Java言語による多言語処理プログラミングについて

永田　清

大東文化大学経営学部

nagata@ic.daito.ac.jp

キーワード　自然言語処理，多言語対応，Java アプリケーション，

1　はじめに

インターネットをそのバックボーンとし，スマートフォンなどの通信機器の普及による情報通信基盤の整備により，2020 年には我が国のオンライン学習環境はハード面とそのインターフェースにおいてかなり整備された状況にあった．また，新型コロナウィルスの蔓延によって対面授業が難しくなったことが，大学等におけるオンライン学習利用を否応なく促す結果となった．

オンライン学習自体の歴史はかなり古く，1974 年設立の現教育情報システム学会 (JSiSE) は CAI(Computer-Aided Instruction) 学会という名称の時もあり，スタンドアローンの PC を念頭に置いていたとしても，コンピュータを使った学習に関する実践とその研究を目的に設立されたものと考えられる．

インターネットの商用利用が解禁となった 1990 年以降は，教材をストリームでオンライン配信できる環境が整い，CBT(Computer-Based Training) や WBT(Webe-Based Training) などが盛んに行われるようになった．しかし，実際の教育現場では，それらはあくまでも補助的学習としての扱いであり，対面式授業による教育効果を高めるための手段として扱われていた．コロナ禍で対面授業ができなくなった 2020 年以降，オンライン学習はなくてはならない学習手段となったが，多くの教員は対面学習が教育の主体である考えているだろう．

2000 年に発表された，大学で経済学を学ぶ学生に対する反転学習 (Inverted Classroom) についての論文 ([4]) では，学習者の学習スタイルも加味した，マルチメディア教育を主体として，教育者がその補完を行う学習方法が有効であることを述べている．我が国においても，前述の JSiSE などで反転学習の実践例が多く発表されている．反転学習においては，特に学習者自身が自分の学習スタイルに合った学習方法で学び，その効果を自己判定して，教育者とのコミュニケーションを通じてその後の学習につなげてい行くことが大切になると考えられる．

我々は，学習者の特性と学習スタイルを考慮した e-Learning 教材作成に関する研究を行い，いくつかの論文を発表している ([3, 9, 10, 11])．これらの研究の発端は，我が国における情報セキュリティ教育が留学生の出身国事情や，留学生に限らず本人の意識特性を考慮したものとなっておらず，それらに対応するためには意識構造の分析と学習スタイルを組み込んたものとすべきである，といった問題意識があった．

アジアを中心としたいくつかの国における情報セキュリティ意識調査を経て，明らかとなった意識構造をもとに e-Learning 教材の作成の試みを行った．また，その段階で多言語化を行うこととしたが，与えられたテーマや学習した項目についてのまとめとしての自由記述やレポートを作成し，その 評価を行うことも重要であると考えた．このようなシステムを使えば e-Learning による自己学習だけでなく，提出されたレポートなどを教員が評価する際にも役立てることができるだろう．

以下は次のような構成になっている．次節で現在まで我々が作成した e-Learning System の概要と，多言語化の方法を簡単に説明する．次に，自然言語処理と既存のプログラムについて述べる．続いて，現在まで状況と今後の方針について示す．

本研究におけるプログラム開発では，統合開発環境 Eclipse を用い，プログラミング言語は Java で，グラフィカルインターフェースの部分にはクラスライブラリとして JavaFx を使っている．

2　情報セキュリティe-Learning System の概要

情報セキュリティe-Learning System における我々の提案には、学習スタイルと情報セキュリティに対する学習者の意識といった 2 つの独立した概念がある．

図 1 は全体像であり，その上部において学習者はアンケートに回答することによって自身の意識構造を分析し，その結果を基にコンテンツリストからコンテンツを選択するプロセスを決定する．下部は，学習者の学習スタイルに関する部分であり，M. J. Lage et al. ([4]) が示した学

習スタイル理論と関係づけられた学習法表を参照することで，適切な学習スタイルを用いて，上部で決定された学習内容が順次表示される．図 1 においては MBTI(Myers-Briggs Type Indicator)([6]) が示されており，その結果を 4MAT([5]) にマップする設計になっている．

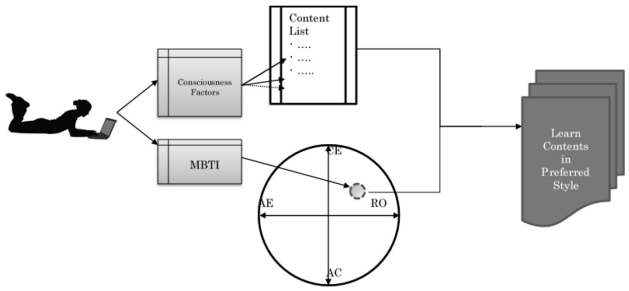

図 1. 情報セキュリティe-Learning システム概要

意識構造に関しては，我々がアジアのいくつかの国において行ったアンケート調査の結果から抽出した意識構造を基に，それらの国々の因子得点平均と学習者の特性を比較してその特徴を求め，コンテンツの選択に反映させることとした．また，学習スタイルの基本となる性格調査には MBTI を用いるようにしているが，MBTI の特許関係でそのままシステムに組み込むことができていない．

