自然言語処理班 活動報告書

立命館コンピュータクラブ
2020 年度通年プロジェクト活動

-Leader-
阿部健太朗[*1]

-Members-
B1
山本京介
B2
奥川莞多　小柳雅文　深田紘希
星名藍乃介　八木田裕伍
B3
原 佑馬
B4
玄元奏　山岡聖弥

2021 年 2 月 8 日

[*1] 情報理工学部　二回生

目次

1　はじめに

文責：阿部健太朗

本プロジェクトは，日本語や英語といった自然言語をコンピュータで処理する手法を学ぶプロジェクトである．通年活動において，前半は自然言語処理の基礎技術を，後半はそれらを応用したセンチメント分析，テキスト分類といった応用技術を，ハンズオンな勉強会形式で実施した．

2　活動概要

2.1　自然言語処理とは

文責：山本京介

自然言語処理とは人間が意思疎通のため日常的に使っている言葉（自然言語）を機械的に解析する技術である．プログラミング言語などの人工言語は一定の解釈を持てるよう人為的に作られた，規則的な言語である．対し，自然言語はその名の通り社会的，文化的背景に沿って自然発生したものなので，規則が曖昧で複数の解釈ができてしまう．そこで，あらゆる手法を用いて自然言語に一定の規則性を持たせ，実用的な形にするのが自然言語処理の主な目的である．

ここで自然言語処理のタスク例を下記に示す．

- **形態素解析**
- **構文解析**
- **意味解析**
- **文脈解析**
- テキスト分類
- テキスト生成
- 機会翻訳
- 情報抽出
- 質問応答

中でも，上 4 つ（形態素解析，構文解析，意味解析，文脈解析）は基礎技術と呼ばれ，様々な自然言語処理タスクの要素として利用される．

2.2　前処理

文責：奥川莞多

処理対象のテキストは一般に非構造データである．特に Web テキストには HTML タグや JavaScript のコードといった処理する際にノイズとなる情報が含まれている．このようなノイズは，前処理によって取り除かなければ期待する結果は得られないことがある．

本プロジェクトでは下記に示す 4 つの前処理について学んだ．

- テキストのクリーニング
- N-gram
- 単語の正規化
- ストップワードの除去

2.2.1 テキストのクリーニング

テキストのクリーニングとは，不要な文字を除去する処理のことである．この処理には，Web スクレイピングや正規表現などが用いられる．ただし，ここでは正規表現のみを扱う．例えば，下記のテキストからハッシュタグ（#Python）を除去することを考える．

```
>>> text = "Python でテキストをクリーニングする　#Python"
```

まずは，ハッシュタグを下記のように一般化する．

- 「#」から始まる文字列
- 「#」以降は，アルファベットが 1 文字以上続く

上記で立てた仮説から，ハッシュタグを除去できる正規表現を作成する．下記コードを実行することにより，ハッシュタグの除去に成功した．

```
>>> import re
>>> def clean_hashtag(text):
... cleaned_text = re.sub(r'#[a-zA-Z]+', '', text)
... return cleaned_text
...
>>> clean_hashtag(text)
'Python でテキストをクリーニングする'
```

2.2.2 N-gram

N-gram は，テキストを連続した n 文字で分割する手法である．特に，n が 1 の場合をユニグラム，2 の場合をバイグラム，3 の場合をトライグラムと呼ぶ．

```
>>> def n_gram(target, n):
... return [target[idx:idx + n] for idx in range(len(target) - n + 1)]
...
>>> n_gram("私は昨日の夕食を覚えていない。", 1)
['私', 'は', '昨', '日', 'の', '夕', '食', 'を', '覚', 'え', 'て', 'い', 'な', 'い', '。']
>>> n_gram("私は昨日の夕食を覚えていない。", 2)
['私は', 'は昨', '昨日', '日の', 'の夕', '夕食', '食を', 'を覚', '覚え', 'えて', 'てい', 'いな', 'ない', 'い。']
>>> n_gram("私は昨日の夕食を覚えていない。", 3)
['私は昨', 'は昨日', '昨日の', '日の夕', 'の夕食', '夕食を', '食を覚', 'を覚え', '覚えて', 'えてい', 'ていな', 'いない', 'ない。']
```

