特 別 研 究 報 告 題 目

単語間の距離を利用した
クラスター分析による次元圧縮

Dimension Reduction by Cluster Analysis using Semantic Distance between Words

指 導 教 員　　主査 出口 利憲　教授
副査 山田 博文　准教授

岐 阜 工 業 高 等 専 門 学 校　　専攻科 先端融合開発専攻

2019Y05 遠藤 大介

令 和　3　年（2021　年 ）　　2　月　8　日 提 出

Abstract

In this study, we proposed a method to reduce the dimensions of the similarity calculation between documents by using the distance between words, and considered its effectiveness. In this method, we first calculate the distance between words using Word2vec. Next, we perform hierarchical clustering analysis using Ward's method and classify the words into clusters by the distance between words. Using these clusters, we perform dimension reduction and calculate the similarity between documents.

In this study, the syllabuses of the department of electrical and computer engineering of National Institute of Technology, Gifu College were used. In order to consider the effectiveness, we also performed dimension reduction on LSA and LDA and compared the results. We also compared the results using the concept distance between words calculated from JP WordNet. The reason for this is to compare the methods for getting the meanings of words.

As a result, the proposed method performed better than the conventional method. In addition, Word2vec was better at getting the meaning of words than JP WordNet. We think the reason for these results is that Word2vec learns word meanings from a large number of sentences.

目 次

第1章　序論

近年のコンピュータやスマートフォンの普及は著しく、誰しもが手軽にインターネットを利用することが可能になっている。それに伴い、ソーシャルネットワーキングサービスやクラウドコンピューティングの発展が進んでおり、インターネット上で使用されるデータは増加し続けている。インターネット上のデータには重要な情報から虚偽の情報まで多種多様な情報が含まれている。もちろん、これらの情報は多種多様なだけでなく、量も膨大であり、人間が全てに目を通し、有用な情報を選定するのは非常に困難である。そこで、コンピュータを使用して膨大な情報から有用な情報を得るためのコンピュータ技術が開発された。これをデータマイニングといい、特にテキストデータを対象とした技術をテキストマイニングという。テキストマイニングは、文書を単語ごとに分割し、各単語の出現頻度や相関関係等を分析することで有用な情報を見つけだす。この技術は、SNS への投稿の分析、企業のアンケート分析、新聞からの株式市場の予測などといった活用がされている。社会で実践的な利用をされるほど、テキストマイニングは膨大なテキストデータをデータとして処理することができるが、単語の多義性や書き言葉と話し言葉の違いなど、自然言語から有用な情報を得るには課題も多く現状では完成された技術とは言えない。

本研究では文書間の類似度計算について、各文書に含まれている単語の意味を考慮した手法を提案し、有効性について検討した。提案手法は、Word2vec によって求められる単語間の距離とクラスター分析を利用した方法である。有効性を検討するための実験を行う際には、本校電気情報工学科の科目のシラバスを実験対象とした。提案手法との比較のため、潜在的意味解析 (LSA : Latent Semantic Analysis) や潜在的ディリクレ配分法 (LDA : Latent Dirichlet Allocation) でも類似度計算を行った。また、単語の意味を考慮する手法の比較をするため、日本語 WordNet から求められる単語間概念距離を利用した場合も比較対象とした。

第2章　テキストマイニング

2.1　テキストマイニング

2.1.1　データマイニング[1)]

データマイニングとは、大量のデータに対して統計学やパターン認識などといったデータ解析の手法を使用することで有用な知識を取り出す技術のことである。取り出す知識は、既知でなく、かつ、役立つ可能性があるものとされている。そのため、大量のデータから単純に目的とするデータを検索するような行為はデータマイニングではない。

データマイニングの対象となるデータは一般的に、量、種類共に膨大であり、人力で有用な知識を取り出すのは困難である。よって、コンピュータを活用した高速な処理が求められる。そのような場面において用いられる技術が、データマイニングである。

2.1.2　テキストマイニング

テキストマイニングとは、テキストデータを対象としたデータマイニングのことである。ここでのテキストというのは、人間が普段から使用している言葉を指している。テキストは単語ごとに意味を持っていたり、人の感情表現の役割を担っているため、通常の数値を対象とした分析よりも高難度である。

テキストデータは通常の文章であり、テキストマイニングの最初の処理は文章を単語や文節に区切ることから始まる。その後、出現する単語の頻度や傾向を解析し、有用な情報を取り出す。本研究では、特に文書間の類似度計算について注目し、単語の意味を考慮した計算方法を提案、検討している。

2.1.3　形態素

形態素とは、意味を持つ最小の単位である。それ以上分割できない語を指しており、厳密には単語とは異なる。そのため、他の語と結びついて初めて語になれるものと、単独で語となれるものの2種類に大別される。

2.1.4　形態素解析

形態素解析とは、人間が普段使用している言葉である自然言語を形態素に分割することである。形態素に分割することで、文章に含まれる各形態素の品詞や変化を判別でき、

文書の意味を割り出す作業の一助となる。また、各形態素の出現頻度等を類似度計算に利用できる。形態素解析エンジンは複数公開されており、本研究では MeCab を使用した。

2.1.5 MeCab

MeCab は、オープンソースの形態素解析エンジンである。言語、辞書、コーパスに依存しない汎用的な設計がされており、日本で最もメジャーなツールの 1 つである。また、設定することで様々な辞書を利用することができ、辞書を用意できれば日本語以外も解析可能である。

MeCab によって形態素解析をするときは、オプションを指定することで様々な結果を出力できる。例として以下の文を形態素解析し、形態素、品詞、標準形等を出力した結果を示す。

「すももももももものうち」

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

2.2 自然言語

2.2.1 自然言語と人工言語

自然言語とは、我々人間が日常的に使用している日本語や英語等の意思疎通のための言語のことである。対義語は人工言語であり、これはコンピュータを制御するプログラムを記述するためのプログラミング言語を指している。

自然言語と人工言語の大きな違いは解釈の自由度であり、自然言語の方が自由度が大きい。自由度が大きいということは、文章による表現の柔軟性が高いと言える。しかし、その反面、文章の背景や前後関係等を読み取る能力が解釈する側に必要となり、解釈する側にも柔軟性が求められる。これら 2 言語の自由度の違いは、自然言語が理解の柔軟性が高くとも問題の無い人間同士のコミュニケーションで使用されているのに対し、人

工言語がコンピュータに不変な解釈を求めていることが原因である。

2.2.2　自然言語の曖昧さ

自然言語の曖昧さには、多義性と類犠牲の 2 つの性質がある。

多義性とは、ある単語が複数の意味で用いられる、あるいは解釈される可能性があることを指す。例えば、「上手」と書かれていても、ジョウズ、カミテ、ウワテ等の読み方があり、前後関係を見て判断しなければならない。

類犠牲とは、異なる単語が同じ意味を持つことを指す。例えば、「雷」と「稲妻」は同じ意味を持っている。

自然言語の解釈は我々人間であっても間違うことがあり、適切な解釈をいつでも行うというのは非常に難しい。このような自然言語の曖昧さは、コンピュータによる自然言語処理に残っている課題の要因でもある。

第3章　実験で使用した技術

3.1　単語の重要度と文書の類似度を算出する手法

3.1.1　TfIdf

TfIdf とは、文書中に含まれる単語の重要度を表す手法である。TfIdf は、文書内の単語の出現頻度 Tf(Term Frequency) と単語の情報量 Idf(Inverse Document Frequency) の積によって求められる。

Tf は単語の出現頻度で、文書内の全単語の出現回数のうち、その単語の出現回数が占める割合である。つまり、同じ文書内で単語が多く出現するほど大きくなる。この値は、同じ文書内で頻繁に出現する単語はより重要な単語であろうという考えが基になっている。

Idf は単語の情報量で、演算対象の全文書のうち、ある単語が含まれる文書の割合の逆数である。つまり、単語が他の文書に出現していないほど Idf は大きくなる。この値は、多数の文書に出現する単語は一般的な単語であろうという考えが基になっており、一般語フィルタのような働きをしている。

全部で N 個の各文書を $d_i(i = 1, 2, \ldots, N)$、文書 d_i に出現する M 種類の各単語を $t_{ij}(j = 1, 2, \ldots, M)$ とすると、Tf、Idf、TfIdf は次の式で求められる。

$$Tf_{ij} = \frac{\text{文書 } d_i \text{ における単語 } t_{ij} \text{ の出現回数}}{\text{文書 } d_i \text{ における全単語の出現回数の和}} \tag{3.1}$$

$$Idf_{ij} = \log \frac{\text{全文書数 } N}{\text{単語 } t_{ij} \text{ が出現する文書数}} \tag{3.2}$$

$$TfIdf_{ij} = Tf_{ij} \times Idf_{ij} \tag{3.3}$$

Idf は、1 を足すことで重要度を 0 にしないようにする場合がある。しかし、全シラバスに出現する単語は類似度に影響がないと考えたため、本研究では 1 を足す処理は行わなかった。また、Idf の計算式にある log の底は e とした。

3.1.2 cos 類似度

cos 類似度とは、ベクトル空間モデルにおいて、文書同士を比較する際に用いられる類似度計算手法である。この手法は、そのまま、ベクトル同士の成す角度の近さを表現するため、通常の三角関数のコサイン通り、1 に近ければ類似しており、0 に近ければ類似していないことになる。

ベクトル $\vec{a}$ とベクトル $\vec{b}$ の cos 類似度は、次の式で求められる。

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} \tag{3.4}$$

3.2 次元圧縮の手法

3.2.1 主成分分析 [2)]

実験や調査において、記録している項目数が少なければ、グラフや統計量によってその特性を容易に知ることができる。しかし、実際にはデータは多種多量であることがほとんどで、データ同士の関係が複雑になり、分析が難しくなる。そのような問題を解決する手法として、主成分分析 (PCA : Principal Component Analysis) が存在する。主成分分析は、高次元のデータが持つ情報をできるだけ損なわずに、低次元空間に情報を縮約する手法である。高次元のデータを低次元空間に縮約できれば、データ全体の雰囲気を視覚化することができる。この視覚化により、データが持つ情報を解釈しやすくなる。主成分分析の手順について、以下に式を用いて説明する。