もともと在日留学生を念頭において開発を始めたシステムであるので，多言語対応が容易に行えるようにプログラム設計を行った．具体的には，英語によるスタート画面で言語を選択し，その後のページではすべて選択言語による表示を行うようにする．その為に HashMap クラスを使ったクラス作成し，switch 文で選択言語ごとにキーワードに対応する文字列を振り分けることとした．

学習者は選択言語による指示に従って学習を行い，理解度確認のために小テストやレポートを作成するが，ここで問題になるのがレポートの評価である．学習内容に関するキーワードや模範解答を設定して，評価することが考えれるが，それ以前の問題として文章の構成，文法，誤字脱字，句読点混合，文章統一性などといった基本的な評価が望まれる．これらの自然言語処理は難しい問題であるが，以下で一般的な事柄も含め観ていくこととする．

3 自然言語処理

ここで扱う自然言語処理は，基本的にレポートや語句理解などについて，文字として記述された文書を対象とするので，音声認識などの分野に立ち入らないこととする．このような自然言語処理を現実のものとしたツールが，Google に在籍していた Tomas Mikolov により提案された単語分散表現 (Word2Vec) で，2 層のニューラルネットワークのみで構成されるシンプルな構造により，大規模データによる分散表現学習が現実的な計算量で可能となり，自然言語処理が飛躍的に進む契機ともなった [7, 8].

自然言語処理プログラムは Python によるものが多いが，我々の開発プログラムは Java 言語によっており，主に参考文献 [14] のプログラムを借用した．ただし，Word2Vec の取得には deeplearning4j を使うことになっており，この部分も別途作成したが，実際にプロジェクトに組込んだものは鈴木氏らによって作成された「日本語 Wikipedia エンティティベクトル」(http://www.cl.ecei．tohoku.ac.jp/˜m-suzuki/jawiki_vector/) のテキストファイルである．これは，日本語版 Wikipedia の本文全文から学習した単語や Wikipedia で記事となっているエンティティの分散表現ベクトルであり，Wikipedia Entity Vectors (https://github.com/singletongue/WikiEntVec/ releases) からバイナリとテキストの両方をダウンロードすることができる．ベクトルの次元も 100，200，300 から選べるが，当然ファイルサイズは順に大きくなり，処理にも時間がかかる．

これらは日本語の分散表現表であり，多言語対応のプログラムを作成するためには，それぞれの言語の分散表現表を作成するか，作成済みのものを探してダウンロードしておく必要がある．実際，各言語に対応する Wikipedia の DumpData が公開されているが (https://dumps.wikimedia.org/backup-index.html)，例えば英語の場合 2019 年 4 月時点でのファイルサイズが 19TB で，bz2 の圧縮フィルでも 937GB にもなる．また，deepLearning4j を使って分散表現表を作成する場合，各言語ごとに前処理をする必要があり，特に日本語，中国語，韓国語などのように単語がスペースで区切られていない言語では，それぞれの辞書を使って形態素解析を行わなければならない．

因みに，このような多言語対応自然言語処理，評価システムとしては同志社大学の MTMineR(Multilingual Text Miner with R) があり，テキスト型データを構造化して集計し，R を用いて統計的に分析することができる．日本語，中国語，韓国語，英語，ドイツ語とフランス語等のデータを扱うことができ，ホームページ上で無償で公開されている (https://mjin.doshisha.ac.jp/MTMineR/mt.html)．我々のシステムでこれを使うことも考えられるが，ここでは独自のプログラムを開発することとする．

以下では，日本語対応と欧米言語対応の別に，評価指標を含めて述べていくこととする．

3.1 日本語自然言語処理

前掲の文献 [14] における処理プログラムは，日本語を念頭に置いたものである．日本語や英語の文書処理に関しては，大学入試改革に関連して導入が検討された，論述・筆記問題でも話題となったが，AI による評価手法など多くの研究が進行している段階である [13].

文章から特殊記号などを除去した後，第 1 段階として各文章を単語に分解し，品詞などの特徴の付加するといった形態素解析 (Morphological Analysis) を行う．日本語形態素解析ソフトには，ChaSen (https://chasen-legacy.osdn.jp/), JUMAN (https://nlp.ist.i.kyoto-u.ac.jp/?JUMAN)，KAKASI (http:// kakasi.namazu.org/index.html.ja) などがあるが，ここでは MeCab (https://taku910.github.io/mecab/) を使う．

MeCab 本体のダウンロードは，例えば https://github.com/taku910/mecab にある “Code” から “Download ZIP” により Zip ファイルを解凍して，適当なフォルダの置いてパスを通しておく．形態素解析を行うには辞書が必要であり，これは https://taku910.github.io/mecab/から gz ファイルがダウンロードできるので，解凍した csv ファイルなどを MeCab\dic\ipadic フォルダ内におく．コマンドラインでの MeCab による文書解析は，

mecab “入力ファイル名 ” -o “出力ファイル名 ”

