Web サーチエンジンを用いた
名詞間類似度の測定

藤井丈明
茨城大学大学院
理工学研究科

平成18年3月7日

概要

自然言語処理において，名詞間の類似度を測定することは非常に重要である．なぜなら多くの自然言語処理システムでは，単語間の類似度を測る処理を本質的に必要としているからである．例えば，事例ベースの手法では事例間の距離を測る処理が必須であり，その処理の核には名詞間類似度の測定がある．また，名詞間類似度を測定することができれば，シソーラスを自動構築できる．これはシソーラスを利用したシステムでは，シソーラスを使う部分を名詞間類似度を測る処理に置き換えられることを意味している．つまり名詞間類似度の測定は自然言語処理の重要な要素技術と言える．

通常，名詞間類似度はシソーラスを用いて測定される．シソーラスとは単語をある目的に応じて分類したものである．一般に上位，下位関係が記されており，階層構造すなわち木構造をなしている．この階層関係を利用することで二つの単語の意味的な距離を測ることができる．しかし，シソーラスでは，登録されていない単語（未知語）が必ず存在する．そのためシソーラスを使う場合，未知語に対する名詞間類似度を測定することは不可能である．そこで，コーパスを用いて名詞間類似度を測定する方法が提案されている．コーパスとは大量の電子文書データのことである．コーパスを用いた場合，基底単語群との共起頻度を用いて単語をベクトル化し，名詞間類似度を測定する．基底単語群とは，単語をベクトル化する際に指標となる単語列のことである．しかし，コーパスを利用する場合，頻度の低い単語に関しては正しい名詞間類似度を測定できないというスパース性の問題が発生する．

そこで，本研究ではWebサーチエンジンを利用し，名詞間類似度を測定することを提案する．Web文書の数は莫大であるため，未知語の問題が生じず，名詞間類似度を正しく測定できると考える．まず，Webに適した基底の単語群b1,b2,…,bnを選出し，対象の単語aと基底の単語biからクエリ"a bi"を作り，Webサーチエンジンを利用して単語aと単語biの共起したページ数fiを得る．(f1,f2,…,fn)を正規化することで単語aの特徴ベクトルを作成する．単語をベクトルで表現できれば，任意の単語間の類似度を測定することができる．

ここでは本手法を利用した3つの実験を行った．語義識別，単語クラスタリング，未知語との類似度測定，の3つである．語義識別とは，文章中にある単語が複数の意味を持つ場合，どの意味をとるかを識別するタスクである．未知語との単語間類似度が必要な部分に本手法を利用した．実験の結果，語義識別の正解率が向上したことから，本手法の有効性が確認できた．単語クラスタリングとは，名詞間類似度から概念の似ている単語同士を一つのクラスタとしてまとめることである．意味的に近い単語が一つのクラス内に存在している場合，名詞間類似度が正しく測定されていると判断できる．２５単語を適当に選出し，本手法によりクラスタリングを行った．また，この実験では，基底単語についても評価した．基底単語を新聞記事から選出した場合とWeb文書から選出した場合とを比較した．その結果，基底単語は新聞記事よりもWeb文書から選出したほうが良いという結果

が得られた．しかし，正しいクラスタリング結果は得られなかったため，通常の単語に関しては本手法は有効ではないと考える．未知語との類似度測定の実験では，未知語を選び，その未知語と似ている意味の単語２個（A 群）と似ているが少し意味が異なる単語２個（B 群）を選出し，それら４単語と未知語間の距離を測定した．未知語と最も類似した２つの単語がA 群の単語である場合，正解とする．この実験を３０個の未知語に対して行った．結果，３０個中１１個の正解を得た．ランダムに並べた場合に正解する個数は５個である．

以上の実験の結果，本手法は未知語に対して有効であると言えるが，一般の単語については有効であるとはいえない．これは，基底の単語選出の際に用いた単語が新聞記事を利用して得られたためであると考える．また，Web を利用して基底単語を選出する際に用いたデータも，Web 文書全体ではなく一部のデータであったことも問題であると考える．また，今回，基底の単語数を１００単語として実験を行った．単語クラスタリング，未知語との類似度測定の実験において，基底単語の数を変化させた場合，正解率に変化が生じた．このことから，名詞間類似度は基底単語数によって結果が異なると考えられる．そこで，最適な基底単語及び基底単語数を選出することができれば，さらに精度が向上できると考える．基底単語の最適な作成方法及び基底単語数を検討することが今後の課題である．

Abstract

It is very important to measure the similarity between nouns in the natural language processing, because many natural language processing systems need to measure the similarity between nouns essentially. For example, the example based method needs to measure the similarity between instances, and the core process of it is to measure the similarity between nouns. Furthermore, if we can measure the similarity between nouns, we can construct a thesaurus automatically. This means that we can use the similarity between nouns instead of thesaurus in the system using the thesaurus. Therefore, the measurement of the similarity between nouns is an important technology of the natural language processing.

Generally, the similarity between nouns is measured by a thesaurus. A thesaurus is sorted a word by the purpose. General thesaurus describes super-subrelation of words, and has hierarchical structure, that is tree structure. By the hierarchy, we can measure the semantic the similarity of two

words. However, a thesaurus has a not registed word (an unknown word). Thus, we cannot measure the the similarity between nouns if the noun is unknown word. To overcome the problem, the method to measure the similarity between nouns by using the corpus is proposed. A Corpous is a lot of electronic document data. When a corpus is used, a word is transformed into a vector by using the cooccurrence frequency with base words. Then, the the similarity between nouns is measured. The base words is the set of words that is corresponding to the base vector when a word is transformed into a vector. However, the use of corpus has a big problem that is the sparseness problem. We cannot measure the proper the similarity between nouns if the frequency of the noun is low.

To overcome above problems, I propose the method to use Web search engine to measures the the similarity between nouns. The Web document is huge, so unknown word problem does not happen. So, we can measure the the similarity better. Let's consider transforming the word "a" into a vector. First, the set of base words b1, b2$\cdots$bn, which is suited for the Web,

is selected. Next, we give the query "a bi" to the Web search engine, and get the frequency fi that is the number of pages in which words "a" and "bi" appear. We normalize (f1, f2$\cdots$fn) to make the feature vector of the word "a". If the word can be expressed with a vector, it is possible to measure the similarity between any words.

I conduct three experiments using my proposed method;

1) the word sense disambiguation,

2) the word clustering,

3) measurement of unknown word.

The word sense disambiguation is a task to judge the sense of ambiguous word. I

use my proposed method when I measure the the similarity between an unknown noun and a noun. The result of this experiment showed that my method is useful, because the precision of this task was improved. The word clustering is a task to divide words into some groups according to the sense the similarity. If the clustering task works well, it means that the measure of the the similarity is excellent. In the experiments, I selected 25 words at random, and conducted the clustering of these words. I used my method to measure the the similarity between nouns. At same time, I evaluate the used base words in this experiment. To do it, I compared the base words selected from newspaper articles with the base words selected from the Web documents. As the result, it is shown that the base words selected from the Web documents is better. However, I could not get the correct clusters in this experiment. Thus, my method is not effective for usual words. The third experiment was conducted as follows. First, I selected an unknown word. Next, I selected two words similar to that unknown word. I named these two words as A-group. Next, I selected two words whose meaning is a little different from the unknown word. I named these two words as B-group. Last, I measured the the similarity between the unknown word and each 4

words. If the two similarest words are in A-group, I evaluate this test as

success. This experiment was conducted for 30 unknown words. The number of success was 11. In the random choice, the average number of success is 5.

These experiments showed that my method is effective for the the similarity between an unknown noun and a noun, but not for the the similarity between general nouns. I guess two reasons. First is that the base words were selected from newspaper articles. Also used the base words selected from the Web documents in the

experiment. However, the quantity of the used Web documents is small. That is the second reason. In above experiments, the number of base words is fixed 100. In additional experiment, I change the number of base words, and conducted the clustering and measurement of unknown word again. The precision was changed. I think that the number of base words is an essential factor. If we can find the best number of base words, the precision will be improved more. In future, I will investigate the method to make the base words and to find the number of basis words.

目 次

第1章 序論 **4**
- 1.1 概要 4
- 1.2 本論文の構成 6

第2章 名詞間類似度の測定 **7**
- 2.1 シソーラスを用いた方法 7
- 2.2 コーパスを用いた方法 9
  - 2.2.1 特徴ベクトルの作成 10
  - 2.2.2 属性ベクトルの作成 11
  - 2.2.3 類似度計算 11
- 2.3 名詞間類似度の利用例 12
  - 2.3.1 語義識別での利用 12
  - 2.3.2 シソーラスの自動構築 13
- 2.4 未知語の問題 14

第3章 Web サーチエンジンの利用 **21**
- 3.1 単語のベクトル化 21
- 3.2 基底単語 21
- 3.3 On the fly の類似度の測定 22

第4章 評価実験 **25**
- 4.1 基底単語の選出 25
- 4.2 クラスタリング 25
- 4.3 単語間類似度の測定 37

第5章 考察 **54**
- 5.1 クラスタリング 54
- 5.2 名詞間類似度の測定 55

第6章 結論 **56**

第7章 謝辞 **57**

付 録A プログラムソースリスト **59**

図目次

2.1 分類シソーラス 8
2.2 上位下位シソーラス 9
2.3 素性の説明 12
2.4 語義の連想 15

3.1 Web サーチエンジンを用いた共起頻度取得 22
3.2 on the fly による特徴ベクトルの作成 24

4.1 ward 法 (基底：Web, コーパス:Web) 28
4.2 k-means 法 (基底：Web, コーパス:Web) 29
4.3 ward 法 (基底：新聞記事, コーパス:Web) 30
4.4 k-means 法 (基底：新聞記事, コーパス:Web) 31
4.5 ward 法 (基底：新聞記事, コーパス:新聞記事) 32
4.6 k-means 法 (基底：新聞記事, コーパス:新聞記事) 33
4.7 50 個の基底単語によるクラスタリング 35
4.8 25 個の基底単語によるクラスタリング 36

表目次

2.1 「うまれる」の語義 15
2.2 「開発」の語義 16
2.3 「精神」の語義 16
2.4 「核」の語義 16
2.5 「のる」の語義 17
2.6 「かかる」の語義 (1) 18
2.7 「かかる」の語義 (2) 19
2.8 「かかる」の語義 (3) 20
2.9 「かかる」の語義 (4) 20

4.1 Web 文書から得た基底単語 (100 単語) 26
4.2 Web 文書から得た基底単語 (50 単語) 34
4.3 Web 文書から得た基底単語 (25 単語) 34
4.4 実験対象単語 37
4.5 名詞間類似度の測定対象単語 38
4.6 名詞間類似度の測定 (100 単語) 39
4.7 名詞間類似度の測定 (100 単語) 40
4.8 名詞間類似度の測定 (100 単語) 41
4.9 名詞間類似度の測定 (100 単語) 42
4.10 名詞間類似度の測定 (100 単語) 43
4.11 名詞間類似度の測定 (50 単語) 44
4.12 名詞間類似度の測定 (50 単語) 45
4.13 名詞間類似度の測定 (50 単語) 46
4.14 名詞間類似度の測定 (50 単語) 47
4.15 名詞間類似度の測定 (50 単語) 48
4.16 名詞間類似度の測定 (25 単語) 49
4.17 名詞間類似度の測定 (25 単語) 50
4.18 名詞間類似度の測定 (25 単語) 51
4.19 名詞間類似度の測定 (25 単語) 52
4.20 名詞間類似度の測定 (25 単語) 53

第1章　序論

1.1　概要

自然言語処理において，名詞間の類似度を測定することは非常に重要である．なぜなら多くの自然言語処理システムでは，単語間の類似度を測る処理を本質的に必要としているからである．例えば，事例ベースの手法では事例間の距離を測る処理が必須であり，その処理の核には名詞間類似度の測定がある．また，名詞間類似度を測定することができれば，シソーラスを自動構築できる．これはシソーラスを利用したシステムでは，シソーラスを使う部分を名詞間類似度を測る処理に置き換えられることを意味している．つまり名詞間類似度の測定は自然言語処理の重要な要素技術と言える．

