ブラウザ上でユーザが編集可能な言語パターンマッチシステムの構築
Building a user-editable language pattern matching system in the browser

桂　辰弥[1)]　　竹内　孔一[1)]
Tatsuya Katsura　Koichi Takeuchi

1　はじめに

テキスト中の特定のフレーズや表現を見つけることは，言語および教育分野において必要となることがある．テキストデータから特定のキーワードやフレーズの出現位置や文脈を抽出するためのプログラムとしてコンコーダンサがある．コンコーダンサは語学学習において特定のフレーズや表現の使用例を実際の文脈で把握することで，語彙や文法の理解、単語の使用法や文脈の把握に役立ち，学習者の語彙や表現力の向上に役立つ．パターンマッチングはテキストの表層で検索を行う正規表現とは異なり，情報を抽出したい文を対象に予め関係する文や文の一部に対応する文構造のパターンを用意し，そのパターンに合致する結果を取得するものである．有名なコンコーダンサの例として，Sketch Engine がある．Sketch Engine[1)], はクエリ言語として CQL（Corpus Query Language）[2)]が使用されており，コーパス内で正規表現や演算子を組み合わせることでパターンマッチを行うことができる．しかし Sketch Engine は，コーパスベースの言語分析や統計的な情報抽出が主な役割であるため，テキスト中から依存関係解析を持つ表現を抽出する際には前処理が必要となる．

ユーザ自らがこれらを考慮してテキスト中の特定フレーズや表現を抽出するようなシステムを構築することは容易ではない．そこで本研究では解析モジュールで解析した結果をユーザ自身が求める表現をあらかじめ用意された検索ブロックで組み合わせてシステムに投入し，事例を検索できるシステムの開発を行っている．先行研究　において WEB アプリケーションとして JavaScript と Python を利用した基本システムを構築したが，システムの本格利用にはいくつかの課題が残されている．そこで本報告では検索エンジンの中心部分である Prolog データベースの実装の改良，および、大規模なテキストが扱えるためにデータベースをシステムに導入したので、この改良について報告する．

2　提案する言語パターンマッチシステム

本章では提案する言語パターンマッチシステムを構築している環境についての概要と処理の流れについて述べる．

2.1　言語パターンマッチシステムの概要

本システムは図 1 のようにユーザが視覚的に操作を行うフロントエンドシステムとユーザが要求したテキストの処理バックエンドシステムに切り分けて構成している．フロントエンドシステムは Javascript のライブラリである React.js で構成されており主な機能としてはテキストファイルのアップロード，検索する言語パターンの構築，解析結果の表示，検索結果の表示などがある．

バックエンドシステムは Python の Web フレームワークである Django とデータベースシステムである

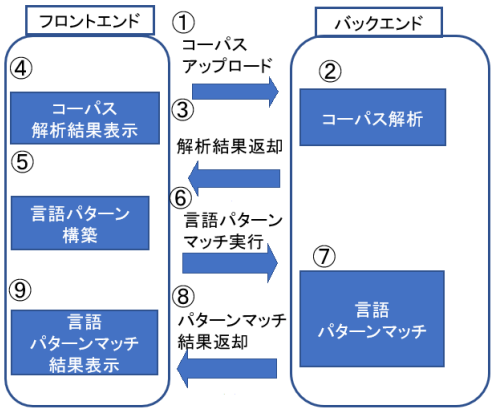

図 1　システムの構成図

Elasticsearch で構成されており，主な機能としてはテキストファイルの解析，ユーザが構築した検索クエリの言語パターンマッチ実行などがある．

詳しい処理の流れについては以降の節で述べる．

2.2　処理の流れについて

バックエンドで行われるテキスト解析，言語パターンマッチ実行の際の処理の流れについてそれぞれ説明する．

テキストの解析の処理はユーザがアップロードしたテキストファイルの文に ASA を用いて形態素解析，係り受け解析，項構造解析を適応させる．解析した結果の文の木構造を図 2 に示す．その後 Prolog 述語に変換を行い，データベースに保存する．変換する Prolog 述語は以下の表 1 ように定義しており．図 3 のように変換される．言語パターンマッチの処理はユーザが構築した検索パターンを送信し，データベースから各文に対応する Prolog を取得し，Prolog パターンマッチを実行する．パターンマッチを実行する Prolog 処理系として SWI-Prolog の python モジュールである pyswip を使用している．