であり，ファイル名は拡張子 (.txt) を付け，文字コードは utf-8 とする．

出力ファイルを指定しない場合は，cmd 内に出力されるのでこれを入力ストリームとして受け取り，String クラスの split メソッドを使って分解することになるが，一行の出力順が “表層形”，“品詞”，“品詞細分類 1”，“品詞細分類 2”，“品詞細分類 3”，“活用形”，“原形”，“読み”，“発音” であり，最初の “単語” と “品詞” の間がタブ区切り，その後はカンマ (,) 区切りであるため (図 2)，split(”\t) と split(”, ”) の二段階で行う．

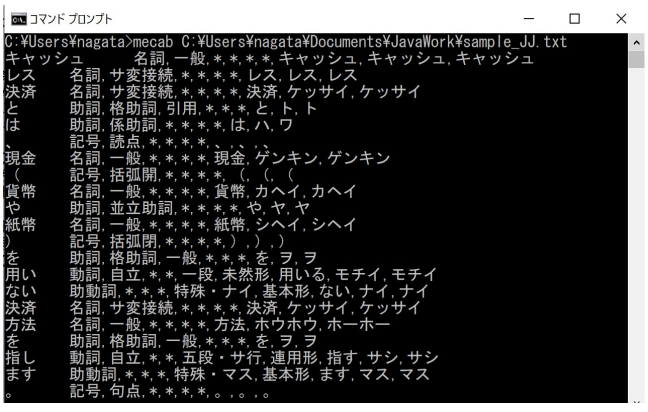

図 2. MeCab 出力例

ここで我々が使うのは，文章に使われている単語である “表層形”，大分類の “品詞”，“品詞細分類 1”，およびカタカナで記された “読み”(または “発音”) くらいであるが，その他いくつかの項目と欧米系単語に係る “音節”，“文字数”，および分散表現としての “単語ベクトル” などをフィールド変数として持つ Word クラスのインスタンスを生成する．

[文字列 “私は学校へ行きます．” の各単語インスタンス]

```
public String text;  // 表層形 (ex:私，は，学校，へ，行き，ます)
public String pos;
// 品詞 (ex: 名詞-代名詞-一般，助詞一係助詞，名詞一一般，助詞-格助詞一一般，動詞一自立，助動詞)
public String lemma; // 基本形 (欧米系単語用)
public String basicForm; // 原形 (ex:私，は，学校，へ，行く，ます)
public String reading; // 読み (ex:ワタシ，ハ，ガッコウ，へ，イキ，マス)
public String pronounce;
public String conjType; // 活用型 (ex:五段カ行促音便，特殊・マス)
public String conjForm; // 活用形 (ex:連用形，基本形)
public int syllablesNo; // 音節数
public int charsNo; // 文字数
public VecWord vecW; // 単語ベクトル
```

形態素解析の後は構文解析として係り受け解析 (Dependency Parsing) を行う．これは一つの文章において，どのまとまりがどこに掛かっているかといった文節間の修飾関係を解析するもので，JUMAN との連携で使われる KNP(https://nlp.ist.i.kyoto-u.ac.jp/index.php?KNP)，Python のライブラリとして提供されている GiNZA を使うなどの手法があるが，ここでは MeCab と連携している CaboCha (https://taku910.github.io/cabocha/) を使う．コマンドラインでの CaBocha による文書解析は，MeCab とほとんど同じで図 3 のようになる．

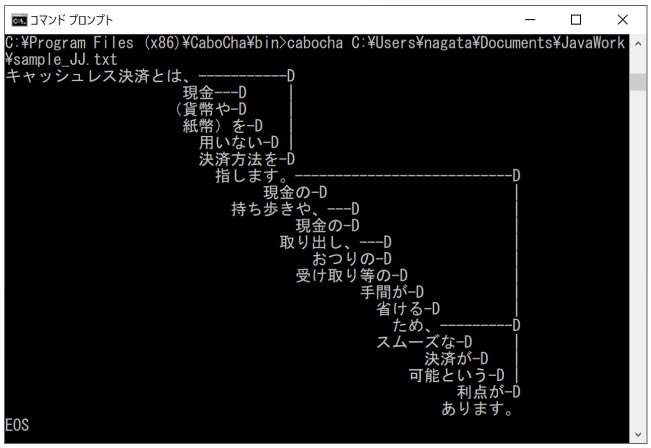

図 3. CaBocha 出力例

この図 3 は，文書構造を図示したものなので，データとして扱ためには “cabocha -f1” としておくと図 4 のように，MeCab の解析結果を含んだ出力が得られる．