通常，名詞間類似度はシソーラスを用いて測定される．シソーラスとは単語をある目的に応じて分類したものであり，各語について同義語，類義語，上位語，下位語，反義語，対義語などを記述した辞書である．通常は意味が似ているものを集めて分類し，分類したグループをさらにグループ化していくと，概念階層ができる．一般に上位，下位関係が記されており，階層構造すなわち木構造をなしている．この階層関係を利用することで二つの単語の意味的な距離を測ることができる．また，シソーラスには語義の曖昧性解消ができる，単語を抽象的な集合の一要素として扱える，規則を一般化して書くことができるので汎用性が高くなる，直接規則に無くても類似度が高ければ推論して規則を用いることができる，などの利点があり，意味的に近い語ほど近くのノードに属している．しかし，シソーラスでは登録されていない単語（未知語）が必ず存在する．そのためシソーラスを使う場合，未知語に対する名詞間類似度を測定することは不可能である．そこで，コーパスを用いて名詞間類似度を測定する方法が提案されている．

コーパスとは大量の電子文書データのことである．コーパスを用いた場合，基底単語群との共起頻度を用いて単語をベクトル化し，名詞間類似度を測定する．基底単語群とは，単語をベクトル化する際に指標となる単語列のことである．コーパスの利点として，大量のデータによる網羅性，文書における出現単語の分布明示性等が挙げられる．網羅性を用いて，シソーラスでは不可能な未知語に対するアプローチが期待できる．コーパスを利用した自然言語処理はある程度の成功をおさめているが，一方でスパース性の問題が発生してしまう．スパース性とは，単語間の出現頻度における偏りのことである．多くの文書中に高頻度で出現する単語もあれば，新聞記事10年分を調べても，数回しか出現しない単語もあると報告さ

れている．そのため，頻度の低い単語に関しては十分な共起頻度を得られず，名詞間の類似度を正しく測定きない．この問題を解決するため，本研究ではWebサーチエンジンを利用し，Web文書をコーパスに用いることを提案する．Web文書は情報が莫大であるため，コーパスのスパース性を回避し，名詞間類似度を正しく測定できると考える．

そこで，本研究ではWebサーチエンジンを利用し，名詞間類似度を測定することを提案する．Web文書の数は莫大であるため，未知語の問題が生じず，名詞間類似度を正しく測定できると考える．まず，Webに適した基底の単語群b1,b2,…,bnを選出し，対象の単語aと基底の単語biからクエリ"a bi"を作り，Webサーチエンジンを利用して単語aと単語biの共起したページ数fiを得る．(f1,f2,…,fn)を正規化することで単語aの特徴ベクトルを作成する．単語をベクトルで表現できれば，任意の単語間の類似度を測定することができる．

そこで本手法を用いた実験を3つ行った．語義識別，単語クラスタリング，未知語との類似度測定，の3つである．語義識別とは，文章中にある単語が複数の意味を持つ場合，どの意味をとるかを識別するタスクである．未知語との単語間類似度が必要な部分に本手法を利用し，on the fly の類似度の判定を提案した[1]．以前行った実験の結果，語義識別の正解率が向上したことから，本手法の有効性が確認できた．単語クラスタリングとは，名詞間類似度から概念の似ている単語同士を一つのクラスタとしてまとめることである．意味的に近い単語が一つのクラス内に存在している場合，名詞間類似度が正しく測定されていると判断できる．まず，Web文書を利用して基底単語を選出する．これは，Webサーチエンジンを利用して名詞間類似度を測定するため，基底単語もWeb文書を利用した方が良いと考えたためである．次に25単語を適当に選出し[2]，本手法によりクラスタリングを行った．そこでコーパスに新聞記事を利用した場合とWebを利用した場合とを比較した．また，この実験では，基底単語についても評価した．基底単語を新聞記事から選出した場合とWeb文書から選出した場合とを比較した．つまり，Webを利用して得られた基底単語とWebをコーパスに用いたもの，Webを利用して得られた基底単語と新聞記事をコーパスに用いたもの，新聞記事を利用して得られた基底単語と新聞記事をコーパスにもちいたものとを比較し評価する．未知語との類似度測定の実験では，未知語を選び，その未知語と似ている意味の単語2個（A群）と似ているが少し意味が異なる単語2個（B群）を選出し，それら4単語と未知語間の距離を測定した．未知語と最も類似した2つの単語がA群の単語である場合，正解とする．この実験を30個の未知語に対して行った．これらの実験結果から，Webサーチエンジンを用いることの有効性を評価する．

1.2 本論文の構成

第2章では，名詞間類似度の測定について説明する．名詞間類似度が自然言語処理における役割について語義識別での利用，クラスタリングでの利用を元に述べる．また，未知語の問題についても述べる．

第3章ではWebサーチエンジンの利用方法について説明する．本研究で提案したon the flyの類似度の測定方法と，基底単語の選出について述べる．

第4章では，Webサーチエンジンを用いての基底単語作成，また，得られた基底単語を用いた単語クラスタリングを行い，その結果を比較した．さらに，得られた基底単語を用いた名詞間類似度の測定を行った．第5章ではそれぞれの実験に関する結果を示し，その結果に対する考察を行う．第6章では本研究に対するまとめを述べる．

第2章　名詞間類似度の測定

2.1　シソーラスを用いた方法

シソーラスの作成を最初に試みたのはロジェ（P.Roget）である．ロジェはまず語を次の6つのクラスに分類した．

1 abstract relation

2 space

3 matter

4 intellect

5 volition

6 emotion，religion and morality

そして，それらをさらに詳しく細分化するということを行い，結果として木構造の分類体系を作成した，分類された最小の語のグループは木構造の葉の部分に存在するという構造である．このようにロジェのシソーラスは上位語/下位語の体系というよりは，同義語関係に中心を置いたものであった．そしてその主たる目的は，人間が文章を書く場合に自分の考えを最も適切に表現する語を探すという作業の手助けをすることであった．

最近の自然言語処理においては語と語の間の類似度を計算する上でシソーラスが重要な役割を果たしている．これは，シソーラスの木構造において近くに存在する語同士は類似度が高いが，距離が離れるにしたがって類似度が小さくなると考えることができるからである．

シソーラスには，ロジェのシソーラスのように語の分類が中心であって，木構造の葉の部分だけに実際の語が存在するものと，誤の上位下位関係を中心において木構造の内部のノードにも語が対応するものの2種類がある．前者を分類シソーラス，後者を上位下位シソーラスと呼ぶ．

分類シソーラスの場合には二つの語の共通の上位ノードの深さによって語の近さを表すことができる．すなわち，同一の最小グループに分類された場合には最大の類似度，逆に木構造の上位で別のグループに分類された語には小さな類似度が与えられる．

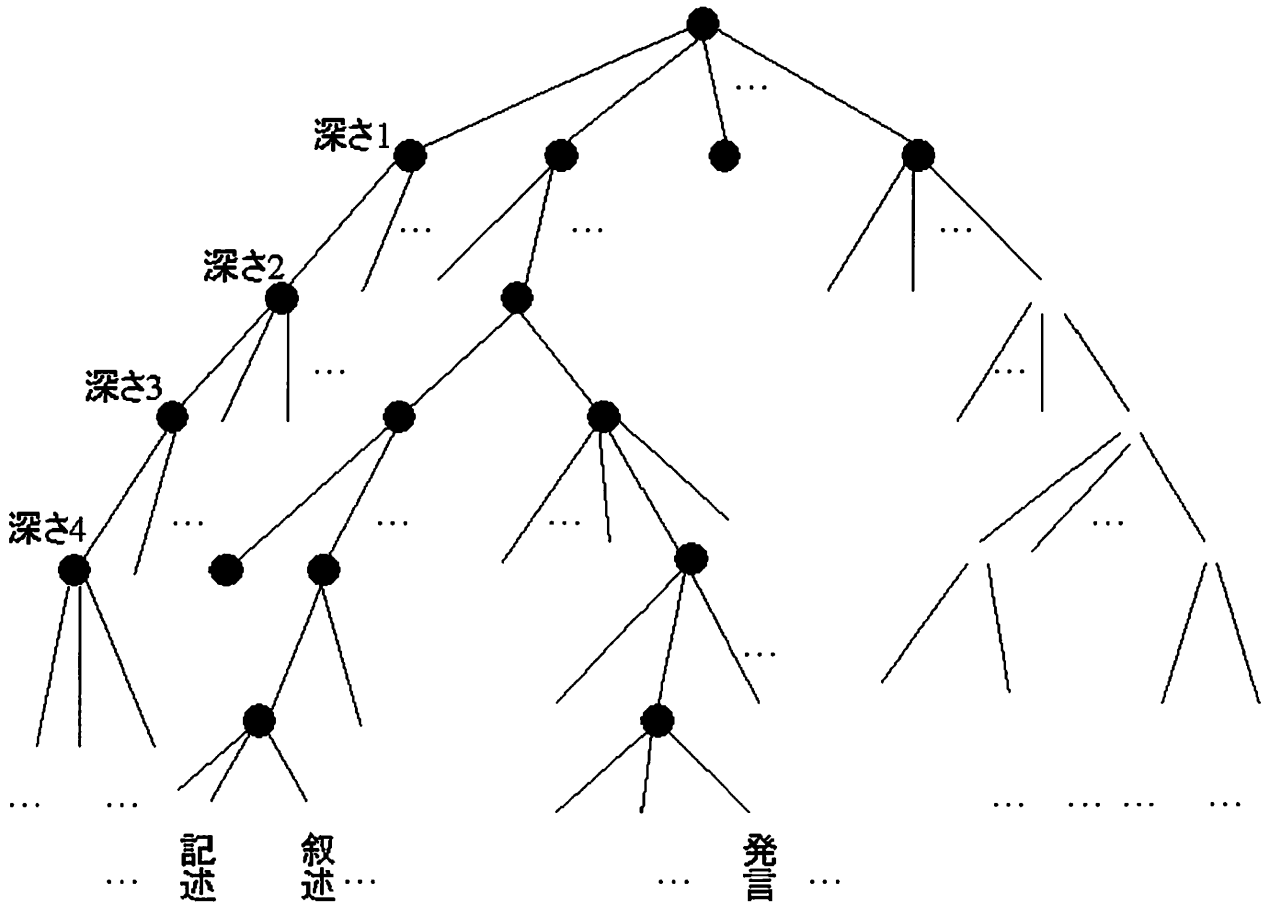


図 2.1: 分類シソーラス

記述と叙述，記述と発言の類似度を例として示す．図2.1では，記述と叙述では，初期ノードから深さ6まで共通の道をたどっている．よって記述と叙述の類似度は6である．また，記述と発言では，初期ノードから深さ3まで共通の道をたどっているため，これらの単語の類似度は3である．

