

# ◆ Pythonを使った機械学習・ディープラーニングの腐食への適用(5)

## 概要

- Pythonを用いたWebスクレイピングで、J-Stage内の腐食に関する論文収集を行った。
- 形態素解析によるAI学習モデルを用いることで、論文が分類できることを確認した。

## 今後の展開

- 腐食研究の情報収集が効率的に行える。

WebページのURLを指定

Requestsを使ってURLを送信

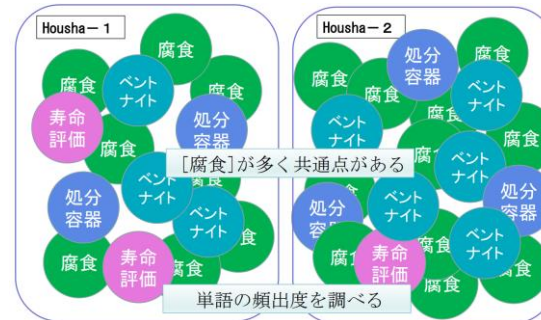
レスポンスからXML全体を取得

パラメータを使って必要な情報を抽出

Webスクレイピングのしくみ

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>材料と環境2024</title>
</head>
<body>
<div id="chapter1">
<h2>Pythonを使った機械学習ディープラーニングの腐食への適用(5)</h2>
<h3>スクレイピングについて</h3>
<ol type="1">
<li>Webページを取得して解析</li>
<li>BeautifulSoupの使い方</li>
</ol>
</div>
<div id="chapter02">
<h3>WebAPIについて</h3>
<ol type="a">
<li>XMLの解析</li>
<li>entryごとにデータ抽出</li>
</ol>
</div>
<div id="chapter03">
<h3>自然言語処理</h3>
<ol type="A">
<li>トピックモデルについて</li>
<li>LDA(Latent Dirichlet Allocation)</li>
</ol>
</div>
</body></html>
```

テスト用WebページのHTMLの内容



AI（人工知能）を使った解析

```
8 import MeCab # 形態素解析エンジン
9
10 tagger = MeCab.Tagger() # 形態素解析するための辞書を指定
11 tagger_text = tagger.parse(text) # 指定した辞書を使用して形態素解析
12 tagger_texts = tagger_text.split('\n')[:-2] # リストの後ろ2つを消す
13 # リスト形式に成形
14 results = []
15 for txt in tagger_texts:
16     parts = txt.split(',')
17     print(parts)
18     surface_pos, pos1, base = parts[0], parts[1], parts[-3]
19     surface, pos = surface_pos.split('\t')
20     results.append(dict(表層形 = surface, 基本形 = base, 品詞 = pos, 品詞1 = pos1))
21 word_freq = defaultdict(int) # 単語の出現頻度を調べる
22 for result in results:
23     if result['品詞'] != '記号': # 記号以外を取得
24         word_freq[result['基本形']] += 1
25 print(dict(word_freq))
```

MeCabを使った形態素解析