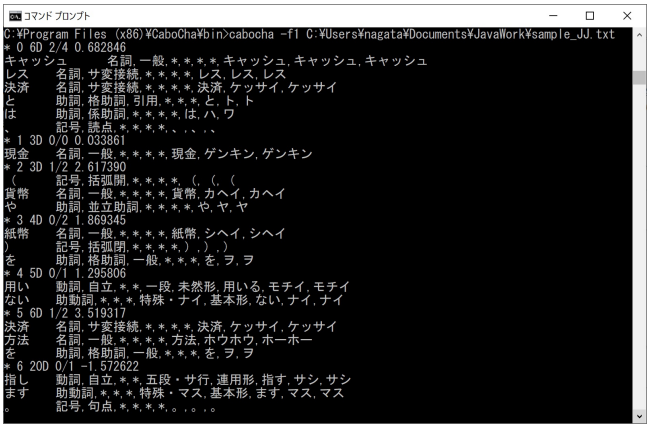

図 4. CaBocha 出力例

記号 “*” 後に 0 から始まる文節連番，“数字+D” がこの文節が掛かっている先の文節番号となっており，それらがスペース (“ ”) で区切られているので，cmd の出力からのストリームを作り，split(” ”) メソッドで分解して，Chunk クラスのインスタンスに，係り受け関係と共に文節に含まれる Word インスタンスを格納する．

単語ベクトルは，Wikipedia Entity Vectores の 200 次元テキストファイル jawiki.word_vectors.200d.txt(約 1.6GB) をダウンロードし，その見出し部分と Word インスタンスの “text” フィールドとを比較して対応するベクトルを取得し Word インスタンスのフィールド変数 vecW 追加した．

語彙レベルの計算には，当面日本語教育語彙表 ver1.0（http:// jhlee.sakura.ne.jp/JEL.html）を利用することとし，ホームページ (http://jhlee.sakura.ne.jp/JEV/) から 17,290 項目の見出しからなる EXCEL ファイルをダウンロードして，0 から 6 段階に分類した表と Word インスタンスの “text” フィールドとを比較して値を取得した．この値は語彙水準平均の計算にだけ用いたので，Word インスタンスのフィールド変数には設定していない．

評価の前にレポートやエッセイのタイトルとキーワードを設定するウィンドウをポップアップで作り，設定があった場合は文章内のキーワード数を求めるボタン項目を作成した．文章内でのキーワード頻度に関しては TF-IDF(Term Frequency-Inverse Document Frequency) を用いることが多いので，その組み込みも検討する．それ以外の評価項目の多くは，山本達の論文 [15] を参照し，以下のようになっている．

1. 句読点の統一，および”です・ます”調，”である”調の統一
2. 文長妥当性 = 文の平均文字数 = 全文字数/文節数
3. 句読点妥当性 (句読点間平均文字数)=全文字数/句読点数, 文節毎の平均
4. 漢字使用率 = 漢字文字数/全文字数
5. 主述妥当性 (係助詞の平均数) = 係助詞の数/文の数 (適正値 1.0)
6. 構文妥当性 (係り受けの平均距離) = 全係り受け数/文節総数（適正値 1.988)
7. 語彙豊かさ = 異なり語数/全語数 (0.0 → 1.0)
8. 語彙水準平均 = (名詞，形容詞，動詞) 語彙水準 (0〜6) 平均
9. 表記ゆれ = 表記の異なり，意味が同じ可能性あるもの
10. 冗長性 (文節毎) = (助詞と助動詞を除いた) 単語の繰り返し数/(助詞と助動詞を除いた) 単語総数
11. 誤字脱字
12. テーマ適合性

ただし，1 の句読点統一等，11 の誤字脱字，12 のテーマ適合性は追加項目であり，11 に関しては誤字脱字表を，12 に関しては分散表現ベクトルの利用を考えているが，未完成部分である．

3.2 欧米系自然言語処理

欧米系の言語では単語がスペースで区切られているので，単語だけを抽出するのならば String クラスの split メソッドだけ使えばよいが，文章評価には単語の品詞が必要になるので，やはり形態素解析ソフトウェアとして Tree-Tagger を使う．これは perl で動いているようなので，まず https://www.activestate.com/products/perl/から OS(ここでは Windows64) に対応したものをダウンロードし，インストールして path を通しておく．次に，ホームページ https://www.cis.lmu.de/ schmid/tools/TreeTagger/ から OS に対応した Zip ファイル (tree-tagger-windows-3.2.3.zip) をダウンロードして展開し，全体を適当なフォルダ内に移す．形態素解析プログラム自体は tree-tagger-windows-3.2.3\TreeTagger\bin 内にあるが，それぞれの言

語に対応したパラメータ (辞書) が必要になるので，先ほどのホームページ上から必要なものをダウンロードし展開して \lib フォルダに移しておく．これらのファイル形式は english-bnc.par.gz のように gz ファイルである．

また，実際に形態素解析を行うときは，\bin 内の tag-english.bat のような言語に対応したバッチファイルを使うことになる．ここには，chunk-english.bat，chunk-french.bat，chunk-german.bat といった，英語，フランス語，ドイツ語にそれぞれ対応した Chunker があり，これらを使うと文節解析が行えるはずであるが，chunk-french.bat はうまく動作しなかった．

