自然言語処理

学籍番号:22120
組番号:222
名前: 塚田 勇人

2025 年 6 月 26 日

1　目的

自然言語処理の基礎を学び、実際に *tf*（*Term Frequency*），*df*（*Document Frequency*），*idf*（*Inverse Document Frequency*）、*tf-idf*（*Term Frequency - Inverse Document Frequency*）の計算を行うことで、テキストデータの重要な単語を抽出する方法を理解する。

2　原理

自然言語処理の基本的な概念と、*tf*，*df*，*idf*，*tf-idf* の計算方法について説明する。

2.1　形態素解析

形態素解析は、文章を単語などの要素に分解する処理であり，自然言語処理の前処理として重要なステップである．今回は形態素解析のライブラリとして *MeCab* を使用する．

2.2　*tf*

tf は、特定の単語が文書内でどれだけ頻繁に出現するかを示す指標である。例えばばい、文書 A において単語「自然」が 5 回出現し，文書 B においては 3 回出現した場合、文書 A のほうが「自然」という単語の *tf* が高いとされる。今回は 100 個の文書を用意し、各文書における単語の出現回数を正規化し，*tf* を計算する。このとき各文書の *tf* の総和は 1 になるように正規化する。*tf* の計算式を式 1 に示す。

$$w_{tf_t}^d = \frac{tf(t,d)}{\sum_{s \in d} tf(s,d)} \tag{1}$$

分子の $tf(t,d)$ は単語 t は文書 d 内に出現する回数，分母 $\sum_{s \in d} tf(s,d)$ は文書 d 内に出現する全ての単語の数である．

2.3　*df*

df は、特定の単語がどれだけの文書に出現するかを示す指標である。例えば、100 個の文書のうち、単語「自然」が 10 個の文書に出現した場合、その単語の *df* は 10 となる。

2.4　*idf*

idf は、特定の単語が全体の文書集合においてどれだけ重要かを示す指標である。*idf* は、特定の単語が出現する文書の割合を考慮し、出現頻度が低い単語ほど高い値を持つ。*idf* の計算式を式 2 に示す。

$$idf(t) = \log_{10}\left(\frac{N}{df(t)} + 1\right) \tag{2}$$

$df(t)$ は全文書中で単語 t を含んでいる文書数であり，N は対象となる全文書数である．

2.5 *tf-idf*

tf-idf は、*tf* と *idf* を組み合わせた指標であり、文書内で出現回数が少なくても全文書での出現頻度が低い単語に高い重みが与えられる．一般に，*tf* による重みづけによって文章の網羅性を評価し，*idf* による重みづけによって文書の特定性を評価する．*tf-idf* の計算式を式 3 に示す。この二側面からの重み付けを行うことにより，より統計的に単語の順位付けが可能となる．

$$w^d_{tf\text{-}idf_t} = \frac{tf(t,d)}{\sum_{s\in d} tf(s,d)} \times \left(\log_{10}\frac{N}{df(t)} + 1\right) \tag{3}$$

3 実験環境

実験環境は以下の通りである。

表 1: 実験環境

OS	Windows 11 Pro
CPU	AMD Ryzen 7 5800H
メモリ	16GB
コンパイラ	gcc version 11.4.0
エディタ	Visual Studio Code
形態素解析ライブラリ	MeCab of 0.996

4 プログラムの設計と説明

プログラムは、以下の機能を持つように設計する。

- テキストファイルから文書を読み込み、形態素解析を行う。
- 各文書に対して *tf* を計算する。
- 全文書に対して *df* を計算する。
- 各単語に対して *idf* を計算する。
- 各文書に対して *tf-idf* を計算する。
- 結果を出力する。

5 プログラム

今回のプログラムは表 2 に示すクラスを用いて実装する。

表 2: クラスの概要

クラス名	概要
Word	単語を表すクラス
WordCount	単語の出現回数を管理するクラス
TfCount	各文書における単語の *tf* を管理するクラス
DfCount	全文書における単語の *df* を管理するクラス
tfIdfCount	各文書における単語の *tf-idf* を管理するクラス
NatunalLanguageProcessing	自然言語処理のメインクラス
TF	*tf* を計算するクラス
DF	*df* を計算するクラス
TFIDF	*tf-idf* を計算するクラス

それぞれのクラスについて，説明する．

5.1 *Word* クラス

Word クラスは表 3 のようなメンバ変数を持ち，表 4 のようなメソッドを持つ．

表 3: *Word* クラスのメンバ変数

メンバ変数	概要
`String hyousoukei`	表層形（単語の表記）
`String hinshi`	品詞
`String hinshi1`	品詞細分類 1
`String hinshi2`	品詞細分類 2
`String hinshi3`	品詞細分類 3
`String katsuyoKata`	活用方
`String katsuyoKei`	活用形
`String genkei`	原形
`String yomi`	読み
`String hatsuon`	発音

表 4: *Word* クラスのメソッド

メソッド名	概要
`equals`	オブジェクトの等価性を比較するメソッド
`safeEquals`	オブジェクトの等価性を比較するメソッド（null 安全）
`setter,getter`	各メンバ変数のゲッターとセッター

Word クラスは，形態素解析の結果を表すクラスであり，各単語の表層形，品詞，品詞細分類，活用形，原形，読み，発音などの情報を持つ．

5.2 *WordCount* クラス

WordCount クラスは，表 5 のようなメンバ変数を持ち，各メンバ変数のゲッター，セッターを持つ．

表 5: *WordCount* クラスのメンバ変数

メンバ変数	概要
`private Word word`	単語の出現回数を管理するための単語オブジェクト
`private Integer count`	単語の出現回数

WordCount クラスは，単語の出現回数を管理するクラスであり，各単語の出現回数を保持するための *Word* オブジェクトとその出現回数を表す整数値を持つ．

5.3 *TfCount* クラス

TfCount クラスは，*WordCount* クラスを継承しており，*WordCount* クラスのメンバ変数に加えて，`private Double tf` という *tf* の格納するためのメンバ変数を持ち，メンバ変数のゲッター，セッターを持つ．

5.4 *DfCount* クラス

DfCount クラスは，*WordCount* クラスを継承しており，*WordCount* クラスのメンバ変数に加えて，

6 実行結果

7 考察