令和２年度学士論文

HTML5 字句解析仕様の
自然言語処理による意味解析

東京工業大学 情報理工学院 数理・計算科学系

学籍番号 17B01064

五十嵐彩夏

指導教員 南出靖彦 教授

提出日 1月18日

概要

概要．概要．概要．概要．概要．がいよう

目次

第 1 章

序論

自然言語は、人間が同士が互いにコミュニケーションをとるために発展してきた言語である. そして自然言語をコンピュータにで処理する技術を自然言語処理（Natural Language Processing）と呼んでいる. 今回は自然言語処理の技術を使って HTML5 の字句解析仕様から命令を抽出することを試みた.

図 1 が HTML5 の字句解析仕様の意味解析の概要である.

第 2 章

準備

2.1　品詞タグ

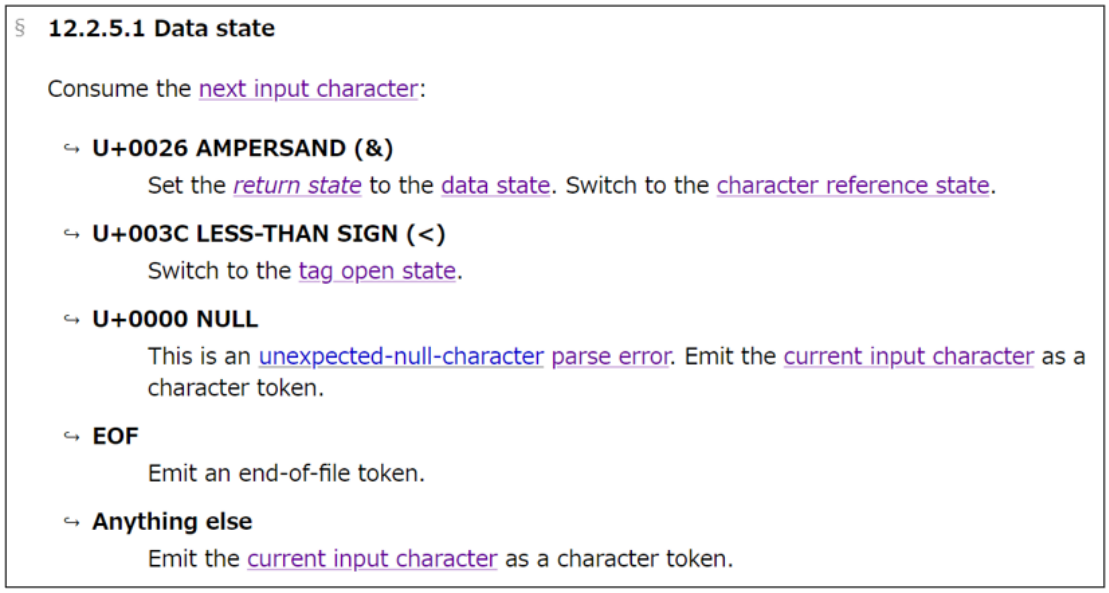
準備 [1] S : 節 VP : 動詞句 NP : 名詞句 VB : 動詞

第 3 章

HTML5 字句解析器

3.1　HTML5 字句解析仕様書

HTML5 字句解析仕様は WHATWG community の web サイトから得られる. [2] HTML5 の字句解析仕様には 80 個の状態がある. aaa 図 3.1



§ **12.2.5.1 Data state**

Consume the next input character:

↪ **U+0026 AMPERSAND (&)**
Set the *return state* to the data state. Switch to the character reference state.

↪ **U+003C LESS-THAN SIGN (<)**
Switch to the tag open state.

↪ **U+0000 NULL**
This is an unexpected-null-character parse error. Emit the current input character as a character token.

↪ **EOF**
Emit an end-of-file token.

↪ **Anything else**
Emit the current input character as a character token.