Tree-Tagger の出力は単語自身 “word”, 品詞 “pos”, および原型 “lemma” の 3 つで， 品詞は言語ごとにその記述記号が異なっている (図 5)．これらに対応するために，やはり HashMap クラスを使って扱う言語ごとに判定を行えるようにした．

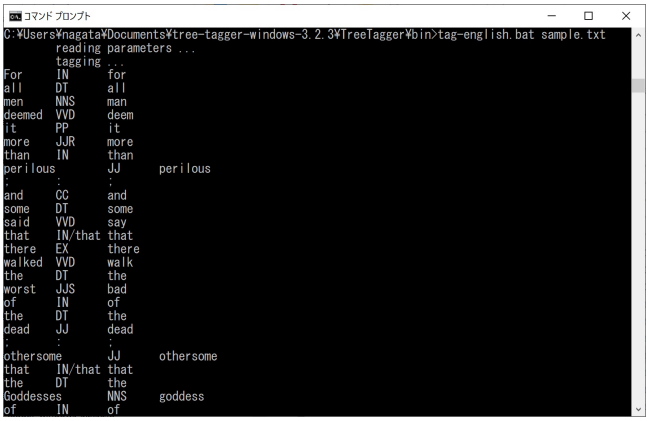

図 5. TreeTagger による形態素解析例

単語ベクトルは，各言語に対応したものが見つからなかったが，幸いにして fastText が非常に多くの言語に対応しているので，それを組み込むことにした．

FastText は，Word2Vec を考案した Tomas Mikolov が Google から Facebook の人工知能研究所 (Facebook AI Research) に移籍して生み出したものである．標準的な CPU を用いた場合でも 10 分以内で 10 億語を学習でき，5 分以内で 50 万もの文を 30 万のカテゴリーに分類できるといわれている．
(https://research.facebook.com/blog/2016/08/fasttext/)

ここでは，多くのコーパスを学習させて独自の辞書を作成するのではなく，https://fasttext.cc/docs/en/crawl-vectors.html 上に，Wikipedia から学習済みのものが多くの言語 (157 言語) ごとに用意されているのでそれを使うことにする．ただし，これらの学習済みファイルデータを使うためには FastText 本体が必要であるが，OS ごとの実行ファイルが配布されるのではなく，GCC++によって各自がコンパイルする必要がある．以下 GCC++，make，git などを含め，fasttext の生成手順を示す．

- GCC++コンパイラーのインストール：
  http://win-builds.org/doku.php/download_and_installation_from_windows から [win-builds-1.5.0] のダウンロードして展開
- make のインストール：
  http://gnuwin32.sourceforge.net/packages/make.htm から “If you download the Setup program of the package, ”
  → http://gnuwin32.sourceforge.net/downlinks/make.php から [make-3.81] の自動ダウンロード
  → path を通す
- git のインストール：
  https://git-scm.com/download/win から 64-bit Git for Windows Setup
  → [Git-2.35.1.2-64-bit] のダウンロード
  → path を通す
- コマンドプロンプト起動：
  > git clone https://github.com/facebookresearch/fastText.git
  > cd fastText
  > make

make や git がインストールされていない場合はこの手順が必要であり，GCC++のコンパイラーはバージョンが違っているとエラーになる．FastText の実行には言語に対応したバッチファイル作成し，予め用意しておいた文書からの単語ファイル (改行区切り) をリダイレクトすることで得られる出力 (図 6) を，Java の入力ストリームに取り込んで Word インスタンスの “vecW” フィールドに設定する．下記は，英単語に対応するバッチファイルであり，“cc.en.300.bin” の部分が対応する言語パラメータとなるので，例えばフランス語処理ならば “cc.fr.300.bin” となる．

[fasttextEE.bat]

```
@echo off
cd C:\Users\nagata\fastText\
fasttext print-word-vectors （下に続く）
        cc.en.300.bin < temp.txt
:end
```

これらは，前述の https://fasttext.cc/docs/en/crawl-vectors.html からダウンロードできるが，それぞれのファイルサ

イズが 7GB 前後となるので容量に注意が必要である．ダウンロードできる bin ファイルは 300 次元のものだけであるが，マニュアルにはこれを以下のコマンドで 100 次元のものに変換できるとの記載があった．