一方，上位下位シソーラスの場合には，木構造の内部にも語が存在するので分類シソーラスと同一の方法では適当な類似度の尺度とはならない．二つの語の深さを d_i，d_j，それらの共通の上位語（ノード）の深さを d_c としたとき

$$\frac{d_c \times 2}{d_i + d_j}$$

という値を2語の類似度とする．この場合類似度の最大値を1とする．

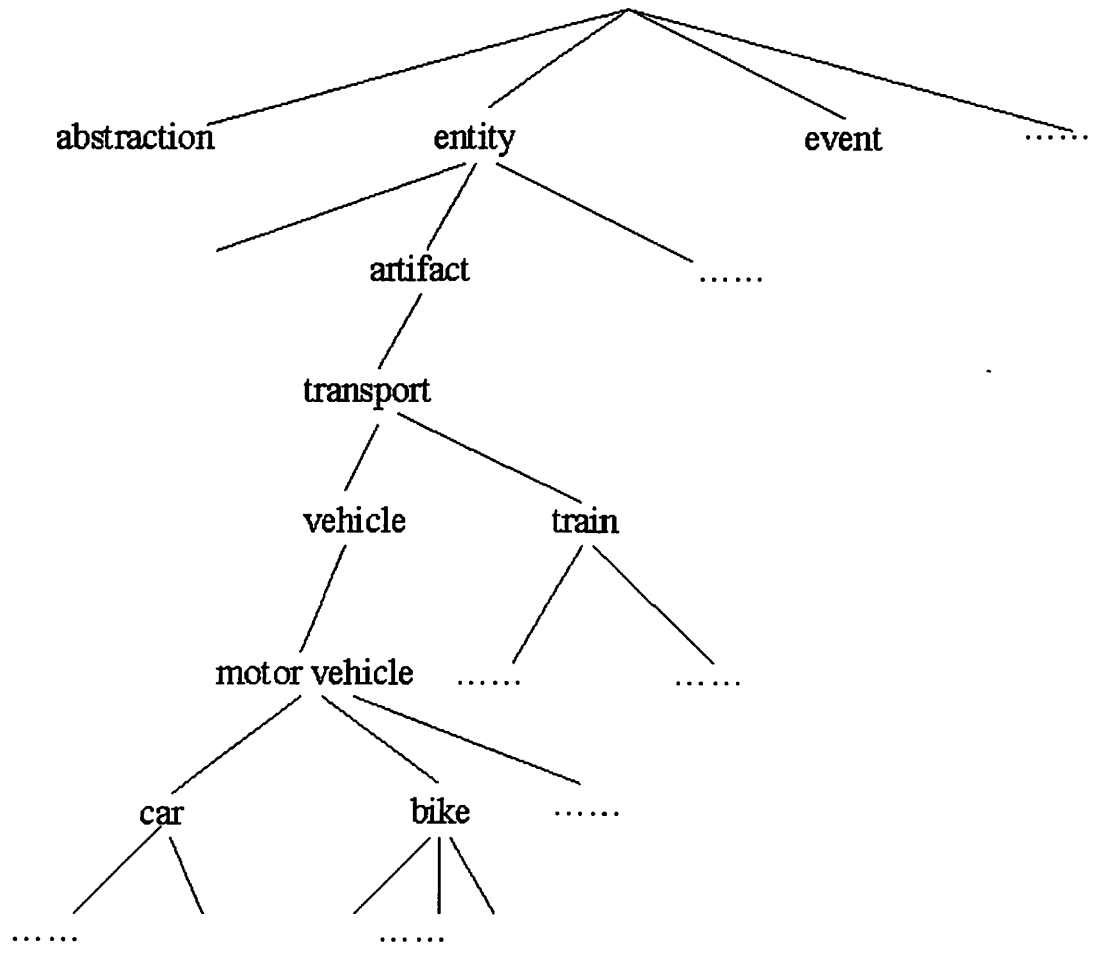


図 2.2: 上位下位シソーラス

car と bike，vehicle と train，car と train の類似度を例として示す．図2.2では car と bike はどちらも深さ6であり，共通の上位ノードの深さが5であるから $\frac{d_c \times 2}{d_i + d_j} = \frac{5 \times 2}{6+6} = 0.83$ となる．同様に vechile と train の場合は，どちらも深さ4であり，共通の上位ノードが3であることから $\frac{d_c \times 2}{d_i + d_j} = \frac{3 \times 2}{4+4} = 0.75$ となる．また，car と train の場合では，car の深さが6，train の深さが4であり，これらに共通する上位ノードは3であることから $\frac{d_c \times 2}{d_i + d_j} = \frac{3 \times 2}{6+4} = 0.60$ となる．

2.2 コーパスを用いた方法

コーパスを用いて名詞間類似度を測定する場合，テキストデータの種類や利用目的に応じて様々な技術が提案されているが，それぞれに共通する手順は次の通

りである.

step1 テキストデータから単語の特徴の重みを要素とするベクトルを作成する. これを特徴ベクトルと呼ぶ.

step2 特徴ベクトル中の特徴を互いに関連性の少ない特徴に変換した属性ベクトルを作成

step3 2つの単語に対する属性ベクトルを用いて単語間の類似度を計算

以下に1から3までの手順および, それぞれにおける既存方式の特徴を説明する.

2.2.1 特徴ベクトルの作成

テキストデータから注目する単語に対応する表層的な特徴を抽出し, 個々の特徴の重みを表す数値を要素とした単語の特徴ベクトルを作成する.

クラスタリングの対象である n 語を $w_1, w_2, \cdots, w_n$ で表す. 単語を表現する特徴数を m とし, 単語 w_i の特徴ベクトル w_i を以下のように表す.

$$w_i = (v_{i1}, v_{i2}, \cdots, v_{im})$$

$v_{ij}(j = 1, 2, \cdots, m)$ は j 番目の特徴と単語 w_i の関係の強さを表す数値で, 特徴の重みと呼ぶ. また, 特徴ベクトルの集合である特徴行列 G_o は以下のように表せる.

$$G_o = \begin{pmatrix} w_1 \\ \cdots \\ w_i \\ \cdots \\ w_n \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{im} \\ \cdots & \cdots & \cdots & \cdots \\ v_{i1} & v_{i2} & \cdots & v_{im} \\ \cdots & \cdots & \cdots & \cdots \\ v_{n1} & v_{n2} & \cdots & v_{nm} \end{pmatrix}$$

テキストより抽出する単語の特徴としては, 新聞記事などのテキストコーパスを用いる場合には, 主語や目的語に対する述語や名詞と連接する名詞, 注目する単語の近傍に存在する単語が提案されている. また, 単語の類似性を考慮した情報検索技術では, 検索対象の文書集合中で注目する単語を含む文書IDを特徴としている. これら特徴の出現傾向を数値化した重みを要素として特長ベクトルが生成される.

一方, 国語辞典を用いた技術では, 語義文中の単語を見出し語の特徴として用いている. 全ての語義文が同じ詳細さで語の説明記述していれば, 語義文中の特徴となる単語の出現頻度のみから特徴の重みを計算すれば十分である. しかし電子化辞書の語義文は, 人間の語の理解のために作成されているため, 1語のみの簡潔な記述しかない語義文や, 形式名詞や辞書に固有な言い回しなどの定義に直接

関わらない単語が含まれる語義文が存在している．そのため，ニューラルネットワークの考えを用いたアプローチや，人間の辞書を利用する際のヒューリスティクスを適用するアプローチのように，見出し語と別の見出し語の語義などの間接的な関連性も考慮して特徴の重みを適切に表現する必要がある．

2.2.2 属性ベクトルの作成

単語の特徴をもれなく記述したテキストデータから特徴ベクトルを作成することができるならば，2つの単語の特徴ベクトル中の特徴ごとの重みの比較によって類似性判別が可能である．しかし，テキストデータから抽出される特徴には常に欠落やノイズが存在するため，適切な判別を行うことができない．そのため，特徴間の関連性を考慮して主要で相互に関連性の少ない特徴に変換して，テキストデータ中の特徴のノイズや欠落の影響を減らした属性ベクトルの作成が行われる．

特徴ベクトル G_o の m の特徴を $k(\leq m)$ の互いに関連性の少ない特徴（「属性」と呼ぶ）に変換する．特徴の線形変換によって特徴行列を個々の単語の属性の重みを要素とする属性行列 G へ変換できると仮定すると，その変換は m 行 k 列の変換行列を掛け合わせることに等しい．

$$G = G_o K$$

また，K を特徴ベクトルに掛け合わせることで属性ベクトルに変換できる．

$$\acute{w}_i = w_i K = (\acute{v}_{i1}, \acute{v}_{i2}, \cdots, \acute{v}_{ik})$$

変換の方式としては主として，特徴行列の特異値分解や主成分分析による方式（数学モデル）と，大規模なシソーラスを利用し，特徴を表す語群をシソーラスの分類に一般化する方式（シソーラスモデル）がある．最近，これらの2つの変換方式を併用する方式（併用モデル）が提案されている．特徴行列中の特徴をシソーラスの分類に一般化した後，特異値分解を適用するモデルであり，同じ次元数の属性行列では元となる2つのモデルより類似性判別の精度が高く，200以上の高い次元数まで単調に判別精度が高くなる属性行列が作成できる点で優れていることが報告されている．

2.2.3 類似度計算

属性行列 G を用いて，語群 N 中の任意の2単語間の類似度を計算する．類似度とは，2つの単語の似ている度合いを表す尺度であり，値が大きなほどその単語同士が類似していることを表す．単語 W_p と $W_q(\in N)$ の類似度 $\mathrm{sim}(W_p, W_q)$ は，属性ベクトル w_p と w_q より計算される．代表的な類似度としては，2つの単語の属性ベクトルのなす角度の余弦が用いられている．

$$sim(W_p, W_q) = \frac{\acute{w}_p \acute{w}_q}{|\acute{w}_p||\acute{w}_q|}$$

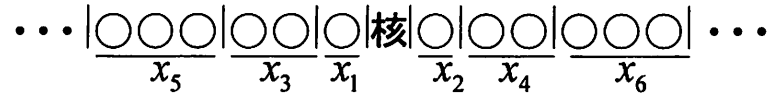
これ以外の類似度の尺度としては，属性ベクトルの中の重みの絶対値の比較に基づいて計算する方式や，重みを確率値とみなし確率分布間の距離を測るダイバージェンスより計算する方式などがある．

2.3 名詞間類似度の利用例

2.3.1 語義識別での利用

語義識別とは，文章中にある単語が複数の意味を持つ場合，どの意味をとるか正確に識別することである．人間は文脈上の意味から，どの語義となるかを瞬時に判断し，識別することができる．一方，機械が語義識別を行う場合，識別対象の単語における周辺の数単語を用い，名詞間類似度から識別する．語義識別における機械学習の手法は様々なものが報告されているが，例として Naive Bayes 法について説明する．

機械学習による学習規則を用いて語義識別を行なう場合，主に統計的手法を用いる．まず多義語の素性に着目する．ここでいう素性とは，識別対象の前後に存在する内容語を指す．多義語の前後にある素性を学習させ，文脈上にある素性が現われた場合，最も現われる確率の高い語義を識別結果として返す．



前後の内容語を素性とする

図 2.3: 素性の説明

ある事例 x が要素ベクトルとして以下の様に表せるとする．

$$x = (x_1, x_2, \cdots, x_m)$$

ここで x を構成する各要素は，多義語の前後にある素性を示している．

また x の分類先のクラスの集合を以下の様に表せるとする．

$$C = \{c_1, c_2, \cdots, c_k\}$$

ここで C を構成する各要素は，多義語の語義を示している．

文脈上にある要素ベクトル x が出現した場合，クラス c_i となる確率が最も高いものが識別結果となる．つまり語義識別を行なうには $P(c_i|x)$ を最大にする c_i を求めれば良い．ベイズの定理より，$P(c_i|x)$ は以下の様に表せる．

$$P(c_i|x) = \frac{P(c_i)P(x|c_i)}{P(x)}$$

ここで，$P(x)$ は不変であるので，$P(c_i|x)$ が最大となる確率を求めるには，$P(c_i)P(x|c_i)$ を最大にする c_i を求めれば良い．$P(c_i)$ は事後確率であるのでクラスの割合などから比較的容易に推測できる．しかし $P(x|c_i)$ は，クラス c_i が出現したとき，要素ベクトル x が現われる確率であるので推定が困難である．ここで以下のように仮定する．

$$P(x|c) \approx \prod_{i=1}^{m} P(x_i|c)$$

この仮定により，要素ベクトルの全てを一度に考慮せず，$P(x_i|c)$ の総積を考えることで結果として $P(x|c)$ が推定できる．これを Naive Bayes 法という．[3]

2.3.2 シソーラスの自動構築

名詞間類似度を用いて，名詞間類似度の値が高い順にクラスタリングを行い，シソーラスの自動構築を行う．なお，クラスタリングとは，対象の単語群をその類似度を基に複数のクラスに分類することである．以下に構築の手順を示す．

step1 1つずつを構成単位とする n 個のクラスタから出発する

step2 クラスタ間の類似度を参照し，もっとも類似度の高い2つのクラスタを融合し，1つのクラスタを作る

step3 クラスタ数が1になっていれば修了する．そうでなければ次のステップへ進む．

step4 ステップ2で新しく作られたクラスタと，他のクラスタとの類似度を計算し，ステップ2へ戻る

また，クラスタリング手法は大きく，最短長距離法などの階層的手法と，k-means 法などの分割的手法とに分けられるが，これらの基本的手法を説明する [4].