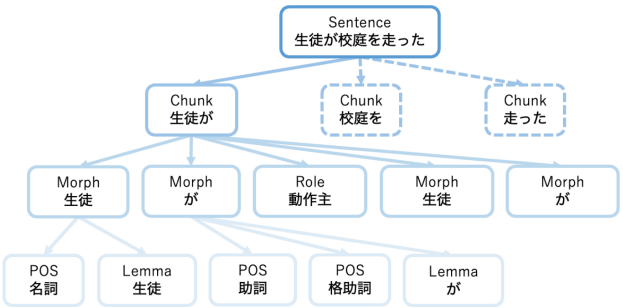

図 2　文を解析した木構造の例

次に

3　評価実験

1）　岡山大学

1）　https://www.sketchengine.eu/
2）　https://www.sketchengine.eu/documentation/corpus-querying/

表 1　Prolog の述語一覧

述語	第 2 引数	第 3 引数
chunk(SENTENCE_ID, 0,_)	根ノード ID	文節 ID
morph(SENTENCE_ID, _, _)	文節 ID	形態素 ID
main(SENTENCE_ID, _, _)	文節 ID	主形態素
part(SENTENCE_ID, _, _)	文節 ID	副形態素
role(SENTENCE_ID, _, _)	文節 ID	意味役割
semantic(SENTENCE_ID, _, _)	文節 ID	概念
surf(SENTENCE_ID, _, _)	ノード ID	ノードの表層
surfBF(SENTENCE_ID, _, _)	形態素 ID	形態素の基本形
sloc(SENTENCE_ID, _, _)	文節／形態素 ID	出現位置
pos(SENTENCE_ID, _, _)	形態素 ID	品詞
dep(SENTENCE_ID, _, _)	文節 ID	文節 ID

```
surf(0,0,生徒が校庭を走った).
chunk(0,0,1).
surf(0,1,生徒が).
sloc(0,1,'0_2').
role(0,1,動作主).
dep(0,1,3).
main(0,1,生徒).
part(0,1,が).
morph(0,1,4).
surf(0,4,生徒).
surfBF(0,4,生徒).
sloc(0,4,'0_1').
pos(0,4,名詞).
pos(0,4,一般).
morph(0,1,5).
surf(0,5,が).
surfBF(0,5,が).
sloc(0,5,'2_2').
pos(0,5,助詞).
pos(0,5,格助詞).
pos(0,5,一般).
chunk(0,0,2).
surf(0,2,校庭を).
sloc(0,2,'3_5').
role(0,2,場所).
dep(0,2,3).
main(0,2,校庭).
part(0,2,を).
morph(0,2,6).
surf(0,6,校庭).
surfBF(0,6,校庭).
sloc(0,6,'3_4').
pos(0,6,名詞).
pos(0,6,一般).
morph(0,2,7).
surf(0,7,を).
surfBF(0,7,を).
sloc(0,7,'5_5').
pos(0,7,助詞).
pos(0,7,格助詞).
pos(0,7,一般).
chunk(0,0,3).
surf(0,3,走った).
sloc(0,3,'6_8').
semantic(0,3,状態変化なし活動).
semantic(0,3,移動動作).
semantic(0,3,移動動作物理).
semantic(0,3,特定の場所での移動動作).
dep(0,3,1).
dep(0,3,2).
main(0,3,走る).
part(0,3,た).
morph(0,3,8).
surf(0,8,走っ).
surfBF(0,8,走る).
sloc(0,8,'6_7').
pos(0,8,動詞).
pos(0,8,自立).
morph(0,3,9).
surf(0,9,た).
surfBF(0,9,た).
sloc(0,9,'8_8').
pos(0,9,助動詞).
```

図 3　システムの構成図

3.1　実験内容

3.2　結果

表 2　パターンマッチの検索時間 (秒)

文の数	検索時間
1	0.110
10	0.531
100	5.54
1000	60
5000	312
10000	620

4　考察と結論

謝辞

謝辞の文章を`\acknowledgment` で指定します。使わなければ謝辞は出力されません。