./reduce_ model.py cc.en.300.bin 100

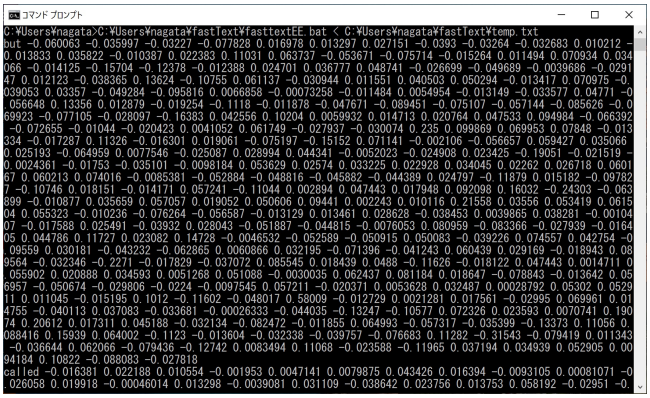

図 6. FastText による (300 次元) 単語ベクトル例

音節 (Syllable) 数は，基本的には単語内の母音の数だが，英語やフランス語では最後の “e” が無音となり，ドイツ語では有音となる．また母音の連続が一つの音節に数えられる場合と，そうでない場合があり，若干の修正が必要である．

語彙水準に関しては，さまざまなホームページ上にデータがあり，水準の設定方法を含め多種多様である．ここでは以下のサイトから単語の頻度順に並べられた “.txt” ファイルを必要な言語ごとにダウンロードして，各単語の頻度によってレベル付けを行ったものを使うこととする．

https://github.com/hermitdave/FrequencyWords/tree/master/content/2018

このサイトからは，62ヶ国で使われている単語が頻度その頻度順に並べられたファイルをダウンロードできる．単語数は言語によって異なっているが，例えば英単語ならば 100 万語以上のリストである．

文章全体の可読性指標 (Readability Index) として [1] による，Flesch Reading Ease Score (p.21)，Dall-Chall Readability Score (p.23)，Fog Grad Level (p.25)，Coleman Index (p.42)，SMOG Grading (p.47)，ARI(Army’s Automated Readability Index) (p.49)，Flesh-Kincaid Readability Grade (p.50) などを組み込む．ただし，いくつかの指標の計算には “読みやすい” 単語数が必要で，英単語に関しては以下のサイトから 3,000 語リストを入手できるが，他の言語では難しい場合もある．また，上記指標の計算式における係数も言語によって若干異なるものもあるので，注意が必要である．

[Dale-Chall Words List]

https://help.readable.com/en/article/dale-chall-words-list-w877fe/

現時点で組み込ん済みの指標は以下の 2 つだけであるが，上記可読性指標は簡単に組み込めるものである．

1. Number of Words(Basic Feature Values)
2. Number of Nouns, Verbs, Adjectives, Adverbs, Conjunctions (Basic Feature Values)

これらの指標は，文章全体の平均値と各文節 (Sentence) 毎の値，および標準偏差を求めて表示する．名詞 (Nouns)，動詞 (Verbs)，形容詞 (Adjectives)，副詞 (Adverbs)，接続詞 (Conjunctions) の個数に関しては，全体平均に円グラフを，文節ごとの比較に積み重ね棒グラフを表示させることとした．

4 現状システム

Start ウィンドウでは，希望する言語を選択する (図 7)．その際，テキストやメディアを読み込むフォルダを指定するようにしている．この画面の表示は英語にしており，選択できる言語は英語，日本語，中国語，韓国語，ドイツ語，フランス語，スペイン語，イタリア語であるが，“情報セキュリティe-Learning” システムでは最初の 4 言語に，テキスト解析では中国語と韓国語を除いた 6 言語に対応する部分だけができている．韓国語の形態素解析は MeCab-ko が使えるようだが，現状ではうまく使えない．

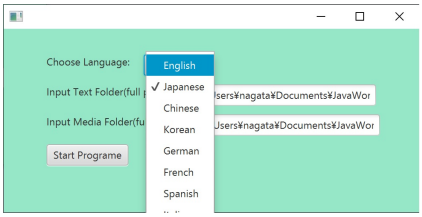

図 7. Start ウィンドウ

図 8 は日本語対応 “情報セキュリティe-Learning” システムの初期画面であるが，ここから左中ほどのボタンをクリックすることによってテキスト解析ウィンドウへ移る．

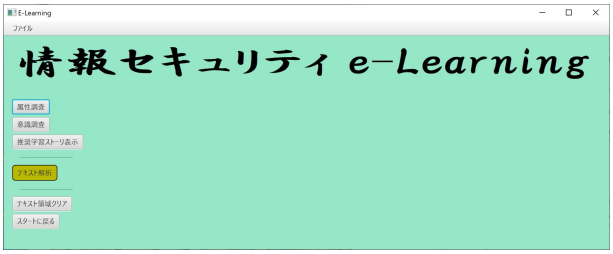

図 8. 日本語版 e-Learning の初期ウィンドウ

テキスト解析ウィンドウは図 9 のように，テキスト領域を持っており，ここにテキストを取り込んで左上の “形態素解析・係り受け解析” ボタンをクリックすると，MeCab と CaboCha による解析が始まり，文節ごとに単語要素を表示する．テキスト領域に入力する文字列は句点 (“．” または “。”) までを一つの文節と考えるようにしたので，改行に意味がない．この時点で，形態素解析結果や単語ベクトルなども計算するので，文の長さにもよるが，処理が終わるまでに時間がかかる．