(1) 階層的手法

階層的手法では，対象間の類似度を指標にし，樹状の分類構造を構成することを目標とする手法である．分類構造を適当な箇所で切ることにより，任意の個数のクラスタを得ることができる．樹状の分類構造であるため，切断箇所所が根に近づくほど多数の構成単位のクラスタが含まれる．つまり階層的構造を持っている．今回の実験では，階層的手法として Ward 法 [5] を用いる．Ward 法の代表的な距離関数 $D(C_1, C_2)$ を表す．

$$D(C_1, C_2) = E(C_1 \bigcup C_2) - E(C_1) - E(C_2)$$

ただし，

$$E(C_i) = \sum_{x \in C_i} (D(x, c_i))^2$$

(2) 分割的手法

分割的手法は，分割の良さの評価関数を定め，その評価関数を最適にする分割を探索する．可能な分割の総数は単語数に対して指数的なので，実際は準最適解を求める．今回の実験では，分割的手法として k-means 法を用いる．代表的な k-means 法では，セントロイドをクラスタの代表点とし，

$$\sum_{i=1}^{k} \sum_{x \in C_i} (D(x, c_i))^2$$

の評価関数を最大化する．

2.4 未知語の問題

名詞間類似度を測定する場合，対象となる単語が訓練データ内に存在していなくてはならない．しかし，全ての単語を訓練データ内に存在させるということは不可能であり，また，新出単語は常に出現するため，未知語の問題が発生する．

未知語の問題の一つとして，語義識別による問題が挙げられる．人間が容易に識別ができる一方で，機械が識別困難な単語が存在する．その例として「開発」，「核」，「精神」，「乗る」，「生まれる」，「かかる」が挙げられている．それらの単語において，人間が正解する場合，単純に周辺単語にその語義を連想させる単語が出現するだけであることが多かった．機械がそのようなテスト文で不正解になるのは，その語義を連想させる単語が訓練データ中に存在しないためであった．つまりその単語が機械にとっては未知語であったためである．例として，文脈上に「核物質」という語が存在したとする．人間は，「物質」という単語から，核兵器という語義をとると連想できる．しかし，機械では訓練データ内に「物質」という

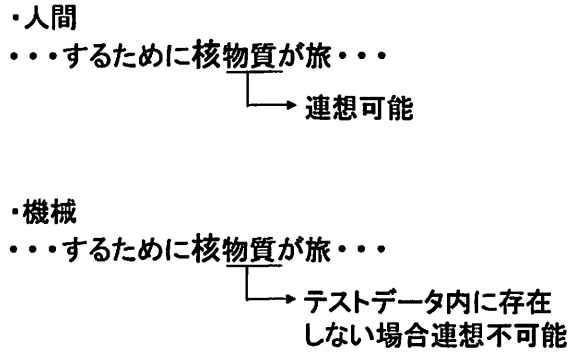


図 2.4: 語義の連想

単語がなければ語義の連想ができないということである．以下に「開発」,「核」,「精神」,「乗る」,「生まれる」,「かかる」の語義を示す．

表 2.1: 「うまれる」の語義

単語	語義
うまれる	1， 母体から子や卵が，その時期が来て，出る．また，卵からかえる．出生する．誕生する．「男子がー」 2， 今までなかったものが出来上がる．「新記録がー」

表 2.2: 「開発」の語義

単語	事例
開発	1， 開きおこすこと ア)， （天然資源などを）人間生活に役立たせること.「電源ー」 イ)， 現実化すること．実用化すること.「新製品のー」「研究ー」 2， 教育で，問答などを使って自発的にわからせる方法

表 2.3: 「精神」の語義

単語	事例
精神	1， 人間の心．非物質的・知的な働きをすると見た場合の心.「ー現象」「ー力」 2， 生命や宇宙の根源と考えられる存在. 3， 一般に，物事の根本の意義.「立法のー」

表 2.4: 「核」の語義

単語	事例
核	1， 物事の中心(となるもの)．かなめ.「ーになる」 2， 物の中心の部分.「地ー・痔ー（じかく）」 ア)， 細胞核.「ー膜・ー分裂」 イ)， 原子核．また，核兵器.「ーの持ち込み」 3， 草や木の芽生える種．内果皮の硬化したもの.「ー果」

表 2.5: 「のる」の語義

単語	事例
のる	1，　運送用の物の上や内部に移る.「馬にー」 2，　（持ち上げられて）物の上に移る. ア)，物に上がる.「台の上にー」 イ)，上に置かれる．載「机にーっている本」 3，　動き，調子によく合う ア)，勢いがついて物事が進む状態にある.「仕事に気がーらない」 イ)，他のものの調子にうまく合う.「リズムにーって踊る」 ウ)，十分によくつく.「あぶらのーった肉」 エ)，物事をする仲間・相手になる.「相談にー」 オ)，他からのたくらみにまんまと引き込まれる.「計略にー」 4，　伝える手段に託せられる.「電波にーって広まる」.特に，新聞・雑誌・書物に記される.「社会面にーった記事」

表 2.6: 「かかる」の語義 (1)

単語	事例
かかる	ア), (a), つるされる. ぶら下がる.「風鈴が軒にーっている」 (b), 繁留される. 停泊する.「船がー」 (c), 料理などのため火の上にすえられる.「なべがガスにーっている」 (d), 人目につくようにとめ揚げられる「額がーったお堂」 イ), 曲がった物, とがった物, 刃物, 張った物, 仕組んだ物にとらえられる. ひっかかる.「ホックがうまくーらない」「凧が電線にー」「わなにー」「催眠術にー」 ウ), (a), 物事がそれでとらえられる.「お目にー」「気にー」 (b), そこで扱われる.「医者にー」 (c), 持ち出されてそこにとまり, または, それで処理される.「重すぎてはかりにーらない」「計算機にーようなデータ」「裁判にかかる」 エ), (a), 倒れてしまわないように, それを頼みとする.「女にもたれー」 (b), 心をそれに寄せて頼みにする.「老後は次男にーつもりだ」 (c), 大切なものが代償となる.「優勝のーった一番」成功した時の賞として約束される.「敵将の首に百両ーった」 (d), 契約した状態にある.「この建物には保険がーっている」

表 2.7: 「かかる」の語義 (2)

単語	事例
かかる	ア), それをかぶった（浴びた）状態になる.「泥水がー」 それが他の物をおおう.「カバーのーった本」 イ), 他の行動の結果としてこちらに負担・不利が生ずる.「迷惑がー」 ウ), (a), 課される.「税がー」 (b), つぎ込む必要がある. 要る.「時間がー」 (c), 働き・力が（いっそう）加わる.「気合がー」「圧力がー」 (d), 強圧的な態度に出る. (e), 掛算をしてある結果となる.「その値には安全係数がーっている」 エ), 交配される.「スピッツにテリアがーっている」 オ), とりつかれる. 病気になる.「結核にー」

表 2.8: 「かかる」の語義 (3)

単語	事例
かかる	ア), (a), (両端を支えとし) またぐように渡される.「川に橋がー」 (b), 細長い物が他の物のまわりに渡される.「ひもがーった」 イ), (a), 声や言葉が送られる.「電話がー」 (b), 開いたり働いたりしないように, それが施される.「部屋にかぎがーっている」 (c), そこに作用が向かう.「誘いがー」「麻酔がー」 (d), 道具・機械が働きをする.「エンジンがー」 ウ), (a), 物事が関係する. それに関する.「国家機密にー重大事件」 (b), かけ言葉で表現される.「沼津食わずのヌマズには飲マズがーっている」 (c), その語句の文法的な働きが他の語句に向かう.「色よく咲くのイロヨクはサクにー」 エ), 動作・作用・状態がそれに (まで) 及ぶ.「坂道に (さし) ー」 オ), 張りめぐらしたり仕組んだりして, 作られる.「くもの巣のーった天井」 カ), (a), 攻撃をしに向かう.「素手でー」 (b), 仕事と取り組む (ために手を着ける).「工員五名が作業にー」

表 2.9: 「かかる」の語義 (4)

単語	事例
かかる	ア), ちょうど … する.「その時, 自動車が通りーった」 イ), … しそうになる.「死にー」

第3章　Webサーチエンジンの利用

3.1　単語のベクトル化

本研究ではWebサーチエンジンを利用し，名詞間類似度を測定することを提案する．名詞間類似度を用いる場合，対象単語と基底単語との共起頻度を用いる．基底の単語とは，共起頻度を得るための指標となる単語のことである．通常，基底の単語と特徴ベクトル化する単語との，コーパス内における共起頻度を用いて特徴ベクトルの作成を行うが，新聞記事等をコーパスに用いて特徴ベクトルを作成した場合，情報量が制限されるため，スパース性の問題が発生する．そこで，スパース性を解消するために，Webを利用する．Web文書の数は莫大であるため，未知語の問題が生じず，名詞間類似度を正しく測定できると考える．コーパスを用いた場合，単語e_jと1文中で共起するかどうかを考え，単語w_iのe_jに関する重みu_{ij}を，コーパス中でw_iとe_jの共起した文の数で測る．Webサーチエンジンを用いた場合，まず，Webに適した基底の単語群$e_1, e_2, \cdots, e_n$を選出し，対象の単語wと基底の単語からクエリ"a b_i "を作り，Webサーチエンジンを利用して単語wと単語e_iの共起したページ数f_iを得る．$(f_1, f_2, \cdots, f_n)$を正規化することで単語wの特徴ベクトルを作成する．単語をベクトルで表現できれば，任意の単語間の類似度を測定することができる．なお，Webサーチエンジンとして Google[1]を用いた．

3.2　基底単語

Webサーチエンジンを利用し，特徴ベクトルを得るにはまず，基底の単語群を選出する必要がある．適当な基底単語を選出できれば，適切な特徴ベクトルが得られ，名詞間類似度の精度が向上する．実験においては，論文[6]によって報告されている基底単語の候補となる922単語を利用した．ただしこの922単語は，新聞記事データをコーパスに用いて選出されたものである．しかしWebサーチエンジンから共起頻度を得るため，基底単語もWeb文書をコーパスに用いて選出した方が良いと考えられる．そこで，Webをコーパスとして用いてこの922単語の中から基底単語を選出する．

基底単語選出の際，単語間の類似度を測る必要がある．単語$w1$と$w2$の類似度は，$w1$の頻度を$f1$，$w2$の頻度を$f2$，$w1$と$w2$が共起した頻度を$f3$とするとき，

[1] http://www.google.co.jp/

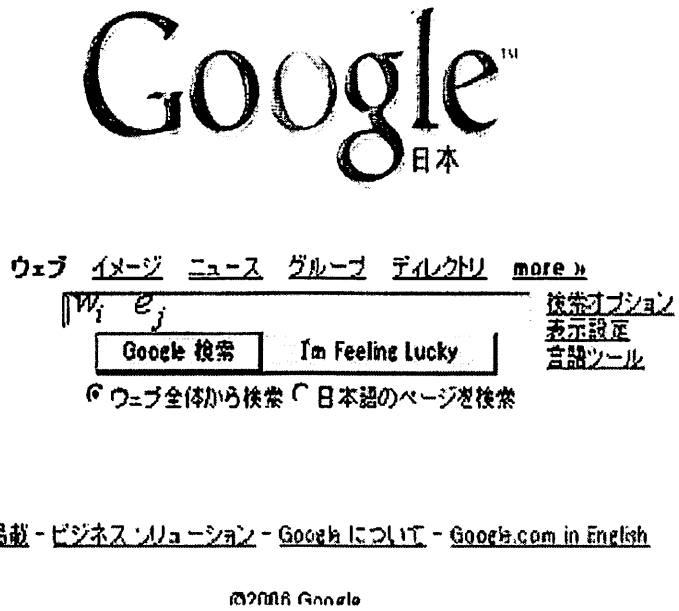


図 3.1: Web サーチエンジンを用いた共起頻度取得

次式で求められる.

$$sim(w1, w2) = \frac{2 \cdot f3}{f1 + f2}$$

これは Dice 係数と呼ばれている. ここで, 頻度は文書数である. つまり, $f1$ は $w1$ を含む文書の数である. また, $f3$ は $w1$ と $w2$ をともに含む文書の数である.