図 3.1　HTML5 字句解析仕様書

3.2　BNF

cList : CommandList
c : Command
b : Bool
cval : CommandValue
ival : InplementVariable

$$\text{cList} ::= c :: cList \mid Nil$$

c ::= if ⟨Bool⟩ then $\langle \text{cList} \rangle_1$ else $\langle \text{cList} \rangle_2$
- Ignore() //
- Switch(⟨CommandValue⟩)
- Reconsume(⟨CommandValue⟩)
- Set(¡ImplementVariable¿, ⟨CommandValue⟩)
- AppendTo(⟨CommandValue⟩, ¡ImplementVariable¿)
- Emit(⟨CommandValue⟩)
- Create(⟨CommandValue⟩)
- Consume(⟨CommandValue⟩)
- Error(⟨CommandValue⟩)
- FlushCodePoint()
- StartAttribute()
- TreatAsAnythingElse()
- AddTo(⟨CommandValue⟩, ⟨ImplementVariable⟩) //⟨ImplementVariable⟩ = ⟨ImplementVariable⟩ + ⟨CommandValu
- MultiplyBy(¡ImplementVariable¿, ⟨CommandValue⟩)

$$\begin{aligned} <\text{Bool}> ::= & \; And(<Bool>, <Bool>) \\ \mid & \; Or(<Bool>, <Bool>) \\ \mid & \; Not(<Bool>) \\ \mid & \; CharacterReferenceConsumedAsAttributeVal() \\ \mid & \; CurrentEndTagIsAppropriate() \\ \mid & \; IsEqual(<CommandValue>, <CommandValue>) \end{aligned}$$

b ::= And(b_1, b_2)
- Or(b_1, b_2)
- Not(b)
- CharacterReferenceConsumedAsAttributeVal() // CharacterReferenceCode が属性の値として消費されているか
- CurrentEndTagIsAppropriate() //EndTagToken が適切であるか
- IsEqual($cval_1, cval_2$)

$$
\begin{aligned}
< \text{CommandValue} > ::= & \ StateName(< String >) \\
| & \ ReturnState \\
| & \ TemporaryBuffer \\
| & \ CharacterReferenceCode \\
| & \ NewStartTagToken \\
| & \ NewEndTagToken \\
| & \ NewDOCTYPEToken \\
| & \ NewCommentToken \\
| & \ CurrentTagToken \\
| & \ CurrentDOCTYPEToken \\
| & \ CurrentAttribute \\
| & \ CommentToken \\
| & \ EndOfFileToken \\
| & \ CharacterToken(< Char >) \\
| & \ LowerCase(< CommandValue >) \\
| & \ NumericVersion(< CommandValue >) \\
| & \ CurrentInputCharacter \\
| & \ NextInputCharacter \\
| & \ Variable(< String >) \\
| & \ CChar(< Char >) \\
| & \ CString(< String >) \\
| & \ CInt(< Int >) \\
| & \ CBool(< Boolean >)
\end{aligned}
$$

$$
\begin{aligned}
< \text{ImplementVariable} > ::= & \ IReturnState \\
| & \ ITemporaryBuffer \\
| & \ ICharacterReferenceCode \\
| & \ ICurrentTagToken \\
| & \ ICurrentDOCTYPEToken \\
| & \ ICurrentAttribute \\
| & \ ICommentToken \\
| & \ IVariable(< String >) \\
| & \ INameOf(< ImplementVariable >) \\
| & \ IValueOf(< ImplementVariable >) \\
| & \ IFlagOf(< ImplementVariable >) \\
| & \ SystemIdentifierOf(< ImplementVariable >) \\
| & \ PublicIdentifierOf(< ImplementVariable >)
\end{aligned}
$$

第 4 章

自然言語処理

4.1　自然言語処理の対象

[2] HTML5 の字句解析仕様には 80 個の状態がある。今回の字句解析仕様の自然言語処理する対象は 80 個のうち 77 個とした。80 個の状態のうち、自然言語処理の対象とした 77 個の状態は、同じような構造で書かれていた。しかし残りの、Markup declaration open state Named character reference state Numeric character reference end state 3 つの状態はそれぞれ特殊な構造で書かれていたので、これらも一括りにして自然言語処理を適用させるのは、複雑になると判断し、自然言語処理の対象から除外した。

尚、テストする際は残りの 3 つは手動で実装することにした。

4.2　使用ライブラリ

StanfordCoreNLP 形態素解析 (品詞タグ付け、単語の原型の取得), 構文解析, 意味解析などが出来る.
以下の例で詳しく述べる.

4.2.1　自然言語処理の例

”Mika likes her dog’s name.”を StanfordCoreNLP で自然言語処理をさせる.