P 個のデータ $x_p (p = 1, 2, \ldots, P)$ について、$N (N \leq P)$ 個の主成分 $z_n (n = 1, 2, \ldots, N)$ とこれらの関係は、次式のような互いに独立な線形結合として表される。

$$z_n = \sum_{p=1}^{P} a_{pn} x_p \tag{3.5}$$

ここで、z_n は第 n 主成分と呼ばれ、その結合係数 a_{pn} は次の式を満たす必要がある。

$$\sum_{p=1}^{P} a_{pn}^2 = 1 \qquad (\forall n) \tag{3.6}$$

主成分ができるだけ多くの情報を持つようにするためには、この結合係数が重要となり、係数を決定する際にはデータの分散に注目する。この例を示すため、Figure 3.1 の

ような 2 次元のデータを考える。この図において、データのばらつきが最大となる方向を軸として表すと、z_1 軸のようになることがわかる。これが第 1 主成分となり、このような軸ができるように式 (3.5) の結合係数を決定する。しかし、これだけでは元のデータが持っている情報を十分に表したとはいえない。そこで、データのばらつきが次に最大となる方向に軸をとり、z_2 軸とする。これが第 2 主成分であり、これによって、第 1 主成分では表せていない元のデータの情報を補う。結果として、Figure 3.1 における X, Y の特性を把握しやすくなった。このような結合係数の決定を繰り返すことで、元のデータの損失を最小に抑えている。

この例では、説明を簡略化するため、2 次元の簡単なデータを取り扱った。しかし、主成分分析の対象とするようなデータは、もっと高次元である場合が大半である。データが高次元であるほど、分散も大きくなり、この手法の利点がより一層表れる。

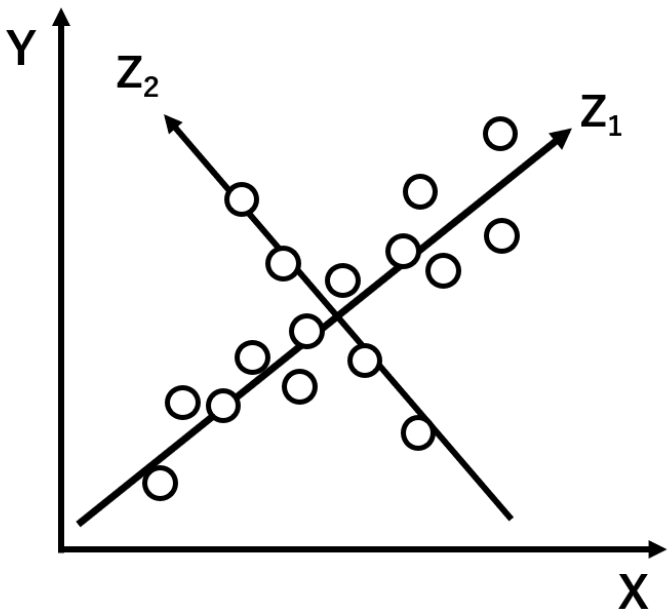


Figure 3.1　Example of two-dimensional data

主成分分析では、分析の対象となるデータ行列の共分散行列の固有値が、主成分の分散と等しくなる。ここで、各固有値は、大きいものから順に第 1 主成分, 第 2 主成分, . . . , 第 N 主成分にぞれぞれ対応している。つまり、固有値が大きいほど、多くの情報を持っているということになる。

3.2.2　主成分数の設定

主成分分析は、その特性上、主成分数をいくつに設定するかが重要となる。もし主成分数が少なすぎると、元のデータから損失するデータが大きくなってしまう。逆に、多すぎる場合、本来の役目である次元の削減がされず、主成分分析が意味を成さない。そこで、主成分の分散と等しい共分散行列の固有値に注目し、以下に示す 3 通りの方法によって主成分数を設定する。

- 固有値が 1 を越える主成分を採用する。
- ある固有値とその次の固有値の差が小さくなるまでの主成分を採用する。
- 累積寄与率がある値に達するまでの主成分を採用する。

1 つ目の方法は、平均と分散を 1 に標準化することで、分散がこの標準化された値である 1 よりも大きければ、説明力のある主成分として利用できるという考えに基づいている。2 つ目の方法は、ある固有値とその次の固有値の差が小さければ、主成分の採用、非採用の区別に大きな意味はないという考えに基づいている。3 つ目の方法は、主成分分析後のデータが、本来のデータから得られる情報の何割かを含んでいれば良いという考えに基づいており、普通は累積寄与率が 60〜80 パーセントに達するまでの主成分数を採用することが多い。

累積寄与率 (cumulative contribution ratio) は、寄与率 (contribution ratio) に関係しており、全ての主成分の寄与率を足し合わせた値である。寄与率とは、ある主成分の固有値が表す情報が、全ての情報の中でどの程度の情報を表しているかを示す値であり、次式で表される。

$$P_n = \frac{\lambda_n}{\sum_{p=1}^{P} \lambda_p} \tag{3.7}$$

ここで、λ_n は、n 番目の主成分の固有値を示す。この寄与率の総和が累積寄与率であるため、主成分数を N とすると、累積寄与率は次式で表される。

$$C_n = \sum_{i=1}^{N} P_i \tag{3.8}$$

3.2.3 LSA[3)]

潜在的意味解析 (LSA : Latent Semantic Analysis) は、単語の出現数に注目し、文書に出てくる単語の集合を解析し、文書を統計的にいくつかのカテゴリに分類する手法である。この手法は、式 (3.9) のような各文書における単語の出現頻度をまとめた単語-文書行列を分析の対象としている。単語-文書行列は、文書行列とも呼ばれ、形態素解析後に生成される行列であり、高次元な場合がほとんどである。行列が高次元のままでは、文書の分類や検索などの処理に必要な時間が膨大となる。また、高次元になるにつれて分類の妨げになる単語も増え、それがノイズのような存在となる。これらの問題を解決するために、LSA は用いられる。

$$TD = \left(\begin{array}{c|cccc} Term & doc_1 & doc_2 & \cdots & doc_N \\ \hline w_1 & I_{w_1,doc_1} & I_{w_1,doc_2} & \cdots & I_{w_1,doc_N} \\ w_2 & I_{w_2,doc_1} & I_{w_2,doc_2} & \cdots & I_{w_2,doc_N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_M & I_{w_M,doc_1} & I_{w_M,doc_2} & \cdots & I_{w_M,doc_N} \end{array}\right) \tag{3.9}$$

ここで、doc、w はそれぞれ N 個の文書、M 個の単語を示し、I_{w_M,doc_N} は doc_N における w_M の TfIdf の値を表す。

LSA では、特異値分解という高次元の行列を低次元に縮約する手法を、文書行列に適用して次元を削減している。以下に、式 (3.9) のような文書行列 TD を、特異値分解する式を示す。

$$TD = U\Sigma V^T \tag{3.10}$$

この式の U, Σ, V^T は行列を示す記号であり、右辺は 3 つの行列の積を表している。U,V^T は共に直交行列であり、それぞれを左特異 (ターム) ベクトル、右特異 (文書) ベクトルと呼ぶ。Σ は対角行列であり、この行列の対角成分が左辺の行列の特異値となる。この分解によって得られた左特異ベクトルの列ベクトルは、左に格納されている要素ほど重要度が高い。そのため、左から k 列だけ抜き出した行列を U_k とすると、以下の式により元の文書行列の近似であり、次元を削減した行列を作成できる。

$$TD_k = U_k^T TD \tag{3.11}$$

この特異値分解と先述の主成分分析とは非常に似通った手法であり、Python にて主成分

分析を実行できる scikit-learn ライブラリ内での処理にはこの特異値分解を行っているほどである。[4)]

3.2.4 LDA[3)5)]

潜在的ディリクレ配分法 (LDA : Latent Dirichlet Allocation) は、LSA に確率の概念を追加した pLSA を発展させた手法である。LDA は、pLSA にて検討する 2 つの確率密度関数に対して事前分布を仮定し、パラメータの事後分布を推定する。事前分布、事後分布にはディリクレ分布を利用することが一般的であり、これが LDA の名前の由来にもなっている。ディリクレ分布とは、ある n 個の事象について、i 番目の事象が $\alpha_i - 1$ 回発生する場合の確率が x_i である確率を表す分布である。確率変数 $X(x_1, \cdots, x_n)$ が $\alpha(\alpha_1, \cdots, \alpha_n)$ をパラメータとするディリクレ分布に従うとき、次の式で表される。

$$p(X;\alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{n} x_i^{\alpha_i - 1} \qquad B(\alpha) = \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)} \tag{3.12}$$

ここで、$x_i \geq 0$、$\Sigma x_i = 1$、$\alpha_i > 0$ である。また、x_i の期待値と分散については、次の式で表される。

$$E[x_i] = \frac{\alpha_i}{\sum_{j=1}^{n} \alpha_j} \tag{3.13}$$

$$var[x_i] = \frac{\alpha_i \sum_{j \neq i} \alpha_j}{(\sum_{j=1}^{n} \alpha_j)^2 (1 + \sum_{j=1}^{n} \alpha_j)} \tag{3.14}$$

LDA では、上記のようなディリクレ分布を利用して事前分布、事後分布を仮定した後、それらの分布を繰り返し更新していくことで、各文書が各トピックに属する確率と各単語が各トピックに属する確率を算出する。LSA に確率の概念を追加した pLSA と比較すると、汎化性能が高く、拡張しやすいという利点があり、新しいデータを投入してモデルの更新を継続する必要がある場合に適している。

3.2.5　クラスター分析

クラスター分析とは、与えられたデータから互いに類似したデータを集め、自動的に分類するデータ解析手法である。類似したデータの集合をクラスターと呼び、解析対象のデータをいくつかのクラスターに分けることで、分類を行う。クラスターに分ける際の基準は外部からは与えらないため、教師なしの分類手法である。クラスター分析は大きく分けて、階層的手法と非階層的手法の 2 種類に分かれる。