以上の方法を用いて, 単語クラスタリングを行い, 基底単語を選出する. 単語クラスタリングとは, 名詞間類似度から概念の似ている単語同士を一つのクラスタとしてまとめることである.

3.3 On the fly の類似度の測定

on the fly の類似語判定法を提案する [1]. 語義識別において, 未知語の名詞間類似度を測定する. 未知語が出現した段階で, 与えられた単語群の中からその未知語と最も類似の単語を選択する手法である.

未知語が出現した段階で, 未知語と単語群の各単語とが特徴ベクトルとして, 表現することができれば, 単純に距離を測るだけで問題の未知語と最も類似の単語を選択することができる.

問題はどのようにして, 未知語が出現した段階で, その未知語の特徴ベクトルを得るかである.

基底単語を選出し，得られた基底単語群 $(v_1, v_2, \cdots, v_{100})$ と対象の単語 w の特徴ベクトルを得るのに，"w v_i" をクエリにして Google[2] から検索を行なう．[1] そのヒット数を h_i とおく．今回，共起頻度として検索ヒット数を用いる．そして対象単語 w の特徴ベクトルを以下で表現する．

$$w = (\frac{h_1}{Z}, \frac{h_2}{Z}, \cdots, \frac{h_i}{Z}, \cdots, \frac{h_{100}}{Z})$$

ここで Z は w を正規化する定数である．

$$Z = \sqrt{\sum_{i=1}^{100} h_i^2}$$

on the fly の類似後判定で処理できない部分については Naive Bayes 法を用いる．その理由としては，Senseval2 の辞書タスクでは，Naive Bayes 法が最も良い成績を収めたと報告されているためである．[7]

名詞の語義識別ではその名詞が複合語になっていれば，その名詞の直前あるいは直後の単語によってほぼ語義識別が可能であることが知られている．そこでテスト事例の問題の多義語が複合語になっていた場合には，その複合語を訓練データから探し，もし存在すれば，対応する語義を識別結果とする．そしてもしも存在しなかった場合に on the fly の類似語判定を行なう．この方法は論文 [3] の複合語を優先して識別を行なう手法と基本的に同じである．

例えばテスト事例の問題の多義語 q が複合語になっており問題の多義語の直前の単語が w だとする．今，"wq" という複合語は訓練データ中に存在しない．しかし "x_1q", "x_2q", ..., "x_nq" という複合語は訓練データ中に存在するとする．ここで未知語を w，対象の単語群を $\{x_i\}$ として先に説明した on the fly の類似語判定を行なう．最も類似の単語が $\hat{x}$ であったとき，その "$\hat{x}q$" を含む訓練データに対する語義を識別結果として返す．

「開発」に対して on the fly の類似度の測定を用いた場合を例とする．「開発」のテストデータ (100 問) の中で「開発」が複合語の一部となっていたのは 71 問であった．このうち 21 問はその複合語が訓練データ中に存在した．この 21 問に対してはその複合語を含む訓練データの対応する語義が識別結果となる．「開発」のテストデータ (100 問) の中で「開発」が複合語の一部となっていたもので，その複合語を構成するもう一方の単語が未知語になっているものは，50 問である．この 50 問が on the fly の類似語判定の実験対象である．

on the fly の類似度の測定を用いたことで「開発」の正解率は 65% から 72% に向上した．人間の正解率が 76% なので，その差は小さくなっている．表 4 では on the fly の類似語判定を利用した識別の正解率が 50% であり，この部分の精度を向上させることができれば，人間の正解率と同等になることも期待できる．

[2] http://www.google.co.jp/

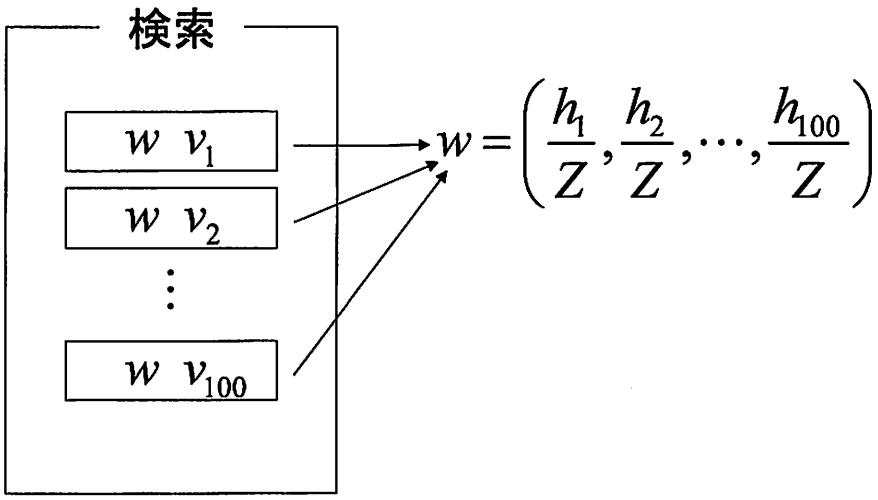


図 3.2: on the fly による特徴ベクトルの作成

第4章　評価実験

4.1　基底単語の選出

本実験で行った基底単語選出の流れを示す.

step1 Web 文書から本文のみを抽出

step2 本文中から文章のみを抽出. ここでは句読点を含む文を文章と定義する

step3 文章中から単語を抽出

step4 得られた単語をコーパスとして用いる

step5 922 単語と Web 文書から得られた単語との共起頻度を用いて類似度を測定する

step6 単語間の類似度より類似している単語をまとめる

step7 単語をまとめたクラスが 100 になれば修了. そうでなければ step6 へ

step8 類似している単語群から最も頻度の高い単語を基底単語として選出

以上を繰り返し, 基底の単語を 100 個得る. これは論文 [2] の結果を考慮してアドホックに選んだものである.

得られた 100 個の基底単語を表 4. 1 に示す. この 100 単語を用いて単語クラスタリング, 名詞間類似度の測定を行う.

4.2　クラスタリング

ここでは, Web をコーパスに用いた単語クラスタリングを行う.

単語クラスタリングとは, 名詞間類似度から概念の似ている単語同士を一つのクラスタとしてまとめることであり, 意味的に近い単語が一つのクラス内に存在している場合, 名詞間類似度が正しく測定されていると判断できる. また, この実験では, 基底単語についても評価する. 新聞記事データを利用して得られた基底の単語と, Web 文書を利用して得られた基底の単語とのクラスタリング結果を比較し評価する. また, 新聞記事から得られた基底の単語と, 新聞記事をコーパス

表 4.1: Web 文書から得た基底単語 (100 単語)

マウス	訪問	コレ	遺伝子	緊張
連載	外国人	パワー	納得	被害
フランス	買い物	時点	仲間	文章
音声	採用	比較	ドラマ	コーヒー
全員	要求	固定	流れ	結論
長期	魅力	週間	半年	事態
連続	本物	発表	申込	空気
年前	以降	スペース	マイ	オンライン
母親	兵庫	山本	公開	選択
政府	国内	記事	問題	毎日
青森	パソコン	ワン	大学院	変更
複数	地元	北海道	世界	了承
自動	制度	普段	身体	状況
総合	子ども	昨夜	定義	地球
就職	ノート	視点	時期	答え
共同	成長	話題	工事	様子
周り	最大	ルール	デザイン	努力
内部	松本	導入	放送	卒業
両方	未来	誕生日	スタート	保存
我が家	都道府県	限り	地図	特徴

に用いた場合のクラスタリング結果とも比較し，評価を行う．なお，クラスタリング手法として階層的手法であるward法と分割的手法であるk-means法を用いる．

実験で用いる単語は以下に示す25単語である．

動物 プードル，チワワ，犬，猿，ゴリラ

動物 カレー，ラーメン，スパゲッティー，焼きそば，ハンバーグ

感情・人生観 幸福，満足，愛情，結婚，運命

繁栄しているもの 情報，知識，手段，交通，設備

地名 今帰仁，沖縄，ブリスベン，オーストラリア，オーストリア

単語クラスタリングを行なった結果を図4.1，4.2に示す．

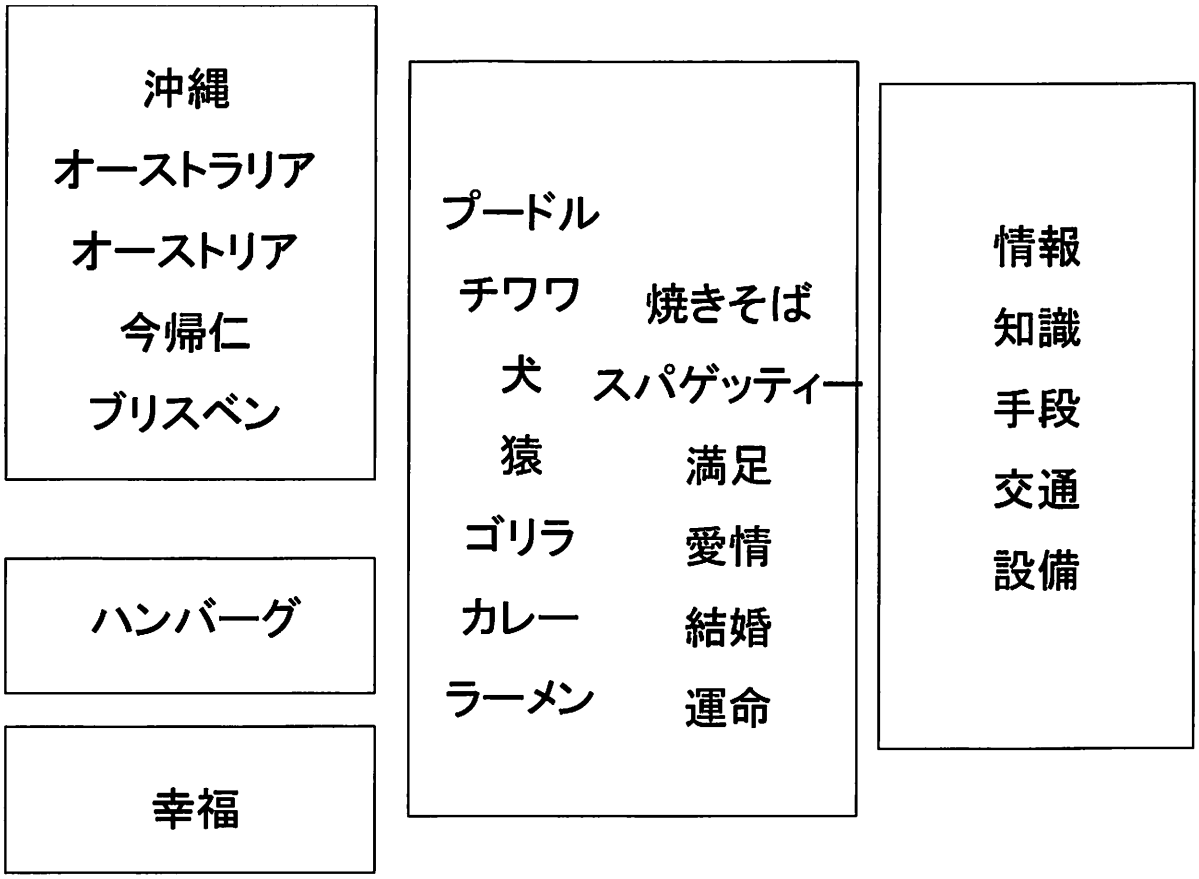


図 4.1: ward 法 (基底：Web, コーパス:Web)