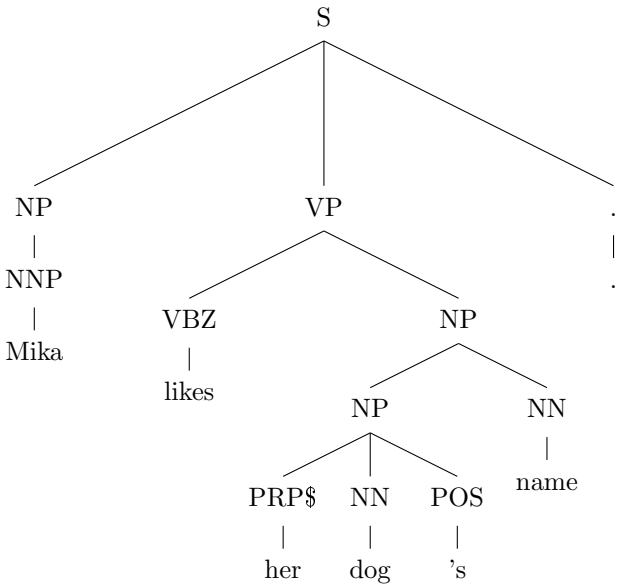
トークン分割、品詞タグ付け、原型

Mika/NNP likes/VBZ her/PRP$ dog/NN ’s/POS name/NN ./.

固有表現抽出

数字や時間、アドレス、人間、地名といった固有な表現を抽出することが出来る.
Mika : PERSON

構文木解析



係り受け解析

係り受け解析とは、単語間の関係を解析するものである.

参照関係の解析

参照関係の解析とは, 文章内で複数個同じものを指し示す単語がある時、それを抽出するものである.it や he などの指示語の指し示すものを見つける時になどに使用される.
Mika, her

4.3 対象の前処理

4.3.1 Scala 構造体

HTML5 字句解析仕様書は構造的に書かれている. name,prev,trans

4.3.2 文字列の置き換え

自然言語処理したい文章をそのまま処理すると, トークンの分割や品詞解析が適切な形で解釈されない.
よって自然言語処理する際に, 前処理として以下の文字列の置き換えをすることによって適切に文章が解釈されるようにした.

状態名の置き換え

attribute value (double-quoted) state =¿ Attribute_value_double_quoted_state 1 つのトークンとして認識させるため、空白、”-”を”_”にする.”(“,”)”を除く. 先頭を大文字にする

Unicode の置き換え

“U+xxxx” =¿ “U_xxxx”(“+”があると 2 つのトークンに分断されてしまうため)

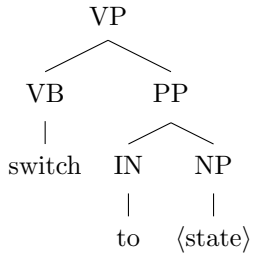
動詞の置き換え

StanfordCoreNLP は命令文の解釈が苦手である. 品詞解析の時点で動詞と認識されるべき単語が名詞扱いされることがあった. 例えば,”Reconsume”は”re”と”consume”の複合語であり, 一般的な辞書にも載ってないので動詞として解釈されないことがあった. よってこのような単語の前に”you”という単語を付け加え,”you Reconsume · · · ”とすることによって,”Reconsume”を動詞として解釈させるようにした. “¡特定の動詞¿” =¿ “you ¡特定の動詞¿” Multiply =¿ multiply

4.4 Lemma tree

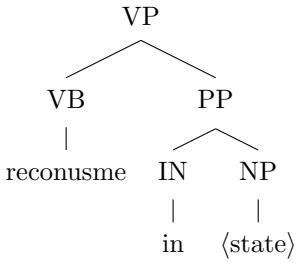
4.5 Tag 型から Command 型への変換

Switch to the · · · state



→Switch(⟨state⟩)

Reconsume in the · · · state



→Reconsume(⟨state⟩)

4.6 If 文の処理

4.7 NP ノードから CommandValue 型への変換

NP ノードを CommandValue 型に変換する際, 単純に文字列に特定の単語が含まれているかどうかを調べるというやり方で実装した.

4.8　参照関係

第 5 章

実装

インタープリタを作成した.