2.2.3 単語の正規化

単語の正規化では，単語の文字種の統一，つづりや表記揺れの吸収といった単語を置換する処理を行う．ここでは，下記の 2 つの処理を扱う．

- 文字種の統一
- 数字の置き換え

文字種の統一タスクには，大文字・小文字の統一や半角カタカナを全角カタカナに変換する処理が含まれる．

```
>>> text = "Ritsumeikan␣Computer␣Club␣is␣a␣great␣organization."
>>> text.lower()
'ritsumeikan␣computer␣club␣is␣a␣great␣organization.'

>>> text.upper()
'RITSUMEIKAN␣COMPUTER␣CLUB␣IS␣A␣GREAT␣ORGANIZATION.'
```

自然言語処理において，数値はさほど重要な役割を持たないことが多い．不必要なタスクにおいては，数値をノイズと見做して同一文字に置換する．

```
>>> import re
>>> def normalize_number(text):
... replaced_text = re.sub(r'\d+', '0', text)
... return replaced_text
...
>>> text = '私の誕生日は 2005年 05月 05日です。'
>>> normalize_number(text)
'私の誕生日は 0年 0月 0日です。'
```

2.2.4 ストップワードの除去

ストップワードとは，一般的に多用されるなどの理由で処理対象外とする単語のことである．例えば，助詞や助動詞などの機能語（は，の，です，ます...）が挙げられる．ここでは，日本語ストップワード辞書として有名な Slothlib[1] を用いて除去する．Slothlib の中身を stopwords.txt にコピーしてから下記コードを実行する．

```
>>> text = '結局あそこのラーメンが一番美味しい。'
>>>
>>> stopwords = open('stopwords.txt', 'r').read().split('\n')
>>> stopwords
['あそこ', 'あたり', 'あちら', 'あっち', ... , '同じ', '感じ']
>>>
>>> for sw in stopwords:
... text = text.replace(sw, '')
...
>>> text
'のラーメンが番美味しい。'
```

辞書が不十分であるため精度は低いが，単語の出現頻度などからストップワードを割り出すことで，より高性能なストップワード除去を実現することも可能である．そこで，高頻度の単語をストップワードとみなす例を示す．ここでは出現頻度を分析するコーパスとして ja.text8[4] を使用する．

```
>>> words = open('ja.text8', 'r').read().split()
>>>
>>> from collections import Counter
>>> counter = Counter(words)
>>> counter.most_common(10)
[('の', 828585), ('、', 785716), ('。', 532921), ('に', 527014), ('は', 488009), ('を
    ', 423115), ('た', 421908), ('が', 353221), ('で', 350821), ('て', 259995)]
```

今回は出現頻度が上位 n=10 件のみを表示しているが，n を調整することでより高精度なストップワード除去を実現できる．

2.3 形態素解析

文責：星名藍乃介

2.3.1 概要

自然言語処理の基礎技術のひとつに形態素解析がある．形態素解析とは，テキストを形態素（意味を持つ最小単位）に分割し，各要素に対して品詞を付与する処理のことである．例えば，「庭には二羽にわとりがいる」という文章を形態素に分割すると，「庭，に，は，二，羽，にわとり，が，いる」となる．さらに，「庭：名詞」，「に：助詞」，「は：助詞」といった具合に品詞分類を行う．単語が空白文字で区切られている英語とは違い，日本語で形態素解析を行うには工夫が必要になる．代表的な手法としては，以下の 2 種類の方法がある．

- 文法規則による方法
- 確率的言語モデルによる方法

どちらの手法も品詞辞書や文法辞書と照らし合わせながら解析を行うが，後者の方が精度が高くなりやすいため形態素解析ではよく使われる．

2.3.2 MeCab

形態素解析ツールは多く開発されているが，中でも人気なのが MeCab[2] である．判別精度が高く，実行速度が速いという特徴から，さまざまな場面で利用されている．

```
すももももももものうち
すもも 名詞,普通名詞,*,*,すもも,すもも,自動獲得:テキスト
も 助詞,副助詞,*,*,も,も,*
もも 名詞,普通名詞,*,*,もも,もも,代表表記:桃/もも 漢字読み:訓 カテゴリ:植物;人工物-食べ物 ドメイン:料理・食事
も 助詞,副助詞,*,*,も,も,*
もも 名詞,普通名詞,*,*,もも,もも,代表表記:桃/もも 漢字読み:訓 カテゴリ:植物;人工物-食べ物 ドメイン:料理・食事
の 助詞,接続助詞,*,*,の,の,*
うち 名詞,副詞的名詞,*,*,うち,うち,代表表記:うち/うち
```