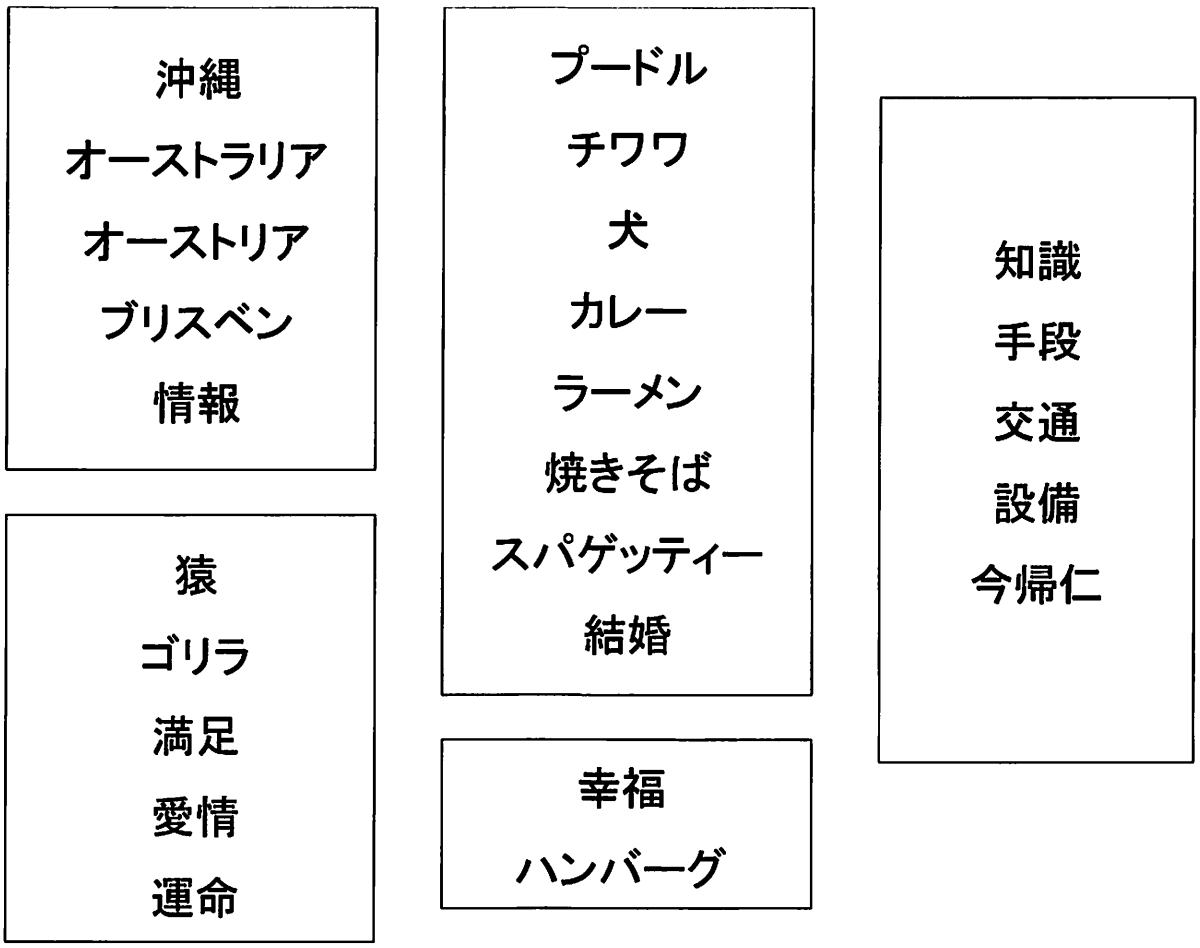


図 4.2: k-means 法 (基底：Web, コーパス:Web)

また，比較のために新聞記事から得た基底単語と，Web をコーパスに用いたクラスタリング結果を図 4.3，4.4 に示す．

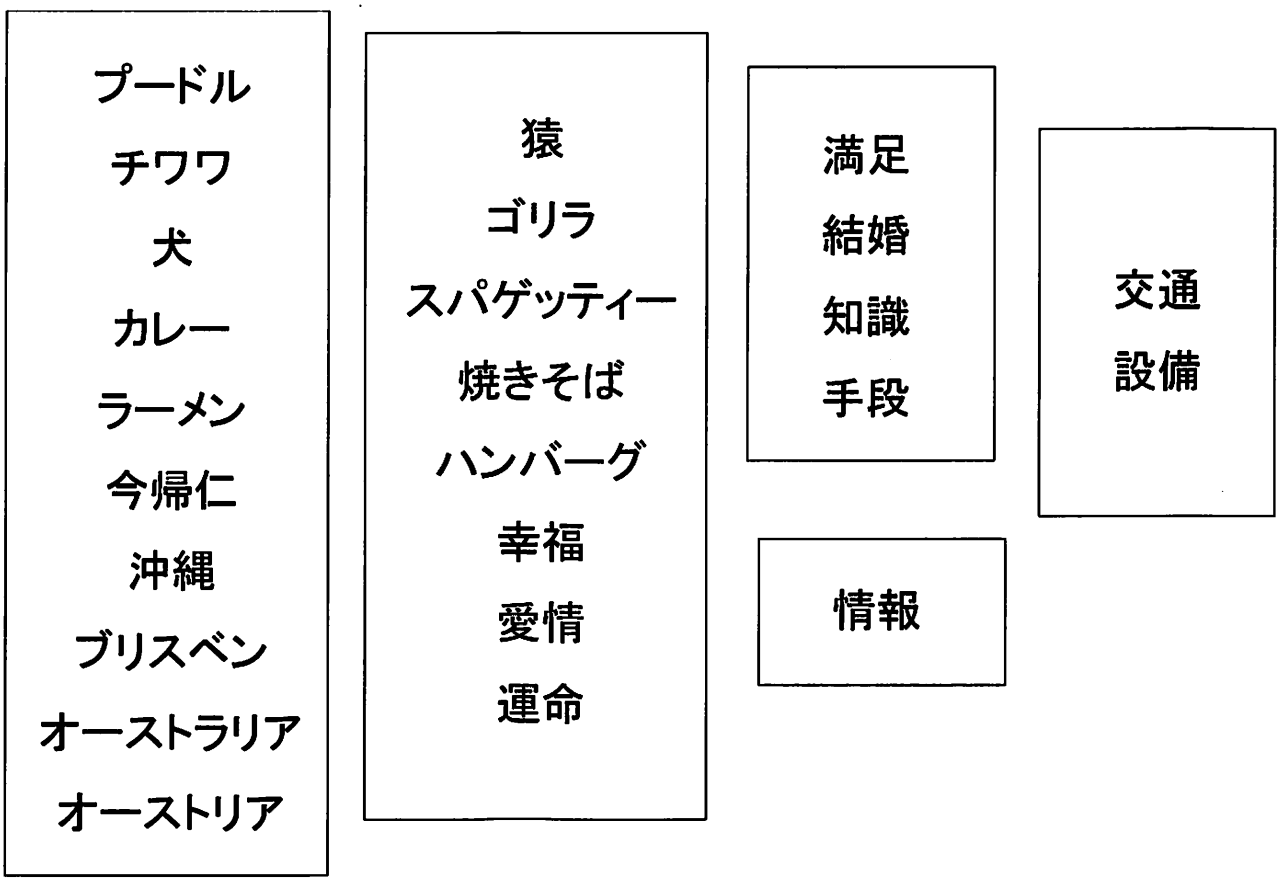


図 4.3: ward 法 (基底：新聞記事, コーパス:Web)

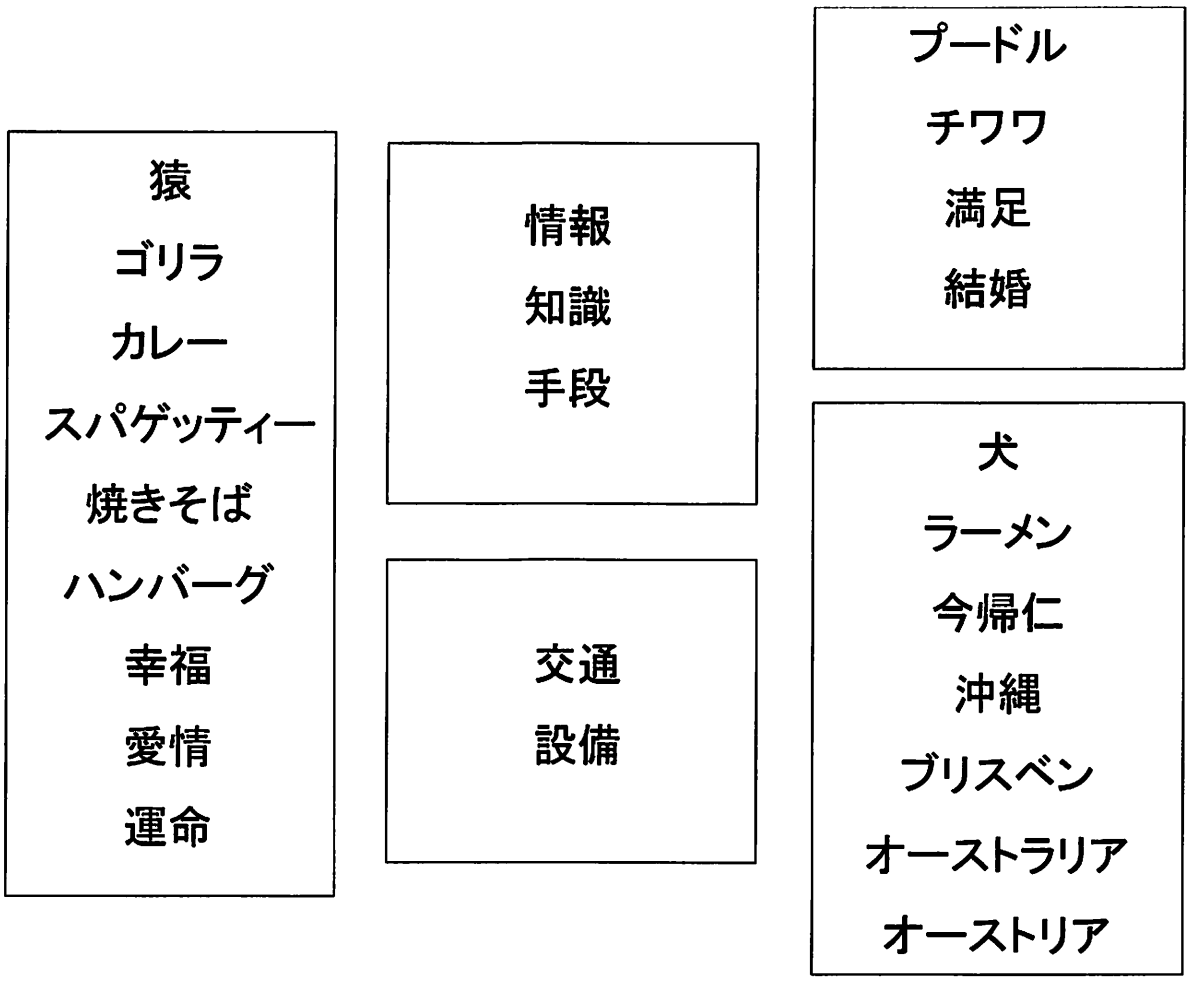


図 4.4: k-means 法 (基底：新聞記事, コーパス:Web)