階層的クラスター分析は、データ間の距離が最も近いデータ同士をまとめることでクラスターを形成する。また、クラスター数は後から設定することができ、クラスターの最小数は全データを 1 クラスターとしたときの 1 で、最大数はデータの要素数となる。一方、非階層的クラスター分析は、データ間の距離が近いデータが同じクラスターになるようにクラスターを形成する。クラスター数は最初に決定し、そのクラスター数となるようにデータを分類する。

本研究では、階層的クラスター分析を利用した。階層的クラスター分析は、類似データをクラスターに分類する過程が樹形図 (デンドログラム) という図によって可視化できるため、結果をわかりやすく把握できると考えたためである。

階層的手法、非階層的手法共に、クラスター分析では、クラスター同士を結合させる処理があり、クラスター間の距離を測定する必要がある。本研究にて利用する階層的クラスター分析にはいくつかの距離測定方法が存在する。その中から、以下では、最近隣法、最遠隣法、群平均法、ウォード法について説明する。

最近隣法

2 つのクラスターに含まれる要素間の距離をそれぞれ求め、最も近い要素間の距離をクラスター間の距離とする手法。Figure 3.2(a) では、距離 $D_{(b,d)}$ をクラスター C_1 とクラスター C_2 の距離として、Figure 3.2(b) のクラスター C_3 を形成する。

最遠隣法

最近隣法とは逆の手法で、2 つのクラスターに含まれる要素間の距離をそれぞれ求め、最も遠い要素間の距離をクラスター間の距離とする手法。　Figure 3.2(a) では、距離 $D_{(a,d)}$ をクラスター C_1 とクラスター C_2 の距離として、Figure 3.2(c) のクラスター C_3 を形成する。

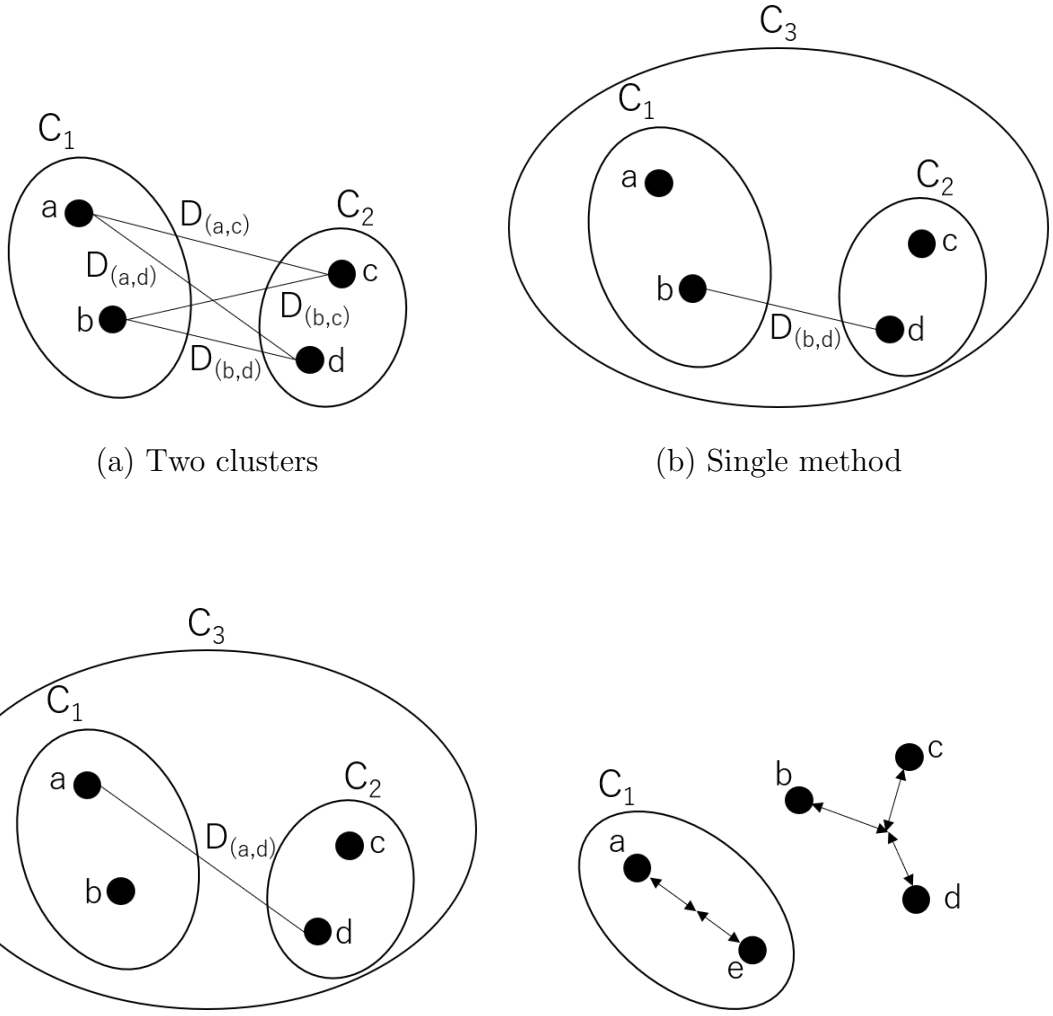
群平均法

2 つのクラスターに含まれる要素間の距離をそれぞれ求め、それらの距離の平均値

をクラスター間の距離とする方法。Figure 3.2(a) では、距離 $D_{(a,c)}$, $D_{(a,d)}$, $D_{(b,c)}$, $D_{(b,d)}$ の平均をクラスター C_1 とクラスター C_2 の距離として、新しいクラスターを形成する。

ウォード法

2 つのクラスターを融合した際に、同クラスター内の分散と他クラスター間の分散の比を最大化するようにクラスターを形成していく方法。Figure 3.2(d) の場合、データ a、e からなるクラスター C_1 を形成する。



(a) Two clusters　(b) Single method

(c) Complete method　(d) Ward method

Figure 3.2　Cluster

3.3 Python

Python は、1991 年にオランダ人のグイド・ヴァン・ロッサム氏によって開発された言語である。オープンソースであり、インターネットから無料で自由に利用することができる。英語のような文法でプログラムを書くことができ、特にプログラミング初心者に最適な言語の 1 つである。Python は初心者だけではなく、IT の最先端で働くプロフェッショナルからも愛用されており、Google や Microsoft などの企業でも頻繁に使用されている。先述の世界的企業で使用されるだけでなく、IT 業界を中心に人気が上昇しており、IEEE が公表した 2019 年度の人気プログラミング言語のランキングでは一位を獲得した。[6] その理由は、Python が可読性が高く、処理速度が速いコードを書けるからである。また、Python には NumPy や SciPy といった数値計算や統計処理を行う優れたライブラリが用意されている。これらのライブラリを使うことで、高次元行列、転置行列の生成や行列同士の内積といった処理を高速かつ、1 行で処理を完結できる。

3.3.1 TfIdf

Python では、scikit-learn ライブラリの TfidfVectorizer によって、簡単に TfIdf を求める事ができる。しかし、ライブラリを使用して出力される TfIdf は正規化されていたり、Idf の値に 1 が足されている等の処理が施されており、式 (3.1), (3.2) 通りの計算がされていない。正規化や 1 を足す処理は、引数を設定することで処理の有無を指定できるが、想定通りの計算をするため TfIdf は関数を自作し計算を行った。

3.3.2 LSA

Python では、scikit-lean ライブラリの TruncatedSVD 関数を使用することで、LSA によって次元削減された行列を出力できる。また、累積寄与率も同時に出力することが可能である。scikit-lean ライブラリでは、特異値分解を高速化するために、乱数ベクトルを生成し利用している。そのため、実行毎に異なる結果を返される。これを解決するには、random_state 引数に何らかの整数を指定する必要がある。指定する整数は一般的に 0 か 42 であり、本研究では 0 を指定した。

3.3.3 LDA

Python では、トピックモデルに特化した gensim ライブラリによって、LDA を行う関数が提供されている。gensim ライブラリでは、辞書、コーパス、TfIdf モデルを順に作成し、それらを基に LDA モデルを作成する。LSA と同様に、LDA でも random_state 引数に 0 を指定することで実行毎の結果の変動を避けた。このようにして作成した LDA モデルからは、各単語が各トピックに属する確率を抽出することができる。その情報から次元削減用の行列を作成した。

3.3.4 クラスター分析

Python では、scipy ライブラリの linkage 関数を使用することで、クラスター分析を実行できる。linkage 関数の引数により、クラスタリングの方法と距離の定義を指定できる。この関数は、階層型クラスタリングを行った結果を、木の情報を表す配列として返す。返された木の情報に対して、指定したクラスター数に分類する fcluster 関数や、デンドログラムとして結果を出力する dendrogram 関数などの処理を行う。

3.4 単語の意味を考慮した次元削減

3.4.1 Word2vec による単語間距離の算出 [7)]

Word2vec とは、2013 年に当時 Google 在籍であった研究者のトマス・ミコロフ (Tomas Mikolov) 氏によって提案された、言語データの分析や応用のための手法である。Word2vec は 2 層のニューラルネットワークであり、大量の文章を学習対象とすることで、文章に含まれている単語をベクトルとして表現可能となるモデルである。単語を表現しているベクトルは数百次元であり、cos 類似度によりベクトル間の類似度算出や、ベクトルの足し引きによって単語間の意味を足し引きできる。単語の足し引きとは、例えば「王」と「女性」を足すと、最も類似している単語として「王妃」が出力されるというようなことである。

本研究では、大量の文章から学習した Word2vec のモデルより、距離を求めたい 2 単語のベクトルを取得し、Word2vec のモデルを扱うライブラリに含まれている 2 単語間の距離を算出する関数を用いて単語間の距離を算出している。また、Word2vec はニューラルネットワークであるため、自分でモデルを学習させたり、学習済みのモデルを利用したりすることができる。本研究では、東北大学の乾、岡崎研究室にて作られたモデルを

