AnimateNoun** (*brother, son* etc.)
- etc.

Tagged sentence, POS tagging, semantic tagging

Examples:

- The book is red →
  The_DT book_NN is_VBZ red_JJ →
  The_DT book_NN/Object is_VBZ red_JJ/Color

  Note: Words *book* and *red* are ambiguous.
- My son goes to school →
  My_PRP$ son_NN goes_VBZ to_IN school_NN →
  My_PRP$ son_NN/Person goes_VBZ to_TO school_NN/Establishment

Task

Q: Suppose we have a text tagged by part-of-speech and semantic tags. What's then? Can we use traditional regular expressions for pattern matching (including submatch)? How easily and efficiently?
A: In my opinion "NO". I believe we need more powerful domain specific language for this task (widely used in NLP) .

Мой любимый велосипед. Expanded regular language

Definition: given a non-empty set of elements Σ (alphabet) and a set of one-place predicates $P = \{P_1, P_2, \dots P_k\}$, $P_i : \Sigma \to \{true, false\}$:

- $\emptyset$ is an expanded regular language.
- For any element $v \in \Sigma$, $\{v\}$ is an expanded regular language.
- For any i $\{v | P_i(v) = true\}$ is an expanded regular language.
- Σ is an expanded regular language.
- If A and B are expanded regular languages, so is $A \cup B$.
- If A and B are expanded regular languages, so is $A \setminus B$.
- If A and B are expanded regular languages, so is $\{ab | a \in A, b \in B\}$, where ab means string concatenation.
- If A is a regular language, so is $A*$ where $A* = \emptyset \cup \{a_1 a_2 \dots a_n | a_i \in A, n > 0\}$.
- No other languages over Σ are expanded regular.

Expanded regular language (interesting facts)

Provable fact:

- $\forall$ expanded regular language R we can build a regular language R^* over alphabet Σ^*, such that $\exists f : \Sigma^* \rightarrow 2^{\Sigma}$ and $L(R) = L(R^*)$ (expanding all elements in $L(R^*)$ with a help of f).

Consequences:

- Expanded regular languages are closed under intersection and negation operations.
- We can build expanded regular expression engine based on well-known FSM-based algorithms!!!

The Word-based regular language

Definition: given

- W — set of words (character sequences), e.g. $"the", "apple" "123", ";", "C_2H_5OH", \ldots$
- T_{POS} — finite non-empty set of part-of-speech tag set, e.g.. $\{DT, NN, NNS, VBP, VBZ, \ldots\}$
- T_{sem} — finite non-empty set of semantic tag set, e.g. $\{LinkVerb, Person, Object, TransitiveVerb, \ldots\}$
- $D_{POS} : W \rightarrow 2^{T_{POS}}$ — POS dictionary, e.g. $D_{POS}("the") = \{DT\}$, $D_{POS}("book") = \{NN, VB, VBP\}$, $D_{POS}("and") = \{CC\}$
- $D_{sem} : W \rightarrow 2^{T_{sem}}$ — semantic dictionary, e.g. $D_{sem}("son") = \{Person\}$, $D_{sem}("mouse") = \{Animal, ComputerDevice\}$
- $EREs$ — finite set of POSIX extended regular expressions, e.g. $\{".*ing", ".multi.*al", "[A-Z][a-z]", "[0-9]+" \ldots\}$ etc.

(to be continued)

The Word-based regular language

(continuation) the **word-based regular language** $(W, T_{POS}, T_{sem}, D_{POS}, D_{sem}, EREs)$ is an **expanded regular language** over alphabet $W \times T_{POS} \times 2^{T_{sem}}$ and one-place predicates $P = \{P^{check}_{t_{POS}}, P^{check}_{t_{sem}}, P^{tagging}_{t_{POS}}, P^{tagging}_{t_{sem}}, P^{word}_{re}\}$, where

- $P^{check}_{t_{POS}}(w, \cdot, \cdot) = true$ if $t_{POS} \in D_{POS}(w)$, and *false* otherwise
- $P^{check}_{t_{sem}}(w, \cdot, \cdot) = true$ if $t_{sem} \in D_{sem}(w)$, and *false* otherwise
- $P^{tagging}_{t_{POS}}(\cdot, tag^{POS}, \cdot) = true$ if $t_{POS} = tag^{POS}$, and *false* otherwise
- $P^{tagging}_{t_{sem}}(\cdot, \cdot, tags^{sem}) = true$ if $t_{sem} \in tags^{sem}$, and *false* otherwise
- $P^{word}_{re}(w, \cdot, \cdot) = true$ if POSIX ERE $re \in EREs$ matches w, and *false* otherwise

The Word-based regular expressions (Finally!).

