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
17B01064 五十嵐彩夏
指導教員 南出靖彦

1 概要

HTML5 の構文解析を対象とした研究として, XSSAuditor の有効性の検証 [4, 3] や, テストの自動生成 [2] などが行われてきた. それらの研究は実装段階において, 自然言語によって記述されている HTML5 の構文解析仕様から, 手作業でその命令や動作を抽出している. このような研究における手作業による翻訳の負担を減らすため, 本研究では, 自然言語処理を用いて仕様書の命令を自動形式化する.

本研究の自動形式化の対象である, HTML5 字句解析仕様は全て英語によって書かれており, 80 個の字句解析の状態の記述で構成されている.

字句解析の各状態は以下の様に記述されている.

> 13.2.5.9 RCDATA less-than sign state
> Consume the next input character:
> ↪U+002F SOLIDUS (/)
> Set the temporary buffer to the empty string.
> Switch to the RCDATA end tag open state.
> ↪Anything else
> ...

字句解析仕様の記述内容をもとに, 以下のような命令を持つ仕様記述言語を定義する.

$$
\begin{aligned}
\text{command} ::= \ & \text{If(bool, commandList}_1\text{, commandList}_2\text{)} \\
| \ & \text{Switch(cVal)} \\
| \ & \text{Set(iVal, cVal)} \\
| \ & \text{Emit(cVal)} \\
| \ & \cdots
\end{aligned}
$$

例えば, If は条件分岐文, Switch は字句解析の状態の遷移命令, Set は変数 iVal に値 cVal を代入する命令を表している. （bool は条件文を表す値, commandList は命令のリストを表す値, cVal, iVal は字句解析器の変数や値を表す値）

上記の字句解析の状態の U+002F SOLIDUS (/) の部分を仕様記述言語のプログラムへ変換すると, 以下のようになる.

```
Set(ITemporaryBuffer, CString()),
Switch(StateName(RCDATA end tag open state))
```

ITemporaryBuffer 字句解析の一時バッファという変数という意味の値, StateName は状態名を引数に持つ, 字句解析状態名という意味の値である.

本研究では, まず, 自然言語処理のライブラリである Stanford CoreNLP [1] を使い, HTML5 の字句解析仕様に対して自然言語処理を適用する. その際, 自然言語処理そのまま適用するのではなく, 前処理として特定の文字列の置き換えを導入する. そして, 自然言語処理による構文解析や意味解析の結果を使い, 構文解析木に単語の原型や参照関係の情報を付加したデータ型へ変換する. 更にそのデータ型から木構造のマッチングを用いて仕様記述言語の仕様に変換する. 最後に, 字句解析のインタプリタを作成し, それをもとに HTML5 の字句解析のテストを行い, 正しく仕様記述言語へ変換出来たことを確認する.

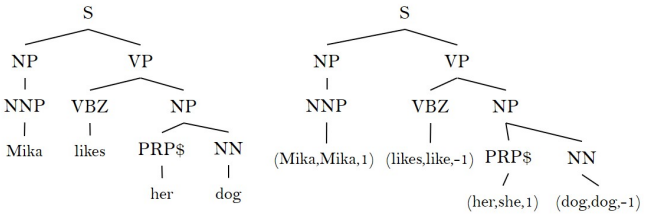
2 仕様への自然言語処理の適用

自然言語処理ライブラリ Stanford CoreNLP を使い, HTML5 字句解析仕様に対して自然言語処理を適用する. そして, 構文解析木, それぞれの単語の原型, 文の語句間の参照関係の情報を取り出す. それらの情報を, Scala のデータ型として定義した, 構文解析木に単語の原型と参照関係の情報を付加した型である, `Tag` 型に変換する.

例えば, “Mika likes her dog” という文章に対して, 自然言語処理を適用すると,

単語の原型がそれぞれ, Mika, like, she, dog であることが分かり, また, Mika と her にラベルが 1 の参照関係が存在することが分かる.

そして構文解析木は以下の左図のようになり, これらの情報から, 右図のような Tag 型の値に変換される.



しかし, 字句解析仕様の文に対してそのまま自然言語処理を適用すると, トークンの分割や品詞解析が適切な形で解釈されない場合がある.

そのため自然言語処理する際に前処理として, 特定の文字列の置き換えをし, 適切に文章が解釈されるようにする.

例えば, 以下のような置き換えを行う.

- 状態名を 1 つのトークンとして認識させるため, 字句解析の状態名を表す語句に対して, 空白 及び “-” を “_” に置き換える. “(” , “)” を除く. 先頭を大文字にする.
- ユニコード “U+xxxx” を 1 つのトークンとして認識させるため, “+” を “_” に置き換える.
- 命令文の解釈が上手くいくように, “Reconsume” など自然言語の解釈が上手くいかない動詞の前に仮の主語を表す “you” を加える.

3 仕様記述言語への変換

自然言語処理で得た Tag 型から, Tag 型のパターンマッチングを用いて, 仕様記述言語への変換を行う.

Tag 型の値が文, 動詞句を表す場合は, 変換元の代表的な文の構文木の形を調べ, それに基づいた パターンマッチを行うプログラムを作成する. それに応じて変換をする.

Tag 型の値が名詞句を表す場合は単純に特定の文字列が含まれているかを判断し, それに応じて変換をする.

条件分岐文の処理

仕様書には, “If $\cdots$,then $\cdots$. Otherwise $\cdots$.” のような条件分岐文も含まれる. その際, 条件文のスコープ（Otherwise で処理する部分）が明らかでない場合がある.

よって, 仕様記述言語への変換の際に, そのようなものがあったら, 変換の途中で人の手でスコープを選ぶようにする.

4 字句解析のテスト

プログラミング言語 Scala で仕様記述言語のインタプリタを作成した. そして, 仕様記述言語のインタプリタと, 自然言語処理によって形式化した字句解析仕様をもとに, HTML5 の字句解析器を実装した.

字句解析仕様から抽出した命令の正しさを検証するため, HTML5 の字句解析のテストデータを使い, 字句解析器のテストを行った.

テストを行った結果, 6,574/7,029 個のテストが成功した. これは, 文字の置き換えの前処理を行ったことや, Tag 型の構文木の変換において, いくつかの文章にアドホックな対応をしたことから, 上手くいったと考えられる.

しかし, 失敗した部分の原因について, 適切に命令を抽出できたと見えても, 実際は正しいものから少しずれていたというケースがあった. それに対してアドホックな形で字句解析インタプリタの処理に対する対処をしたら, 全てのテストが成功した.

参考文献

[1] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[2] Yasuhiko Minamide and Shunsuke Mori. Reachability analysis of the HTML5 parser specification and its application to compatibility testing. *FM 2012: Formal Methods*, 2012.

[3] 小林 孝広. 交代性オートマトンを用いたトランスデューサの包含関係の保守的検査. **東京工業大学　学士論文**, 2019.

[4] 芹田 悠一郎. トランスデューサによる XSS Auditor の有効性の分析. **東京工業大学　学士論文**, 2017.