利用する。このモデルは日本語 Wikipedia の本文を元に学習しており、各単語のベクトルは 200 次元に設定している。[8)]

3.4.2 日本語 WordNet による単語間概念距離の算出 [9)]

日本語 WordNet とは、2006 年から国立研究開発法人情報通信研究機構 (NICT) によって開発が始まった、日本語の概念辞書である。英語 WordNet を基に構築されており、英語 WordNet の synset に対応して日本語が付与されている。synset とは個々の概念がまとめられている単位であり、他の synset と意味的に結びついている。本研究では、シラバスに出現する各単語について、単語間の概念距離を求めるために使用した。

synset は上位から下位へ木構造の様に存在しており、概念間の関係を表している。この synset の関係から、単語間の概念距離を求める。ただし、最上位 (ルート) の synset は複数存在し、概念間で必ず上下関係があるわけではない。そのため、概念距離を求める方法は、2 単語の synset の関係によって次のように変化する。ここで、ある単語 a、b について、ルート synset からそれぞれの synset までの段数を L_a, L_b、二つの共通の synset の段数を C_{ab} とする。

同一の synset に存在する場合

概念距離は 0 とする。

異なる synset に存在する場合

概念距離は式 (3.16) より算出する。

異なる synset に存在し、synset 間に複数のルートがある場合

C_{ab} が最大となるようにルートをとり、C_{ab} が一致するものの中で式 (3.16) の最小値を概念距離とする。

複数の synset に属する場合

それぞれの距離を式 (3.16) より算出し、最小値を概念距離とする。

synset の関係が見つからない場合

概念距離は 1 とする。

ある 2 単語が持つ概念の類似度を $S_{a,b}$ とすると、L_a, L_b, C_{ab} を用いて次の式で表せる。

$$S_{a,b} = \frac{2C_{ab}}{L_a + L_b} \tag{3.15}$$

式 (3.15) は最大値が 1 になるよう正規化されているため、式 (3.16) によって、概念の類

似度 $S_{a,b}$ を概念距離 $D_{a,b}$ に変換できる。

$$D_{a,b} = 1 - S_{a,b} \tag{3.16}$$

3.4.3 クラスター分析による次元削減

クラスター分析は、本来は次元削減を行う技術ではない。しかし、単語間の距離、概念距離を利用して単語のクラスター分析を行うことで、単語をいくつかのクラスターに分類することはできる。そして、その結果を基に次元削減が可能であり、この手法をクラスター分析による次元削減とする。

処理対象の単語間の距離、概念距離をまとめた行列を対象としてクラスター分析を行うことで、式 (3.17) に示すクラスター-単語行列を生成できる。ここで、式 (3.17) の $C_i(i = 1, 2, \ldots, N)$ は各クラスター、 $w_j(j = 1, 2, \ldots, M)$ は各単語を表している。$a_{i,j}$ の計算式については、昨年までの研究で使用され、有効性が確認された計算式 (3.18)[10] を使用する。これらの処理によって生成する事ができる式 (3.17) のようなクラスター-単語行列を、単語-文書行列と掛け合わせることで、クラスター-文書行列を作成できる。以上の手順により、単語の意味を考慮した次元削減を実現できる。

$$CW = \left(\begin{array}{c|cccc} Term & w_1 & w_2 & \cdots & w_M \\ \hline C_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,M} \\ C_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_N & a_{N,1} & a_{N,2} & \cdots & a_{N,M} \end{array}\right) \tag{3.17}$$

$$a_{i,j} = \begin{cases} \frac{1}{|C_i|} \sum_{k \in C_i} S_{k,j} & (w_j \in C_i) \\ 0 & (w_j \notin C_i) \end{cases} \tag{3.18}$$

第4章　実験

本研究における実験対象には、本校電気情報工学科のシラバスを利用した。対象となったシラバスは121科目分であり、そのなかでも73科目が専門科目であった。実験対象シラバスへ形態素解析を行い、得られた単語の中でも名詞のみを利用し、次元圧縮を行った。この際、解析対象の名詞は3497語であった。

4.1　実験の準備

4.1.1　環境構築

先述の通り、本研究では実験にPythonを用いる。そのため、Pythonと統合開発環境であるPyCharmをインストールした。実験に必要なライブラリは、Pythonのパッケージ管理ソフトであるpipを使用してインストールした。また、形態素解析を行うためのライブラリMeCabは外部ライブラリであるため、有志によって開発されたインストーラを使用し、インストールを行った。しかし、外部ライブラリはインストールするのみではPythonから呼び出すことはできないため、MeCabの実行ファイルのパスを通した。

4.1.2　シラバスの取得

シラバスの取得は、Pythonにおいてスクレイピングを可能とするライブラリである Beautiful Soup4によって行った。具体的な手順としては、まず、各シラバスへアクセスできるページのURLをベースURLとし、ベースURLのスクレイピングを行う。次に、取得したHTMLの中からaタグのhref属性を抽出し、得られたURLを各シラバスへアクセスするターゲットURLとして保存した。それらのターゲットURLより再びスクレイピングを行い、各シラバスのHTMLを取得する。各シラバスのHTMLへ正規表現を使用して無駄なテキストや空白を省き、テキストファイルとして保存した。この時、正規表現はrequestsライブラリを利用し、テキストファイルを保存する際には文字コードをutf-8とした。

4.1.3　シラバスの形態素解析

取得したシラバスのテキストデータに対して、MeCabを使用して形態素解析を行った。MeCabによって形態素解析を行うと、各単語ごとに第2章1節5項MeCabで説明したよ

うな出力が得られる。この際、各単語の品詞が判明し、名詞の単語のみを抽出し、改行コード区切りでテキストファイルに保存した。名詞を保存する際の文字コードには、シラバスの保存と同様に utf-8 とした。

得られた名詞を観察してみると、本来は 1 つの単語である専門用語が分割されている事例が存在した。もちろん、これらの専門用語は 1 単語として処理することが理想的である。しかし、このような単語を発見、修正するには約 3000 単語ほどを人力で確認する必要があり、その作業は容易ではない。また、この修正作業で単語の修正漏れが発生すると、シラバスごとの条件や処理にばらつきが出てしまう。そのため、今回は MeCab による処理結果をそのまま実験に利用した。

4.1.4 TfIdf の計算

名詞を保存したテキストファイルを読み込み、自作の関数により TfIdf を計算した。計算した結果は、式 (4.1) のような文書-名詞行列として保存する。ここで、doc、no はそれぞれ N 個の文書、M 個の名詞を示し、I_{no_M,doc_N} は doc_N における no_M の TfIdf の値を表す。ベクトル空間モデルでは、一般的には式 (3.9) のような単語-文書の構造を持つ。しかし、実験で利用している Python では文書-単語の構造での取り扱いがデフォルトになっている。そのため、本研究では一般的な構造を転置した式 (4.1) の構造を採用し、以降の処理においても同様の構造を利用する。

$$\left(\begin{array}{c|cccc} Term & no_1 & no_2 & \dots & no_M \\ \hline doc_1 & I_{doc_1,no_1} & I_{doc_1,no_2} & \cdots & I_{doc_1,no_M} \\ doc_2 & I_{doc_2,no_1} & I_{doc_2,no_2} & \cdots & I_{doc_2,no_M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ doc_N & I_{doc_N,no_1} & I_{doc_N,no_2} & \cdots & I_{doc_N,no_M} \end{array}\right) \tag{4.1}$$

自作の関数の具体的な処理を説明する。まず、シラバスの形態素解析時に保存した各シラバスの名詞リストを 2 次元リストへ格納する。次に、シラバスを格納した 2 次元リストに対して sklearn ライブラリの関数 CountVectorizer を使用し、各シラバス内での各名詞の出現回数をカウントする。カウントした結果は、2 次元配列 A として保存する。次に、名詞の出現回数を格納したリストに対して numpy ライブラリの関数 sum を使用し、

各シラバスに含まれる名詞の総数を算出する。この結果は、2 次元配列 B として保存する。配列 A を配列 B によって割ることで、各シラバスごとの各名詞の Tf を算出できる。この結果を、2 次元配列 Tf に保存する。

ここまでの処理で Tf を求められたので、次に Idf を求める。まず、Python に標準で搭載されている関数 len により、シラバスの総数を取得する。次に、配列 A に対して numpy ライブラリの関数 count_nonzero を使用し、各名詞が 1 つでも含まれているシラバスの数をカウントする。この結果を 1 次元配列 C として保存する。シラバスの総数を配列 C で割り、対数をとることで、Idf を算出できる。この結果を、1 次元配列 Idf に保存する。

配列 Tf と配列 Idf を掛け合わせることで、今回取得したシラバスの TfIdf を算出でき、式 (4.1) のような行列を作成できる。

4.2 データの次元削減

4.2.1 主成分数の決定

先述の操作によって取得した式 (4.1) のような構造の行列に対して scikit-learn ライブラリの関数 PCA を使用し、次元削減を行った際の累積寄与率を求めた。その結果、主成分数が 54 の時に累積寄与率が 80 パーセント以上になった。そのため、これ以降の処理では主成分数を 54 とした。シラバスの総数は 121 個であるため、これから行う次元削減によって、67 次元分を削減することができる。

4.2.2 LSA による次元削減

前節 4 項にて作成した TfIdf 値が格納された文書-名詞行列に対し、LSA による次元削減を行った。この処理によって得られた行列は、コンマ区切りでテキストファイルに保存した。

4.2.3 LDA による次元削減

作成した LDA モデルから得られた各名詞が各トピックに属する確率を基に行列を作成した。その行列を文書-名詞行列に掛け合わせることで、次元削減を行った。この処理によって得られた行列は、LSA と同様に、コンマ区切りでテキストファイルに保存した。

4.2.4　クラスター分析による次元削減

クラスター分析による次元削減を行うには、まず名詞間の距離、概念距離を算出する必要がある。具体的な算出方法は、第3章4節1項、2項で説明した通りである。この処理を行うと、要素が名詞間の距離、概念距離である名詞-名詞行列が作成できる。また、3497語の名詞のうち、Word2vecでは210語、日本語WordNetでは941語が辞書に登録されておらず、それらについては処理の対象外とした。