図 9. 日本語版テキスト解析ウィンドウ

形態素解析結果を保持した状態で，左上から 3 番目のテキスト解析ボタンをクリックすると，図 10 のように評価指標ボタンが表示される．これらの項目は 3.1 で述べたものであるが，上から 2 番目のボタンでテーマやキーワード設定を行っていないと，いくつかの指標が計算できない．また，“誤字脱字”，“テーマ適合性” は単語ベクトルを使って行う予定であるが，現状では最適な手法に至っていない．

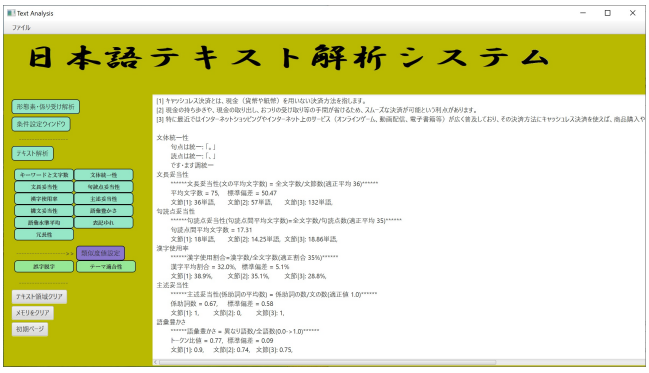

図 10. 日本語版テキスト解析結果

Start ウィンドウで “English” を選択した場合の “情報セキュリティ e-Learning” システムの初期画面が図 11 である．日本語選択時と同様に，“Analysing Text” ボタンでテキスト解析ウィンドウに移行する．

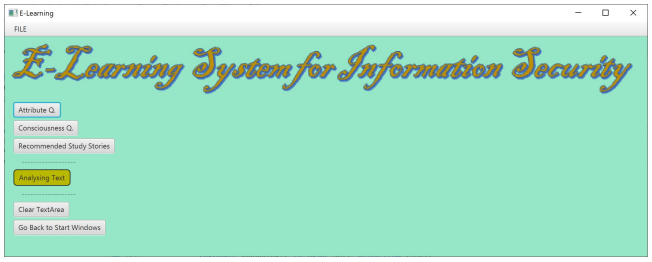

図 11. 英語版 e-Learning の初期ウィンドウ

Start ウィンドウで “English” や “German” を選択した場合のテキスト解析ウィンドウが，それぞれ図 12，図 13 である．テキスト領域には，英語やドイツ語の文を入力して “形態素解析・係り受け解析” に対応するボタンをクリックした時の TreeTagger(Chunker) による解析結果が表示されているが，各品詞を表現する記号が違っていることがわかる．

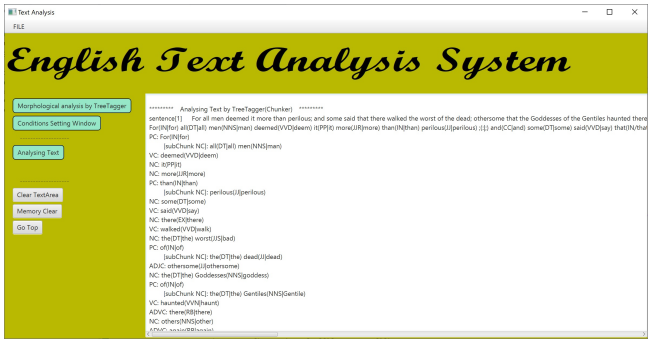

図 12. 英語版テキスト解析ウィンドウ

図 13. ドイツ語版テキスト解析ウィンドウ

図 14，15 は “テキスト解析” に対応するボタンをクリックして表示される “Number of Nouns, Verbs, Adjectives, Adverbs, Conjunctions (Basic Feature Values)” に対応する処理を行った状態である．各品詞の平均値割合が円グラフで，文節ごとの数が積み重ね棒グラフで表示されており，文節ごとのバランスが視覚的にとらえられている．