Web 文書を利用して得られた基底単語と Web をコーパスにした場合のクラスタリング結果と，新聞記事データを利用して得られた基底単語と Web をコーパスにした場合のクラスタリング結果を比較した場合，Web を基底単語にした場合の方が良い実験結果が得られている．

同様に，新聞記事から得た基底単語と，新聞記事をコーパスに用いた場合のクラスタリング結果を図 4.5，4.6 に示す．ここでは毎日新聞記事 1 年分を使用している．

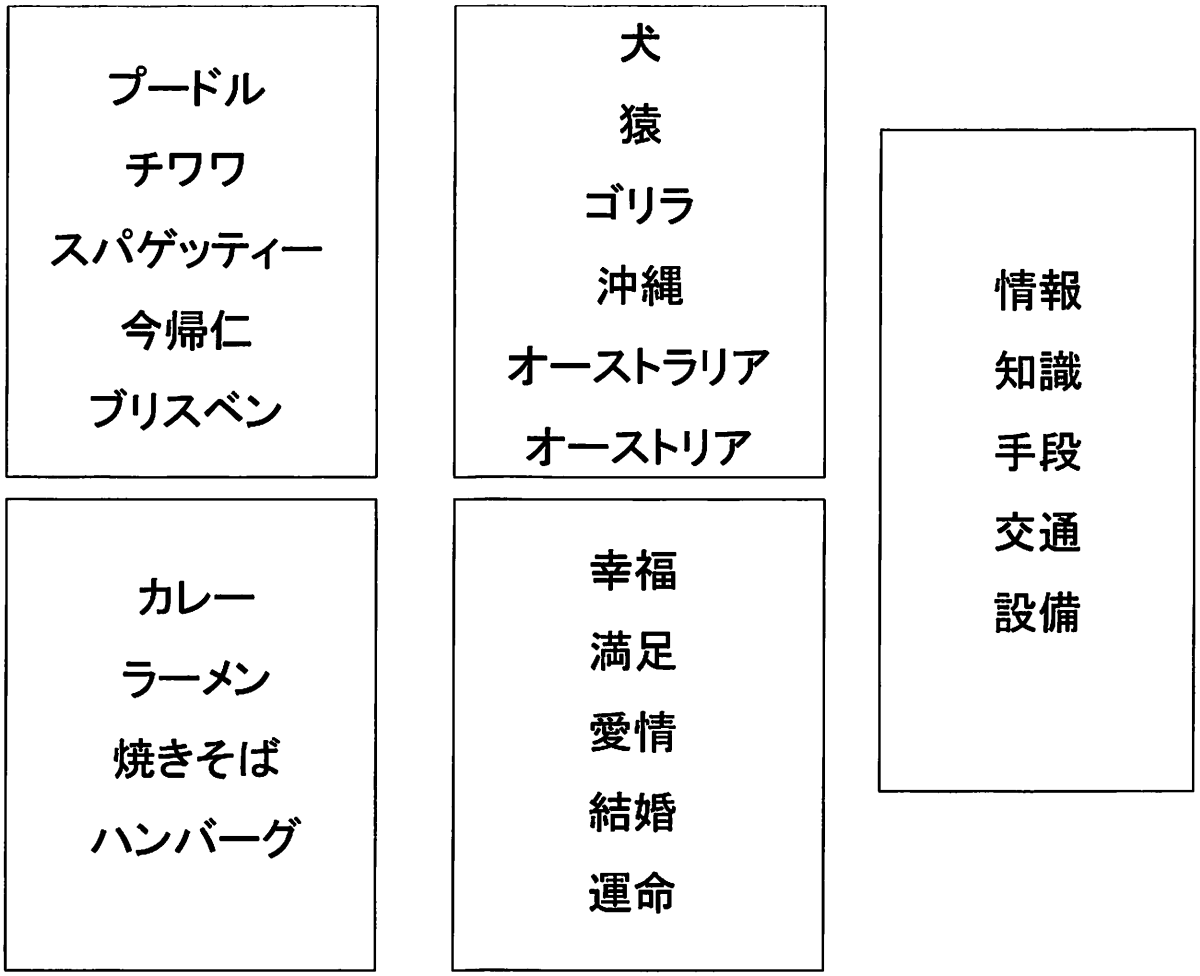


図 4.5: ward 法 (基底：新聞記事, コーパス:新聞記事)

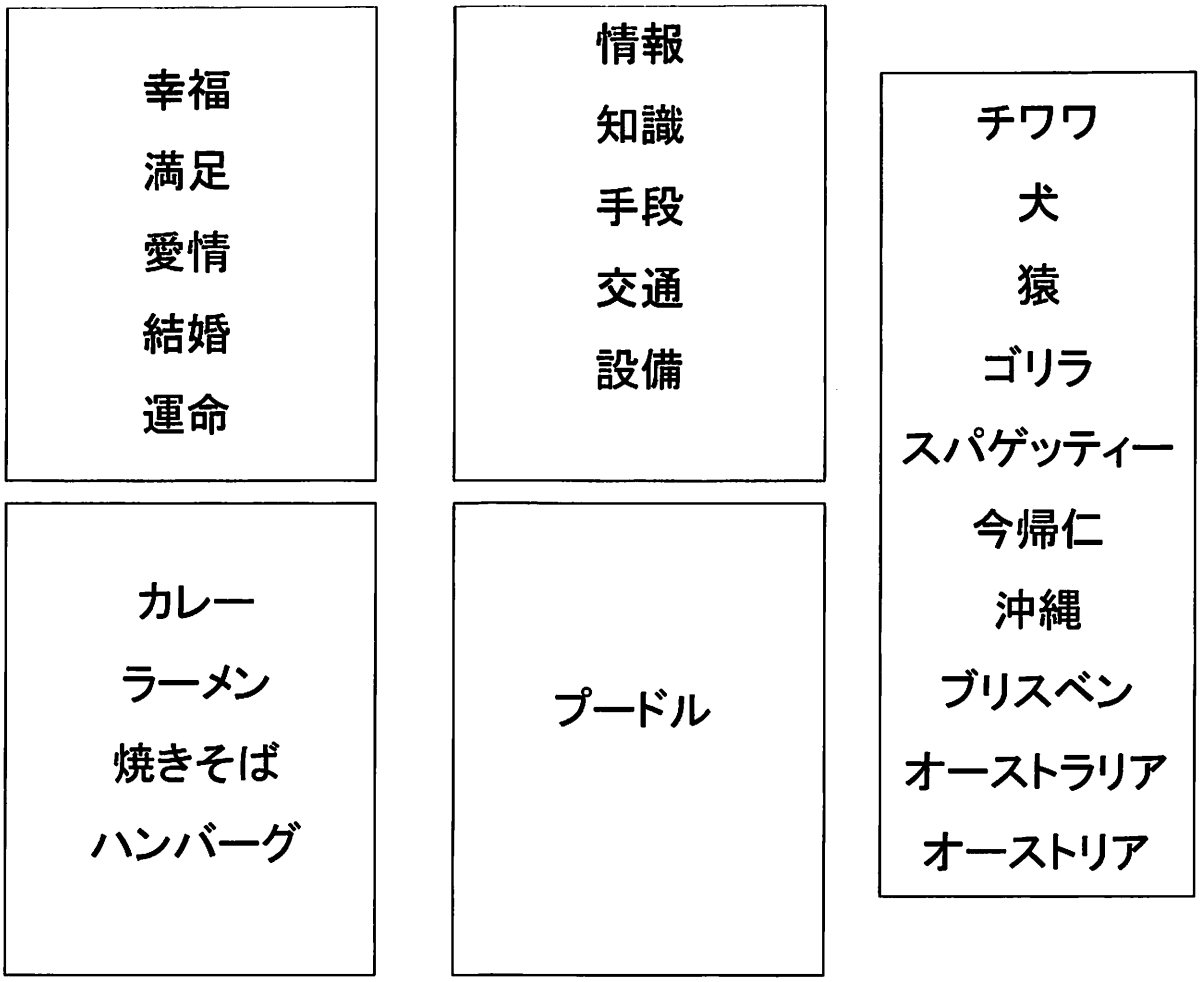


図 4.6: k-means 法 (基底：新聞記事, コーパス:新聞記事)

Web 文書を利用して得られた基底単語と Web をコーパスにした場合のクラスタリング結果と，新聞記事データを利用して得られた基底単語と新聞記事をコーパスにした場合のクラスタリング結果を比較した場合，新聞記事を基底単語にした場合の方が良い実験結果が得られている．

また，Web 文書を利用して得られた基底単語の数を 25 個，50 個とした場合のクラスタリングも行った．用いた基底単語をそれぞれ以下に示す．

表 4.2: Web 文書から得た基底単語 (50 単語)

最大	放送	保存	要求	ドラマ
全員	空気	採用	流れ	本物
半年	申込	週間	事態	長期
マイ	スペース	母親	公開	以降
青森	政府	松本	パソコン	世界
普段	共同	複数	ノート	北海道
了承	就職	内部	記事	地球
周り	限り	スタート	努力	誕生日
答え	導入	卒業	地図	我が家
特徴	都道府県	固定	未来	仲間

表 4.3: Web 文書から得た基底単語 (25 単語)