作成した名詞-名詞行列に対して、クラスター数を54としてクラスター分析を行った。この際の距離測定方法には、昨年までの研究で有効性が確認されたウォード法[11)]を使用した。クラスター分析の結果、クラスター-名詞行列が生成される。最後に、クラスター-名詞行列へTfIdf値を格納した文書-名詞行列を転置して掛け合わせることで、クラスター-文書行列を算出する。これらの処理により、クラスター分析による次元削減を行った。

4.3　類似度の計算

LSA, LDA, クラスター分析のそれぞれの手法によって次元削減された行列について、文書間のcos類似度を算出した。その結果、要素を文書間のcos類似度とした文書-文書行列が生成される。

4.4　実験結果

4.4.1　実験結果の評価方法

実験結果の評価は、上述までの処理により作成した文書-文書行列に対して、ウォード法を用いたクラスター分析を適用し、その結果出力されるデンドログラムを比較することで行う。各手法を比較する際には、LSAによる次元削減を行った結果のデンドログラムを基準とする。また、このデンドログラムにおいて、各シラバスは「科目番号_科目名」と表示している。シラバスの表示に科目番号を利用しているのは、学年を超えて同一の名前の科目が存在しており、それらを区別するためである。

4.4.2　LSAによる次元削減を行った際の科目間類似度

LSAによる次元削減をしたデータについて類似度計算を行い、デンドログラムとして出力した結果をFigure 4.1に示す。

Figure 4.1 に注目すると、学年を超えて存在している同一の名前である科目が最初に結合している。全体的な特徴として、科目を大きく 3 分類しており、各々の分類を観察してみると実験や研究、専門科目、一般科目の分類がされていることがわかる。もちろん、人間が決めた基準によって科目を分類しているわけではないため、各分類にかけ離れた科目が存在している事例も散見される。また、科目の結合が全体的に早いことも読み取れる。

LSA による次元削減を行った際のデンドログラムは、ほぼ同系統の科目同士を同じクラスターに分類できており、実験結果の評価を行う際の基準とするのは問題ないといえる。

4.4.3 LDA による次元削減を行った際の科目間類似度

LDA による次元削減をしたデータについて類似度計算を行い、デンドログラムとして出力した結果を Figure 4.2 に示す。

Figure 4.2 に注目すると、LSA のクラスターの構成と大きく異なっている。デンドログラム全体の特徴として、LDA では科目を大きく 2 分類にしているように見える。分類の傾向を見るために詳しく科目を見てみると、早い段階での結合は同系統での科目同士となっている。しかし、その段階を過ぎると、分類の傾向が不明瞭となっていく。

LDA による次元削減を行った際のデンドログラムは、早い段階での結合は問題ないが、それ以降の結合に問題があるように見え、LSA よりも適切な分類はなされていない。

4.4.4 クラスター分析による次元削減を行った際の科目間類似度

クラスター分析による次元削減をしたデータについて類似度計算を行い、デンドログラムとして出力した結果を Figure 4.3 と Figure 4.4 に示す。Figure 4.3 は単語の概念距離を日本語 WordNet より取得、Figure 4.4 は単語の距離を Word2vec より取得している。

Figure 4.3 に注目すると、大きく分けて 3 もしくは 4 分類していることがわかる。どのような傾向で分類されているのかを見てみると、LDA のデンドログラムと同じように早い段階での結合は同系統の科目同士となっている。しかし、その段階を過ぎると、LDA と同様に分類の傾向が不明瞭になっていく。

日本語 WordNet より単語の意味を取得し、単語間の概念距離を利用するクラスター分析による次元削減には、LDA と同じような特徴が見られる。そのため、LDA と同様に、LSA よりも適切な分類はなされていない。

Figure 4.4 に注目すると、科目を大きく 2 分類している。分類の傾向を見ると専門科目と一般科目の 2 分類となっている。また、専門の中でも、電気系と情報系の科目に分かれている。専門科目の分類と考えられるクラスターを詳しく見てみると、専門科目に数学系の一般科目が分類されており、電気系により早く結合していることがわかる。これは、電気系の科目では数学的な知識が必要であり、情報系や一般科目の文系の科目よりも科目間のつながりが強いためであると考える。

Word2vec より単語の意味を取得し、単語間の距離を利用するクラスター分析による次元削減には、LSA のような各分類にかけ離れた科目が分類されているといったことが少なかった。そのため、LSA よりも適切な分類がされていたと考える。

4.5 考察

4.5.1 LSA, LDA, クラスター分析の比較

今回の実験では、基準としていた LSA よりも提案手法が適切に分類できているという結果になった。この結果の要因として、数学的な手法である LSA では考慮しきれていない人間が定義している単語の意味を、大量の文書から学習を行った Word2vec はより正確に計算に反映できたためと考える。LSA というのは、処理対象の文書とそこに含まれている単語について、関連した概念の集合を生成することで、単語間や概念間の関係を分析している。この手法では、あくまで与えられた文書内の潜在的な意味を解析している。一方、Word2vec は、2 層のニューラルネットワークの学習に十分である量の文書より学習を行い、単語をベクトル化している。これらのことより、より多量の文書より単語の意味を推測している Word2vec によって単語の意味を取得している本提案手法の方がより適切な分類ができたと考える。

上記のようなポジティブな理由以外に、そもそも LSA や LDA の解析を行う際のパラメータ設定が不適切であったのではないかというネガティブな理由もある。この理由が考えられるのは、本来は LSA を改良した手法である LDA が、LSA よりも適した分類になっていないためである。LDA モデルを作成する際には、反復する文書数や、閾値よりも低い確率のトピックの除外などといった設定ができる。これらを変化させることで、LSA よりも適切な文書分類が行える可能性がある。また、これらのことは LSA でも同様である。

以上のことより、従来手法と比較して提案手法の方が優れているとは断言できない。

しかし、提案手法の結果をそれ単体で見ても、文書の分類自体は上手くできている。そのため、本実験によって提案手法に一定の有効性があることを示せたと考える。

4.5.2 単語の意味の取得方法の比較

前項では、文書の類似度計算手法自体の比較を行ったが、本項では単語の意味を取得する方法の比較を行う。本研究の提案手法では、単語の意味を求めるために Word2vec の利用を提案しており、それと比較するために、日本語概念辞書である日本語 WordNet によっても単語の意味を求めた。前節 4 章より、明らかに、Word2vec によって単語の意味を求めた方が文書の分類を適切に行えていた。考えられる要因の 1 つとして、Word2vec と日本語 WordNet に登録されている語彙の差がある。この語彙の差は、Word2vec はコンピューターが大量の文書を自動的に学習する仕組みであるのに対して、日本語 WordNet は人手で概念辞書の整備がされているという差異に起因している。また、日本語 WordNet は英語 WordNet からの翻訳を行っており、それが原因で概念の関係が本当は意味的にずれがある場合がある。例えば、海洋動物の”seal”は「アザラシ」と訳されることが一般的だが、正しくは「アザラシ、アシカなどの総称」のため、”seal”の同義語に「アザラシ」を登録していないなどがある。

本実験の結果と、語彙の差、日本語 WordNet の概念関係と本来の意味とのずれより、日本語 WordNet よりも Word2vec の方が提案手法の単語の距離を計算するのに適しているといえる。