5.1 Env

5.2 Command 型

Command 型のそれぞれの値の解釈

Switch(state: CommandValue)

$\langle$ Switch(state), $env\rangle \rightarrow env[nextState \leftarrow \mathcal{C}[\![state]\!]]$

Recomsume(state: CommandValue)

If(bool: Bool, t: CommandList, f: CommandList)

$$\frac{\langle \text{ clist1}, env\rangle \rightarrow env'}{\langle \text{if } b \text{ then clist1 else clist2}, env\rangle \rightarrow env'} \text{ if } \mathcal{B}[\![b]\!] = true$$

$$\frac{\langle \text{ clist2}, env\rangle \rightarrow env'}{\langle \text{if } b \text{ then clist1 else clist2}, env\rangle \rightarrow env'} \text{ if } \mathcal{B}[\![\mathcal{C}[\![b]\!]]\!] = false$$

5.3 Bool 型

And(a: Bool, b: Bool)

CharacterReferenceConsumedAsAttributeVal()

CurrentEndTagIsAppropriate()

IsEqual(a: CommandValue, b: CommandValue)

5.4 Token 型

tagToken(isStart: Boolean, name: String, attributes: List[Attribute]) DOCTYPEToken(systemIdentifier: String, publicIdentifier: String) characterToken()

5.5 Value

CharVal(c: Char) StringVal(string: String) EOFVal StateVal(statename: String) TokenVal(token: Token)

5.6 CommandValue 型

CommandValue 型から Value 型の値を返す関数
C : CommandValue $\rightarrow$ Value

LowerCaseVersion(cVal: CommandValue)

c.toLowerCase if $C[\![\text{cVal}]\!]$ =c: Char or String

NumericVersion(cVal: CommandValue)

Integer.parseInt(c.toString, 16)

NextInputCharacter

$$\begin{cases} \text{CharVal(c)} & \text{inputText.headOption} = \text{Some(c)} \\ \text{EOFVal} & \text{inputText.headOption} = \text{None} \end{cases}$$

CurrentInputCharacter

currentInputCharacter

EndOfFileToken

TokenVal(endOfFileToken())

第 6 章

評価

6.1　HTML5 テスト

テスト結果

contentModelFlags =¿ 24/24 domjs =¿ 42/58 (((okikae? ,if, CDATA の分岐 (42) entities =¿ 80/80 -¿ 状態 80 のテスト escapeFlag =¿ 9/9 namedEntities =¿ 4210/4210 -¿ 状態 73 のテスト numericEntities =¿ 336/336 -¿ 状態 73 のテスト pendingSpecChanges =¿ 1/1 test1 =¿ 63/68 !!!(((if, test2 =¿ 35/45 !!!(((public,system -¿ 45/45 test3 =¿ 1374/1786 !!!(((pub,sys -¿1786/1786 test4 =¿ 81/85 !!!(((public -¿ 85/85 unicodeChars =¿ 323/323 unicodeCharsProblem =¿ 5/5 xml =¿ 1/4 (((okikae

6.2　上手くいかなかった点

If the six characters starting from the current input character are an ASCII case-insensitive match for the word ”PUBLIC”, then consume those characters この文章を自然言語解析させると”those characters”は”the six characters starting from the current input character”を参照するという出力になる.
もし, この状態へ遷移した時点での入力文字列が”public ···”であったら, まず文字’p’ を消費し, 入力文字列が”ublic ···”となる.
機械的にこの文章を処理しようとすると, 現在の入力文字列”ublic ···”から文字列”public”を消費せよという解釈になるので, 上手くいかない.

第 7 章

結論

命令の種類が限られており, それぞれ決まった書き方をしていることが多かったので, 構文木の情報のみでも命令の抽出がやりやすかった.(機械的な文字のマッチングでも出来そうではあった.)
しかし今回はやらなかったが, 特に命令の記法が一貫していない場合は係り受け解析を用いたほうが様々な形式の文章に対応できるので良いと思われる.

謝辞

本論文を執筆するにあたり，ご指導を賜りました指導教員の南出靖彦先生に，厚く御礼申し上げます．

参考文献

[1] Jeanette Pettibone. Penn treebank ii tags, 2020. https://web.archive.org/web/20130517134339/http://bulba.sdsu.edu/jeanette/thesis/PennTags.html.

[2] WHATWG. Html standard, 2020. https://html.spec.whatwg.org/multipage/parsing.html.