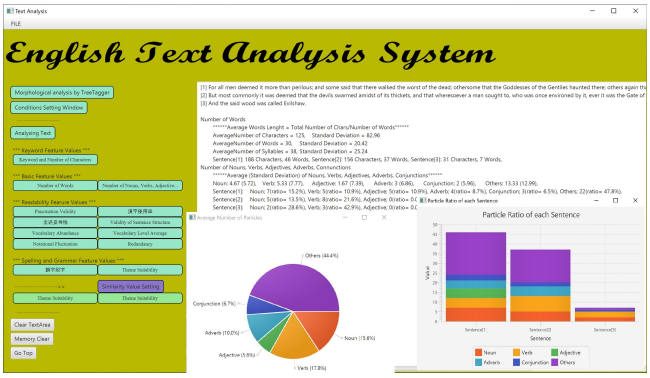

図 14. 英語版テキスト解析結果

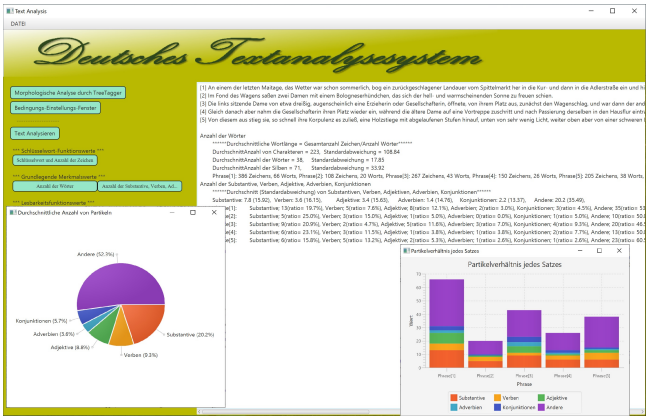

図 15. ドイツ語版テキスト解析結果

5 結果と考察

多言語対応 “情報セキュリティe-Learning” システムの概要説明と，その中に組み込んだテキスト解析システムについての現状報告を行った．自然言語による文書評価には多くの課題があり，メニューとして取り入れた評価指標もまだ開発途上の段階である．

“情報セキュリティe-Learning” システムは，主にアジア各国からの留学生を対象に開発を進めていたが，テキスト解析の段階では，欧米系言語だけとなってしまった．

理由としては，アジアの国々で使われている言語に対し基本的な形態素解析を行うソフトウェアが見つけにくいことがある．また，欧米言語でこれから組み込んでいく予定である文法チェックが難しいといった問題もある．これらを解決するためには，それぞれの国におけるパートナーが必要になってくるかもしれない．

参考文献

[1] DuBay, W. H., *The Principles of Readability*, Impact Information, Cost Mesa California

[2] Ishioka, T. and Kameda, M., Automated Japanese Essay Scoring System based of Articles Written by Experts, *Proceedings of the 21st International Conference on Computational Linguistic and 44th Annual Meeting of the ACL* (2006), pp. 233-240

[3] Kigawa, Y., Nagata, K., and Aoki, T., Multilingual E-Learning System for Information Security Education with Users’ Consciousness, *Proceedings of Advances in Web-Based Learning - ICWL2014*, Lecture Notes in Computer Science 8613, Springer, (2014), pp. 201–206.

[4] Lage, M., Glenn, P., and Michael, T., Inverting the Classroom: A gateway to Creating an Inclusive Learning Environment, *Journal of Economic Education* 31No.1 (2000) pp. 30–41.

[5] McCarthy, B., Using the 4MAT System to bring learning styles to schools, *Educational Leadership*, 48(2) (1990), pp. 31–37.

[6] Myers, I. B., McCaulley, M. H., Quenk,N. L., and Hammer, A. L., *MBTI manual: A guide to the development and use of the Myers-Briggs Type Indicator (3rd ed.)*, Palo Alto, CA: Consulting Psychologists Press, Inc. (1998)

[7] Mikolov, T., Chen, K., Corrado, G., and Dean, J., Efficient Estimation of Word Representations in Vector Space, *ICLR Workshop Paper* (2013)

[8] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J., Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems* (2013), pp. 3111–3119

[9] Nagata, K., Kigawa, Y., Aoki, T., and Nemenzo, F., E-Learning System based on User’s Consciousness and Characteristic *Proceedings of International Conference on Computer Application Technologies* (2015), pp.108–113.

[10] Nagata, K., Kigawa, Y., and Aoki, T., Learning Style and Consciousness Factors in E-Learning System on Information Security *Proceedings of Teaching and Learning in a Digital World*, Intelligent Systems and Computing, Vol.715, Springer, (2017)

[11] Nagata, K., Kigawa, Y., and Aoki, T., Trial for E-Learning System on Information Security Incorporate with Learning Style and Consciousness Factors, *International Journal of Engineering Pedagogy (iJEP)*, Vol. 8, No. 3, Intelligent Systems and Computing, Vol.715, Springer, (2018), pp. 120–136

[12] Suzuki, M, Matsuda, K., Sekine, S., Okazaki, N., and Inui, K., A Joint Neural Model for Fine-Grained Named Entity Classification of Wikipedia Articles, *IEICE Transactions on Information and Systems, Special Section on Semantic Web and Linked Data*, Vol. E101-D, No.1 (2018), pp.73-81

[13] 宇都雅輝, テスト理論と人工知能に基づくパーフォーマンス評価の新技術, 教育システム情報学会誌, Vol. 37, No. 1 (2020), pp. 8-18

[14] 杉本 徹, 岩下 志乃, *Java* で学ぶ自然言語処理と機械学習, Ohmsha, 2018

[15] 山本恵, 梅村信夫, 河野浩之, ルーブリックに基づくレポート自動採点システム, 大学 *ICT* 推進協議会 *2016* 年度年次大会論文集 (2016), https://reg.axies.jp/pdf2016/TP35.pdf, (参照：2022/03/19)