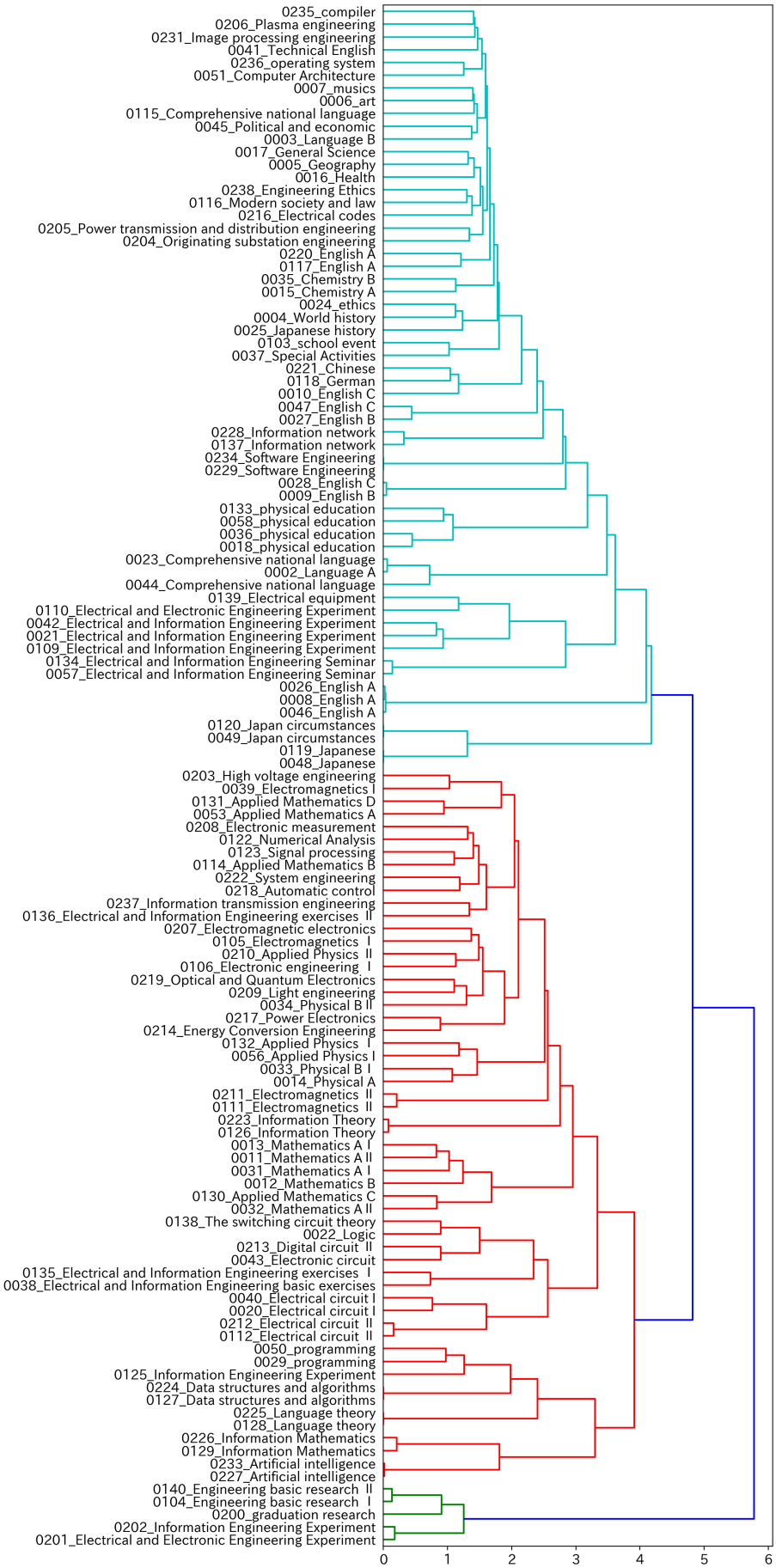


Figure 4.1　Result of dimension reduction by LSA

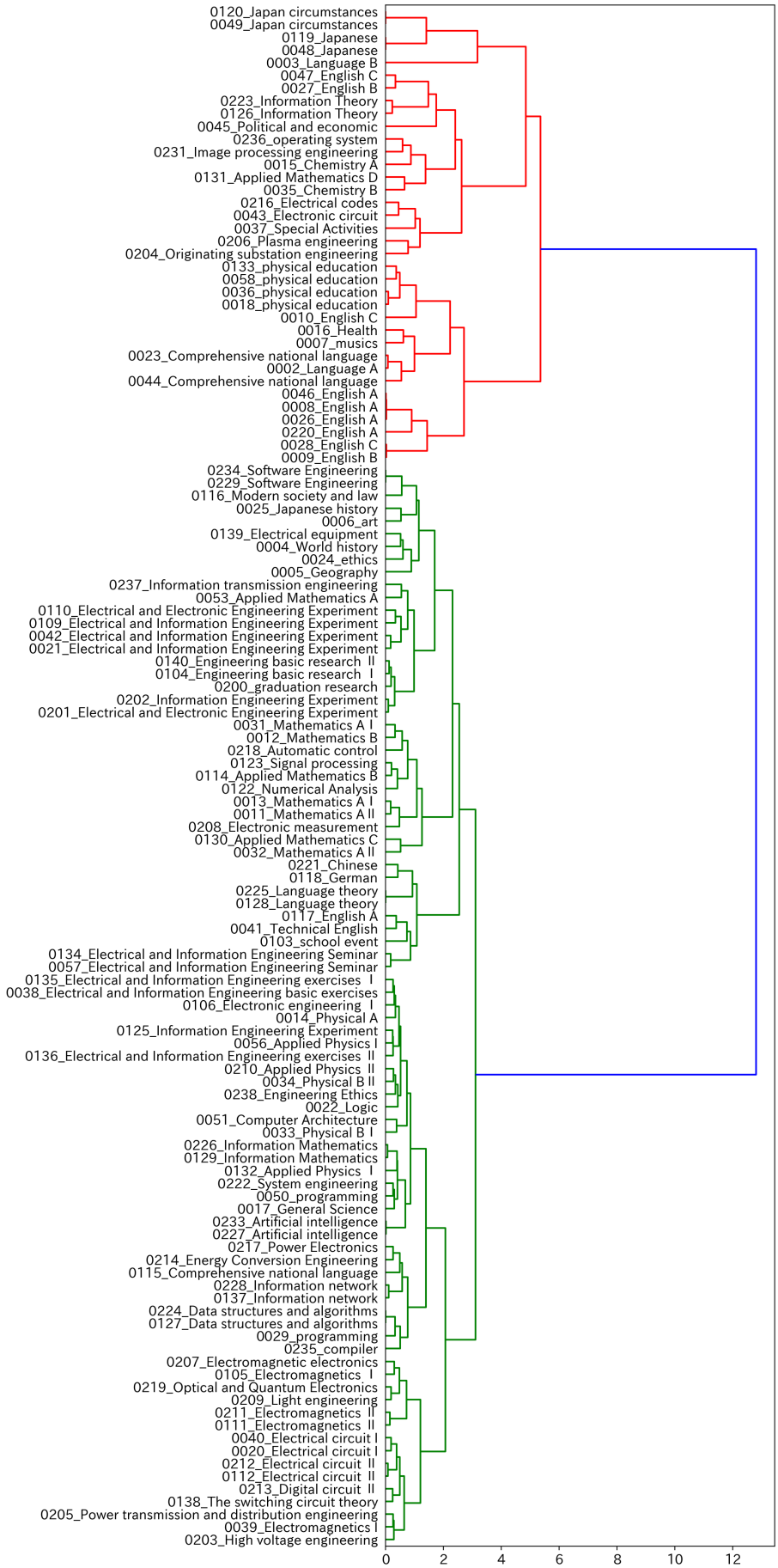


Figure 4.2　Result of dimension reduction by LDA

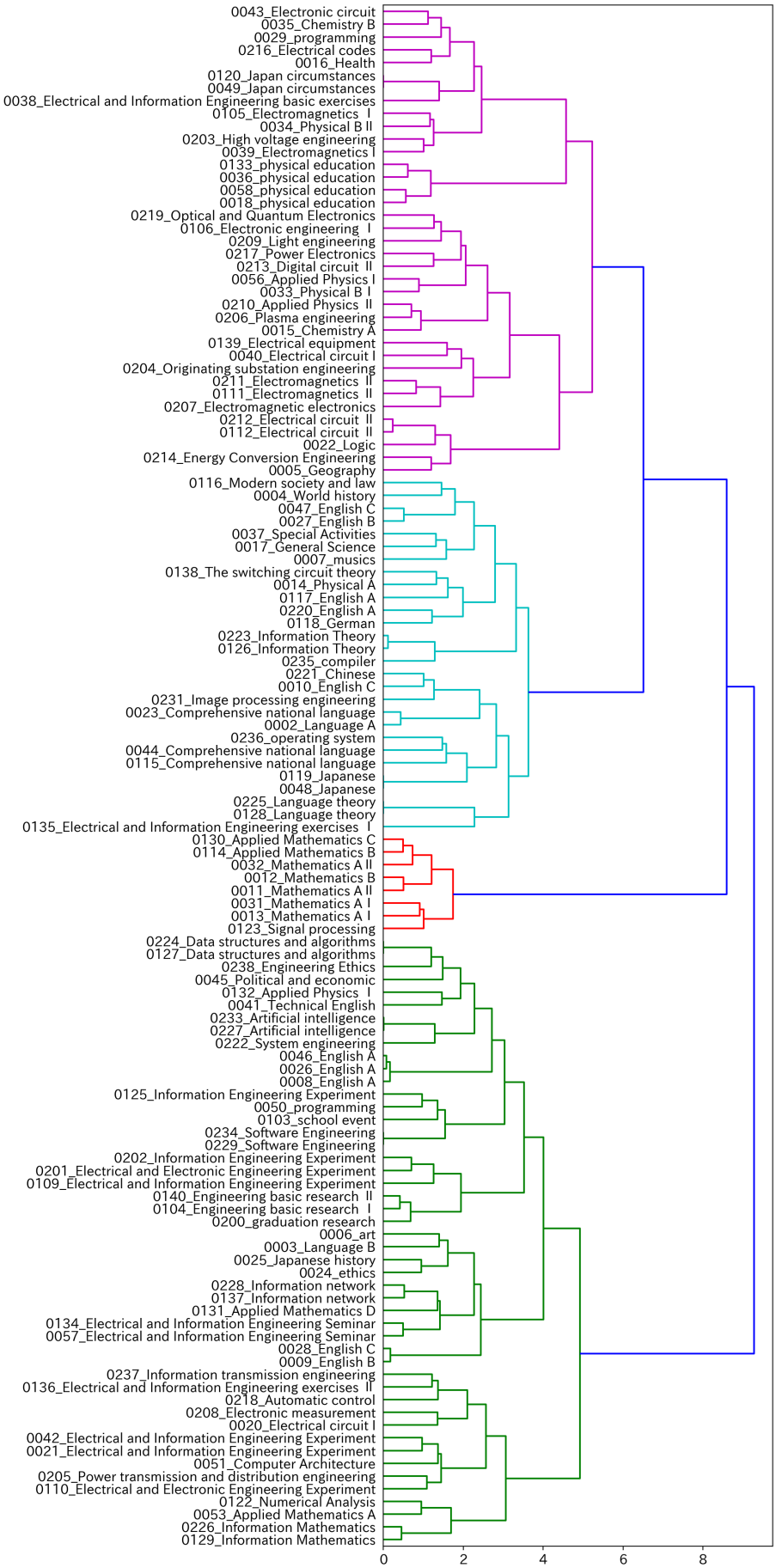


Figure 4.3 Result of dimension reduction by cluster analysis using JPWordnet

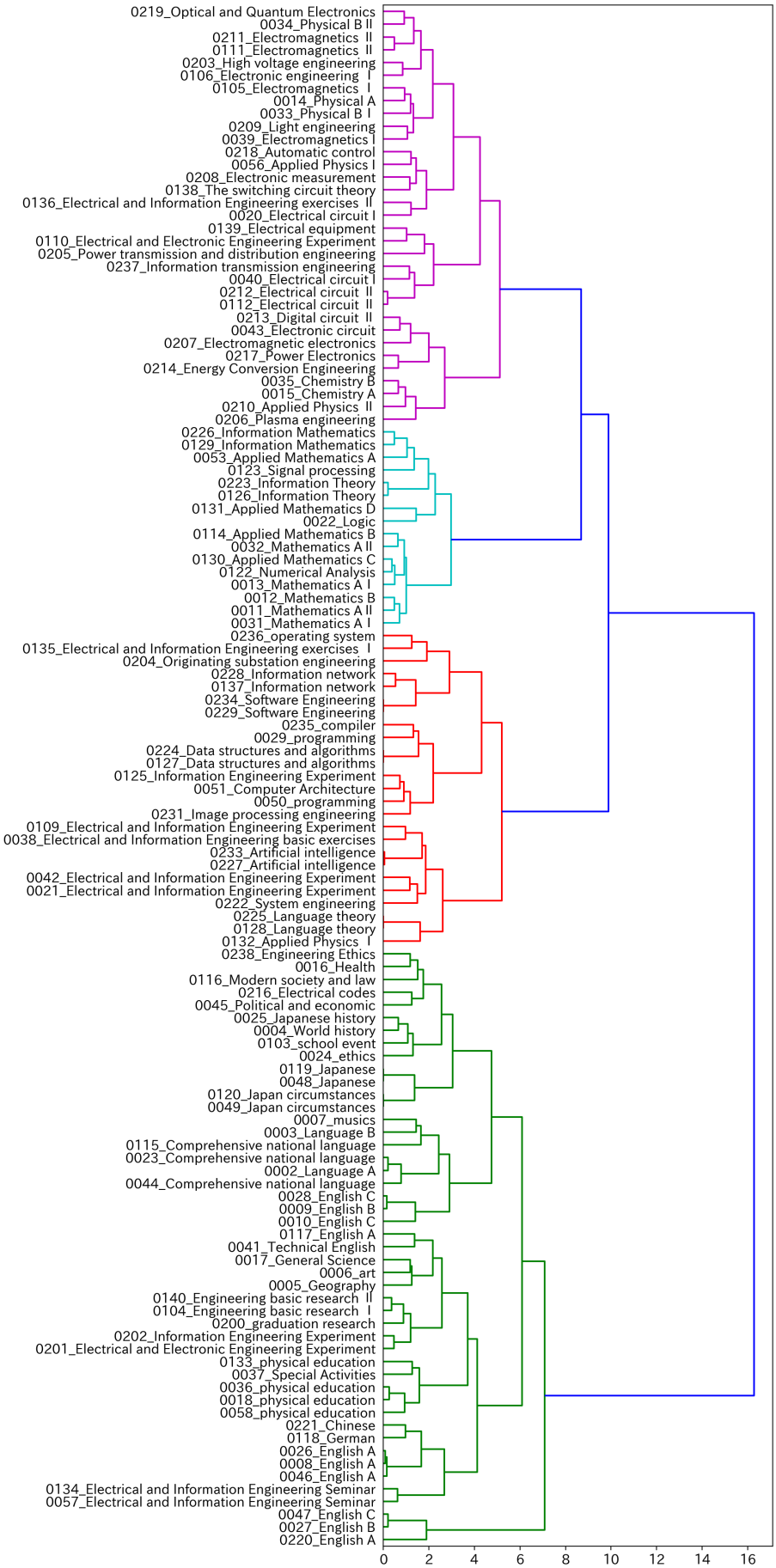


Figure 4.4　Result of dimension reduction by cluster analysis using Word2vec

第5章　結論

本研究では、提案手法の有効性を検討するため、本校電気情報工学科の科目のシラバスを対象とし、文書間の類似度計算を行った。提案手法は、Word2vec より求められる単語間距離とクラスター分析を利用した方法である。有効性の検討のため、従来手法として LSA、LDA による次元削減の結果とも比較を行った。また、単語の意味を取得する手法の比較として、日本語 WordNet によって求められる単語間の概念距離を利用した方法でも次元削減を行った。

本研究の実験では、名詞間の TfIdf 値を算出するまでは、形態素解析などの各手法共通の処理を行った。TfIdf 値算出後は、LSA、LDA、クラスター分析による次元削減を行った。また、クラスター分析による次元削減を行う際には、単語の意味を取得する手法の比較のため、Word2vec より求められる単語間距離、日本語 WordNet より求められる単語間概念距離をそれぞれ利用した。ここまでの処理により、シラバスのデータの次元削減が完了する。次元削減後のデータに対して、cos 類似度を算出してシラバス間の類似度を求め、ウォード法によるクラスタリングの結果をデンドログラムとして出力した。