Syntax:

- **"word"** — word itself (P_{re}^{word}), e.g. "the", "2012-12-29" etc.
- **'regexp'** — words matched by specified regexp (P_{re}^{word})
- **Tag** — words tagged as tag_{POS} ($P_{t_{POS}}^{tagging}$), e.g. NN, DT, VB etc.
- **%Tag** — words tagged as tag_{sem} ($P_{t_{sem}}^{tagging}$), e.g. %Person, %Object, %LinkLerb etc.
- **_Tag** — words having as tag_{POS} in POS dictionary ($P_{t_{POS}}^{check}$), e.g. _NN, _DT, _VB etc.
- **@Tag** — words having as tag_{sem} in semantic dictionary ($P_{t_{sem}}^{check}$), e.g. @LinkVerb, @Object etc.
- **.** (dot) — any word with any POS and semantic tags
- **^** — beginning of the sentence
- **$** — end of the sentence

(to be continued)

The Word-based regular expressions (Finally!).

Syntax (continuation):

- **(R)** — grouping like in mathematical expressions
- <**num R** > — submatch and extraction
- **R ?** and **[R]** — optional WRE
- **R *** and **R** + — possibly empty and non-empty repetitions
- **R {n,m}**, **R {n,}** and **R {,m}** — repetitions
- **R – S** — subtraction
- **R & S** and **R / S** — intersection, / is for single word WREs, & is for complex WREs
- **R | S** — union
- **R S** — concatenation
- **!R** — negation ($L(!R)$ is equal to either $\Sigma \setminus L(R)$ or $\Sigma * \setminus L(R)$ depending on a context of use)

(to be continued)

The Word-based regular expressions (Finally!).

Syntax (priorities from highest to lowest, continuation):

- , / and | in single word non-spaced WREs
- Prepositional unary operation **!**
- Postpositional unary operations **{n,m}**, '**?**', '**+**' and '*****'
- **(R)** and <**num R** >
- **R & S**
- **R – S**
- **R S**
- **R | S**

(to be continued)

WRE examples

- How to select noun phrases
  (leftmost-longest match, only POS tags)
  (DT | CD+)? RB * CC|JJ|JJR|JJS * (NN|NNS + | NP +)
  Ex.: This absolutely stupid decision
  Ex.: The best fuel cell
  Ex.: Black and white colors
  Ex.: Vasiliy Pupkin
- NER (Named Entity Recognition) for person names
  D_{sem}("MrDr") = {"Mrs.", "Mr.", "Dr.", "Doctor", "President", "Dear", ...}
  @MrDr <1 '[A-Z][a-z]+'/!@MrDr␣+>
  Ex.: Doctor <1 Zhivago>
  Ex.: Dr. <1 Vasiliy Pupkin>
  Ex.: Mrs. <1 Kate>
  but **not**
  Ex.: Mr. <1 President>

WRE examples

- Question focus (object attributes)
  ^ "What" "is" "the" <1 @Attribute > "of" <2 .* > "?"
  Ex.: What is the <1 color > of <2 your book> ?
- Tiny definitions (for example from Wikipedia)
  m4_define(NounPhrase, "(see previous slide)") ^ (NounPhrase & (. . . . .)) "is" (NounPhrase & (. . . . .))
  Ex.: What is the <1 color > of <2 your book> ?
- Sentiment analysis
  D_{sem}("BadCharact") = {"sucks", "stupid", "crappy", "shitty", . . . }
  D_{sem}("GoodCharact") = {"rocks", "awesome", "excellent", "best", . . . }
  D_{sem}("OurProduct") = {"Linux", "NetBSD", "Ubuntu", "AltLinux", "iPad", "Android", . . . }
  <1 @BadCharact|@GoodCharact> <2 @OurProduct> |
  <2 @OurProduct> ["is" DT?] <1 @BadCharact|@GoodCharact>

WRE examples

- Context-free or Context-sensitive parsing
- "Features" assignment in machine learning techniques
- Prototyping. Imagine a grep/awk/ruby with builtin WRE! (POSIX regexec(3)/regcomp(3) API is good enough)

The Word-based Regular Expressions is really cool DSL for Natural Language Processing!

Objections? Comments? Questions?