2.3.3 Janome

Janome も高い人気を得ている形態素解析器のひとつである．Janome[3] は Python で記述されている形態素解析エンジンであるため，実行時間では MeCab に劣る．しかし，外部エンジンをインストールすることなく，pip install コマンドだけで環境構築できるため，気軽に利用できることが評価されている．さらに，解析に用いる辞書は MeCab と同じであるため，同等の結果が得られると言われている．本班の活動では，環境構築の容易さを優先して Janome を用いた実践を行った．

```
すもも 名詞,一般,*,*,*,*,すもも,スモモ,スモモ
も 助詞,係助詞,*,*,*,*,も,モ,モ
もも 名詞,一般,*,*,*,*,もも,モモ,モモ
も 助詞,係助詞,*,*,*,*,も,モ,モ
もも 名詞,一般,*,*,*,*,もも,モモ,モモ
の 助詞,連体化,*,*,*,*,の,ノ,ノ
うち 名詞,非自立,副詞可能,*,*,*,うち,ウチ,ウチ
```

2.4 センチメント分析

文責：八木田裕伍

センチメント分析とは，テキストの記述内容の感情を分析する手法である．ただし，今回はネガティブ，ポジティブの 1 軸のみの分析を行なった．

センチメント分析は「ポジティブ（ネガティブ）な記述にはそれ特有の単語が含まれている」という考えに基づき，極性辞書と呼ばれる辞書からセンチメントスコアを導出する．今回使用する単語感情極性対応表 [5] には単語に対して-1 から 1 までのスコアが付与されており，テキストに含まれる各単語のスコアを足し合わせたものが，最終的なセンチメントスコアとなる．

2.5 テキスト分類

文責：深田紘希

テキスト分類とは，文書がどんな内容について書かれているかを調べ，それをもとにトピックごとに分類する作業である．

まず文書を形態素解析で分解したものに one-hot エンコーディングを行う．one-hot ベクトルとは 1 つだけ 1 でそれ以外の要素は 0 のベクトルであり，学習器が学習しやすい形のものである．テキスト分類では形態素解析で分解したレベルでエンコーディングするため，one-hot ベクトルは単語の種類分だけ次元が存在するベクトルとなる．

このときに助詞などのストップワードは無視して名詞でエンコーディングするのが一般的である．次に各単語の出現数を集計して文書の特徴量とする BoW 表現を行う．具体的な処理としては，エンコーディングした one-hot ベクトルをすべて足し合わせればよい．BoW 表現をコサイン類似度等の類似度を定量的に表す手法を用いることで，文書の類似度を求めることができる．また SVM を用いてクラスタリングすることでテキスト分類を行うことができる．本プロジェクトでは RCC 総会文書の類似度を求めた．

2.6 テキスト生成

文責：阿部健太朗

テキスト分類の実践として，マルコフ連鎖で自分の過去ツイートの学習を行う．事前準備として，全ツイート履歴をダウンロードする方法 [6] を参考に，自身のツイッターアカウントのアーカイブを用意しておく．

テキスト生成の参考資料 [7] に沿って学習を進めることで，自分のツイートらしいテキストの生成に成功した．例として会公式ツイッター [8] の過去ツイートを学習した生成器の実行結果を下記に示す．

単語を入力:RCC
RCC 会員の先輩方とシューティングゲームを作ってくれました。

単語を入力:C
C 言語を触る会員はもちろん、オブジェクト指向言語を勉強しているモノもあったそうです。

---

3 おわりに

文責：阿部健太朗

本プロジェクトは通年プロジェクトとして活動を行なった．当初の目標であったチャットボット開発まで到達することはできなかったが，自然言語処理の基本を学ことはできたと考えている．

4 参考文献

[1] Slothlib，http://svn.sourceforge.jp/svnroot/slothlib

[2] MeCab，https://taku910.github.io/mecab/

[3] Janome, https://github.com/mocobeta/janome

[4] ja.text8, https://www.kaggle.com/yoshimitsufurusawa/jatext8

[5] 単語感情極性対応表，東京工業大学，http://www.lr.pi.titech.ac.jp/~takamura/pndic_ja.html

[6] 全ツイート履歴をダウンロードする方法，Twitter，https://help.twitter.com/ja/managing-your-account/how-to-download-your-twitter-archive

[7] テキスト生成，averak，https://github.com/averak/rcc-nlp/tree/master/src/07

[8] 立命館コンピュータクラブ，Twitter，https://twitter.com/rits_rcc