デンドログラムを比較すると、Word2vec より求められる単語間距離を利用する提案手法が最適な分類であった。昨年までの実験結果 [11] より、従来手法のどれかがより適した分類をすると予想していたため、予想と反する結果であった。この理由として、第 4 章 5 節 1 項に記述したような単語の意味を考慮する際の学習対象の規模の違いや、従来手法のパラメータ設定の可否が考えられる。また、単語の意味を取得する手法が変更されるだけで、昨年までより提案手法の分類が適した結果となった。この理由として、第 4 章 5 節 2 項にて記述したような語彙の差や日本語 WordNet の概念関係と本来の意味とのずれが要因として考えられる。Word2vec と日本語 WordNet の語彙の差というのは、第 4 章 2 節 4 項の処理対象外とした名詞の数からもわかる。処理対象外とした名詞の具体的な割合は、Word2vec は約 6%、日本語 WordNet は約 26.9%であった。この差は明らかに大きく、単語の意味を取得する手法として Word2vec の方が適しているのは当然といえる。

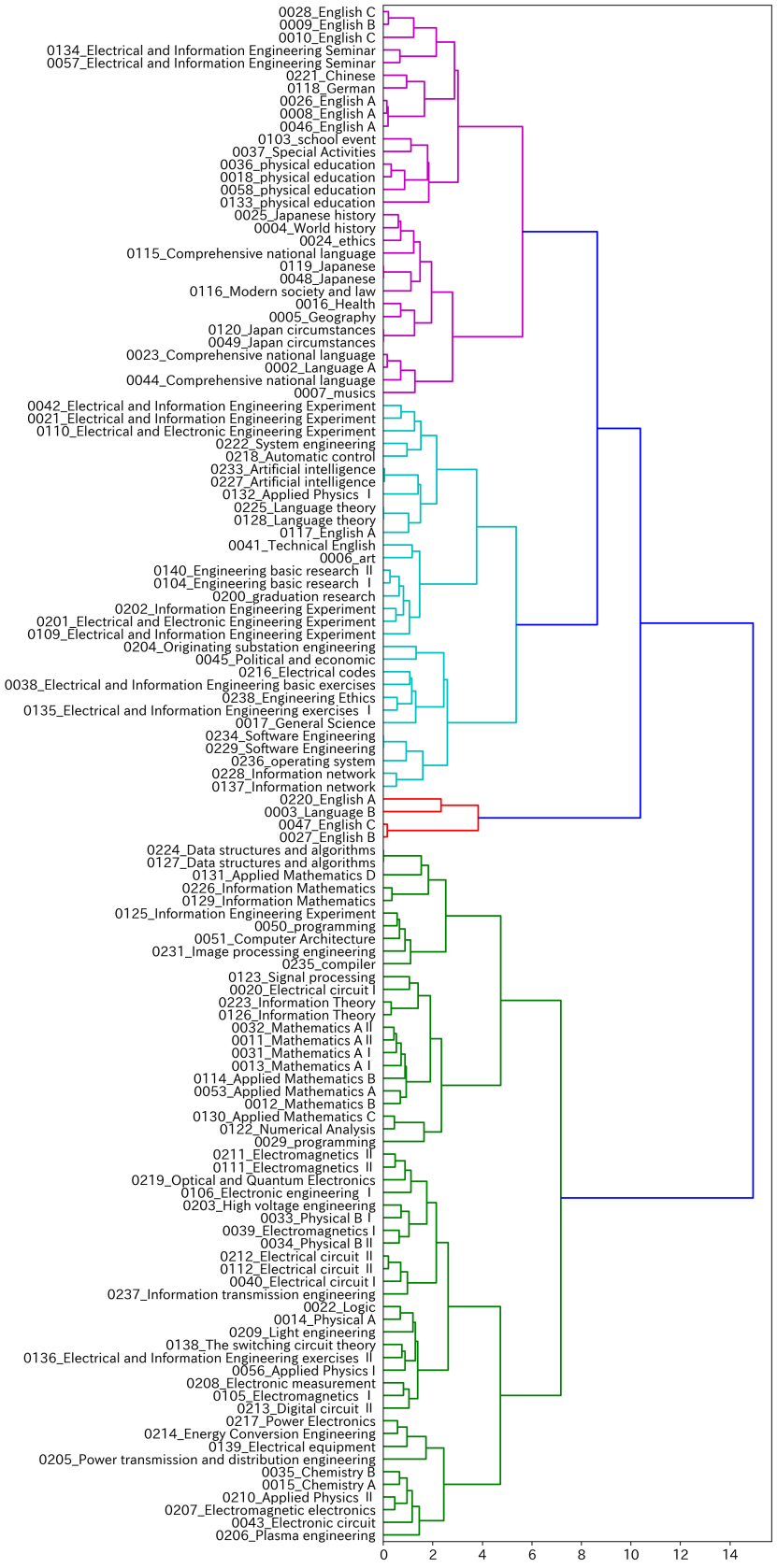
今回の実験では、従来手法より提案手法が適した分類ができていたが、削減次元数の変化が日本語 WordNet の結果に影響が発生することが判明している。[10]Word2vec を利用したクラスター分析による次元削減において、削減次元数を変化させるため、累積寄与率が 60%、70%、90%の主成分数を次元削減に利用した際の結果も出力した (それぞれ

のデンドログラムは付録を参照)。それらの比較を行うと、累積寄与率が高くなると、分類が改悪となってしまうことが判明した。これは、累積寄与率が高くなると主成分数も多くなり、ある意味で余計な情報がクラスタリングの際に残ってしまうことが原因であると考えられる。逆に累積寄与率が低くなると、主成分数が過少になり、これもまた改悪となってしまう。そのため、現在の累積寄与率 80%の主成分数が提案手法には適しているといえる。

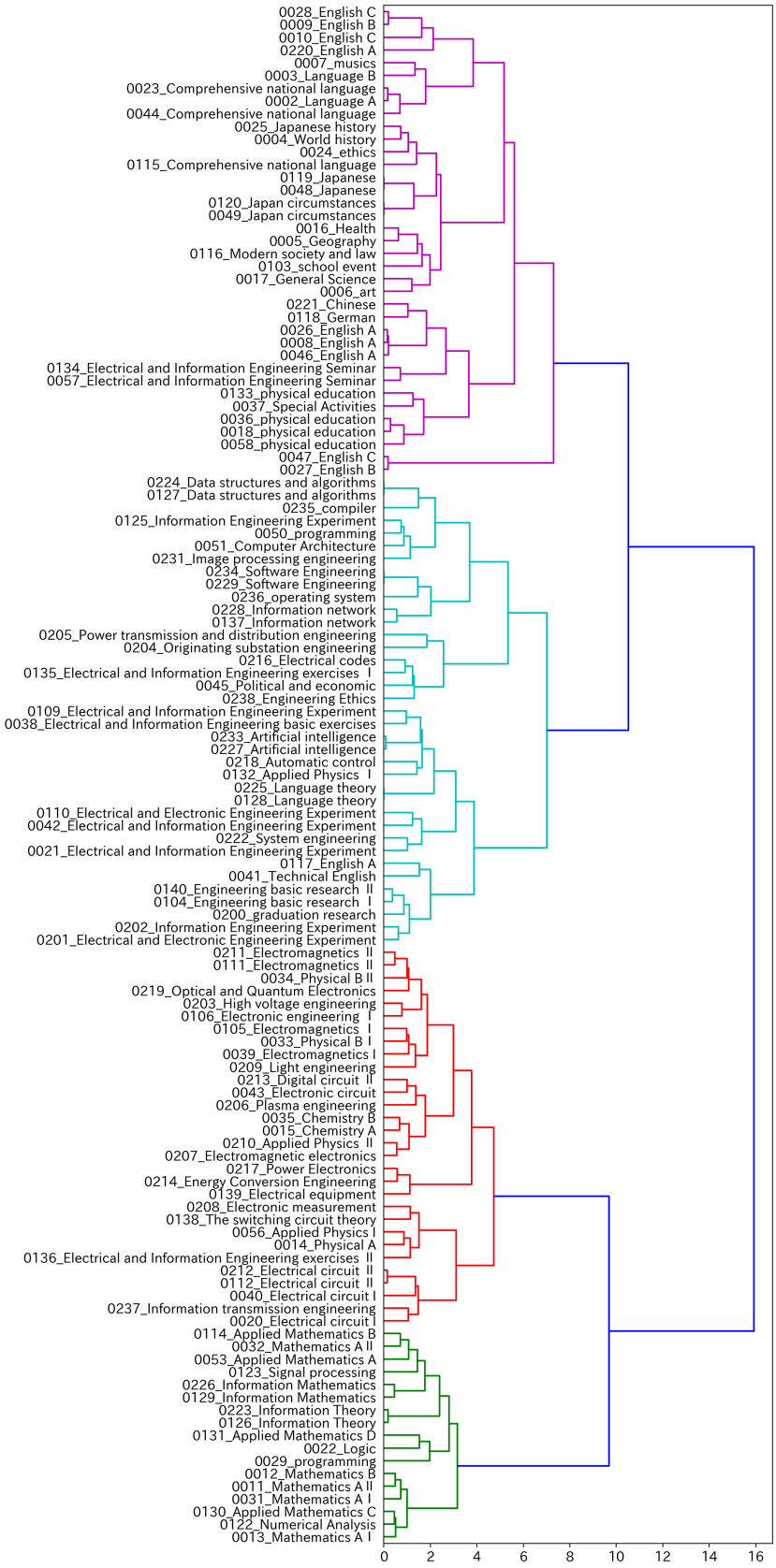
謝辞

本研究を進めるにあたり、ご多忙中にもかかわらず適切な指導をしていただきました出口利憲先生に深く感謝申し上げます。同時に、助言や協力を頂いた同研究室の皆様にも厚く御礼申し上げます。

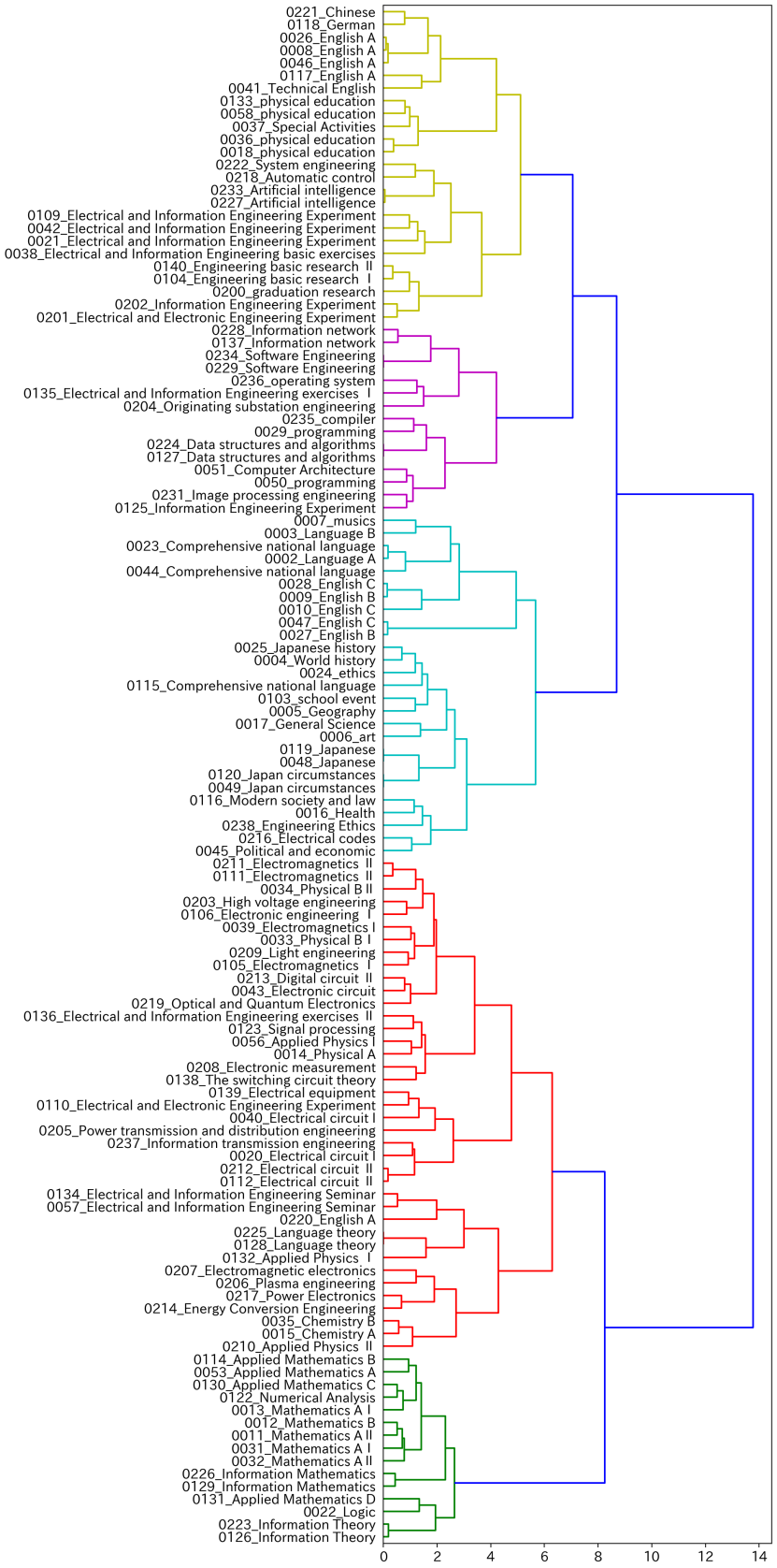
付録 A　累積寄与率 60%時のデンドログラム



付録B　累積寄与率70%時のデンドログラム



付録C　累積寄与率90%時のデンドログラム



参考文献

1) 東京大学, UTokyo Online Education データマイニング入門 2018 森 純一郎. https://ocwx.ocw.u-tokyo.ac.jp/course_11414/ (2021 年 1 月 20 日アクセス).
2) 加納学, 主成分分析, 京都大学大学院工学研究科化学工学専攻プロセスシステム工学研究室, 1997. http://manabukano.brilliant-future.net/document/text-PCA.pdf
3) Developers.IO, 機械学習_潜在意味解析_理論編 https://dev.classmethod.jp/articles/2017ad_20171220_lsa/#sec 6 (2021 年 1 月 15 日アクセス).
4) scikit-learn, sklearn.decomposition.PCA. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html (2021 年 1 月 20 日アクセス).
5) 奥村学 監修, 佐藤一誠 著, 自然言語処理シリーズ 8 トピックモデルによる統計的潜在意味解析, コロナ社, 2015.
6) IEEE, The 2019 Top Programming Languages https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019 (2021 年 1 月 15 日アクセス).
7) Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space, 2013, ICLR. https://arxiv.org/pdf/1301.3781v3.pdf (2020 年 6 月 5 日アクセス).
8) 鈴木正敏, 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/ (2020 年 6 月 5 日アクセス).
9) Francis Bond, Timothy Baldwin, Richard Fothergill and Kiyotaka Uchimoto, Japanese SemCor: A Sense-tagged Corpus of Japanese, The 6th International Conference of the Global WordNet Association (GWC-2012), 2012. http://compling.hss.ntu.edu.sg/wnja/pubs/2012-gwc-jsemcor.pdf
10) 服部修平, テキストマイニングによる文書の類似度計算に関する研究, 岐阜工業高等専門学校電気情報工学科卒業研究報告, 2017.

http://www.gifu-nct.ac.jp/elec/deguchi/sotsuron/hattori/

11) 長尾彪真, 文書の類似度計算における次元削減手法に関する研究, 2019.
http://www.gifu-nct.ac.jp/elec/deguchi/sotsuron/nagao.pdf