未来	保存	流れ	半年	採用
長期	マイ	申込	以降	公開
政府	世界	北海道	内部	卒業
答え	スタート	誕生日	都道府県	固定
地図	導入	努力	特徴	本物

クラスタリング結果を図 4.7，4.8 に示す．

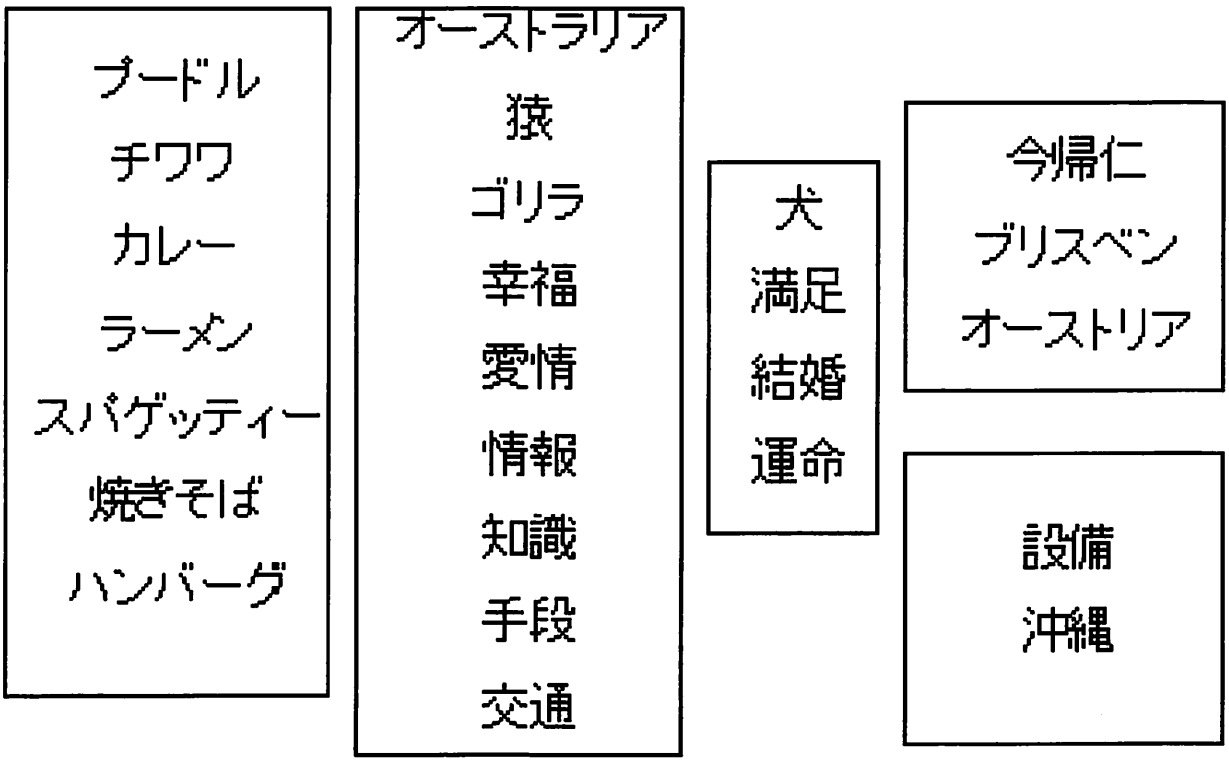


図 4.7: 50 個の基底単語によるクラスタリング

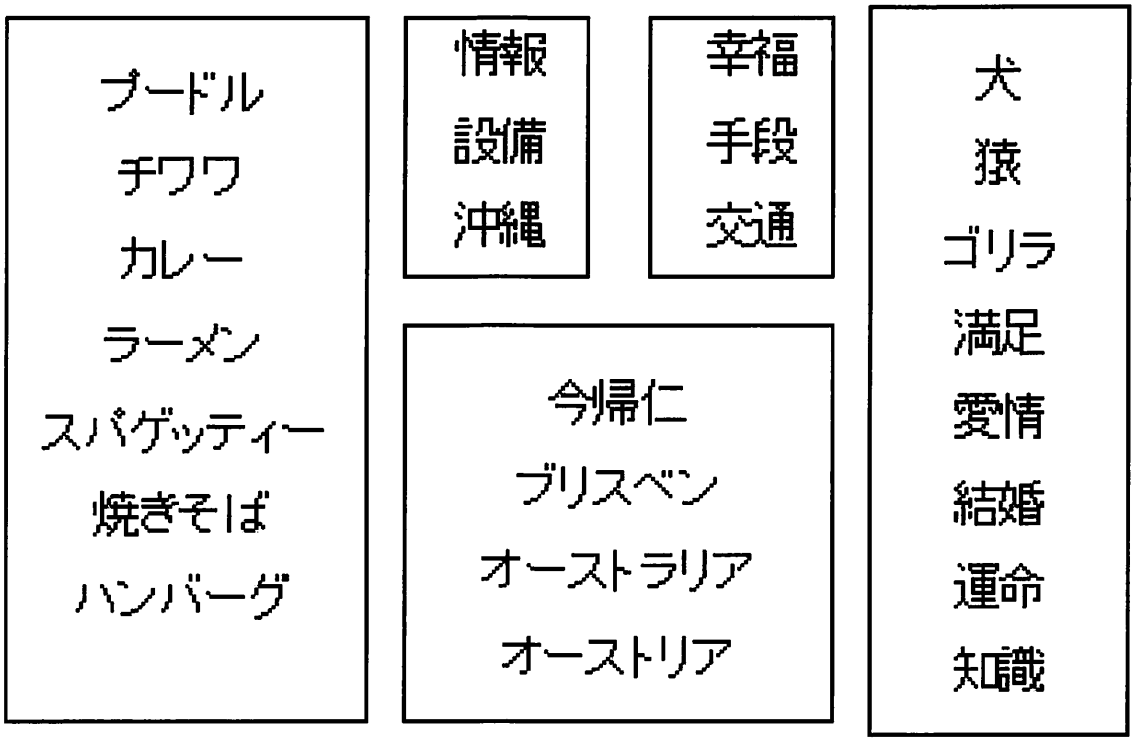


図 4.8: 25 個の基底単語によるクラスタリング

4.3 単語間類似度の測定

実験対象単語を示す．これらは，以下のキーワードランキングサイトの検索回数上位に存在する単語[1]を無作為に30単語選出した．

http : //guide.search.goo.ne.jp/ranking/

これら30単語と似ている意味の単語2個（A群）と似ているが少し意味が異なる単語2個（B群）を選出し，単語間の距離を測定し，距離の短い順に順位付けを行う．未知語と最も類似した2つの単語がA群の単語である場合，正解とする．実験対象単語を表4.4に示す．

表 4.4: 実験対象単語

三井住友銀行	干支	バレーボール
JR東日本	朝日新聞	NHK
オンラインゲーム	集英社	セブンイレブン
ダイエー	年賀状	浜崎あゆみ
チャングムの誓い	バイオハザード4	ドラゴンボール
シャープ	あいのうた	B'Z
仮面ライダー響鬼	電車男	スタッドレスタイヤ
リュ・シュウォン	F1	ハローワーク
ぐるなび	タウンページ	気象庁
ホテル	ロト6	ぷらら

また，これら30単語との名詞間類似度を測るための単語を1単語あたり4単語，計120単語選出した．これらを表4.5に示す．正解数は30個中12個であった．また基底単語数を100，50，25とした場合の結果をそれぞれ示す．基底単語の数が50個の場合，正解数は14個であった．また，基底単語の数が25個の場合，正解数は11個であった．

[1]2005年11月29日前後のものである。

表 4.5: 名詞間類似度の測定対象単語

A 群		B 群	
UFJ	みずほ銀行	野村證券	大和證券
鳥	犬	猫	象
バスケットボール	サッカー	剣道	柔道
JR 西日本	JR 東海	JAL	ANA
読売新聞	毎日新聞	スポーツ報知	日刊スポーツ
TBS	フジテレビ	J-WAVE	bayfm
ファイナルファンタジー	ハンゲーム	けん球	ベー駒
小学館	講談社	大日本印刷	凸版印刷
イオン	西友	ローソン	サンクス
パルコ	イトーヨーカ堂	セシール	ディノス
招待状	書中見舞い	宅急便	小包
中島美嘉	大塚愛	太田光	原田泰造
美しき日々	冬のソナタ	あずみ	座頭市
ドラゴンクエスト	スーパーマリオ	ムシキング	ビートマニア
NARUTO	ワンピース	ライオンキング	美女と野獣
パナソニック	ソニー	ヨドバシカメラ	ヤマダ電器
ゆず	WaT	ロバート	おぎやはぎ
ケロロ軍曹	トランスフォーマー	ギャバン	シャイダー
バカみたい	祖母の肖像	ジョーカー	夏のレプリカ
ワイパー	ホイール	ウェットスーツ	サーフボード
イ・ビョンホン	ペ・ヨンジュン	オダギリジョー	細川茂樹
フォーミュラニッポン	ラリー	モトクロス	オートレース
人材銀行	労働基準監督署	九州厚生局	東北厚生局
グルコン	グルメウォーカー	すぐくるデリバリー	楽天デリバリー
電話帳	ハローページ	郵便番号	住所コード
国土地理院	海上保安庁	人事院	会計検査院
民宿	旅館	クルーズ	ハネムーン
ナンバーズ	ミニロト	toto	パチスロ
OCN	So-net	JWord	Google
木更津キャッツアイ		ハリーポッターと炎のゴブレット	
花より男子		エンパイア・オブ・ザ・ウルフ	

表 4.6: 名詞間類似度の測定 (100 単語)

○	-	0.000000	三井住友銀行
	A	1.443229	UFJ
	A	1.905645	みずほ銀行
	B	2.802141	野村證券
	B	4.396776	大和證券
○	-	0.000000	干支
	A	3.050967	鳥
	A	3.096289	犬
	B	3.392598	猫
	B	4.920151	象
×	-	0.000000	バレーボール
	A	0.941269	バスケットボール
	B	1.221087	剣道
	B	1.759091	柔道
	A	3.597653	サッカー
○	-	0.000000	JR 東日本
	A	1.978020	JR 西日本
	A	2.054509	JR 東海
	B	2.609952	JAL
	B	2.635136	ANA
×	-	0.000000	朝日新聞
	A	2.123571	読売新聞
	B	2.582826	スポーツ報知
	B	3.483817	日刊スポーツ
	A	4.780574	毎日新聞
×	-	0.000000	NHK
	B	4.403133	J-WAVE
	A	4.988541	TBS
	A	5.933880	フジテレビ
	B	7.632924	bayfm

表 4.7: 名詞間類似度の測定 (100 単語)

○	-	0.000000	オンラインゲーム
	A	2.046349	ファイナルファンタジー
	A	2.609031	ハンゲーム
	B	6.052168	けん球
	B	6.794439	ベー駒
○	-	0.000000	集英社
	A	0.614595	小学館
	A	1.312777	講談社
	B	3.722078	大日本印刷
	B	4.180593	凸版印刷
×	-	0.000000	セブンイレブン
	B	3.424775	イオン
	A	4.161266	ローソン
	A	4.535917	サンクス
	B	4.561705	西友
○	-	0.000000	ダイエー
	A	1.772493	パルコ
	A	2.317663	イトーヨーカ堂
	B	5.014071	セシール
	B	5.839545	ディノス
×	-	0.000000	年賀状
	A	3.264431	招待状
	B	3.497167	宅急便
	B	5.090388	小包
	A	7.162703	書中見舞い
○	-	0.000000	浜崎あゆみ
	A	1.511387	中島美嘉
	A	2.426696	大塚愛
	B	4.253255	太田光
	B	5.697662	原田泰造

表 4.8: 名詞間類似度の測定 (100 単語)

○	-	0.000000	チャングムの誓い
	A	1.335406	美しき日々
	A	2.093909	冬のソナタ
	B	2.286873	あずみ
	B	2.724758	座頭市
○	-	0.000000	バイオハザード4
	A	1.073006	ドラゴンクエスト
	A	1.459082	スーパーマリオ
	B	2.113361	ムシキング
	B	2.754985	ビートマニア
○	-	0.000000	ドラゴンボール
	A	1.384124	NARUTO
	A	1.923826	ワンピース
	B	2.186933	ライオンキング
	B	1.940632	美女と野獣
×	-	0.000000	シャープ
	A	2.028459	パナソニック
	B	2.397417	ヨドバシカメラ
	A	3.480045	ソニー
	B	5.011425	ヤマダ電器
○	-	0.000000	あいのうた
	A	1.285980	花より男子
	A	1.729922	木更津キャッツアイ
	B	1.850264	ハリーポッターと炎のゴブレット
	B	5.153165	エンパイア・オブ・ザ・ウルフ
×	-	0.000000	B'Z
	A	1.365911	ゆず
	B	2.294246	ロバート
	A	2.617306	WaT
	B	2.758687	おぎやはぎ

表 4.9: 名詞間類似度の測定 (100 単語)

×	-	0.000000	仮面ライダー響鬼
	B	1.712052	ケロロ軍曹
	B	2.191134	トランスフォーマー
	A	3.950869	ギャバン
	A	4.446824	シャイダー
×	-	0.000000	電車男
	A	1.807085	バカみたい
	B	2.106922	ジョーカー
	B	4.711559	夏のレプリカ
	A	6.322792	祖母の肖像
×	-	0.000000	スタッドレスタイヤ
	B	2.139986	ウェットスーツ
	B	2.376830	サーフボード
	A	2.699664	ワイパー
	A	3.125534	ホイール
×	-	0.000000	リュ・シュウォン
	B	4.356299	オダギリジョー
	A	4.382287	イ・ビョンホン
	B	4.751799	細川茂樹
	A	4.780039	ペ・ヨンジュン
×	-	0.000000	F1
	A	3.127416	ラリー
	B	3.615495	モトクロス
	B	3.885195	オートレース
	A	5.233641	フォーミュラニッポン
×	-	0.000000	ハローワーク
	B	3.132316	九州厚生局
	B	3.203161	東北厚生局
	A	3.611333	労働基準監督署
	A	6.492718	人材銀行

表 4.10: 名詞間類似度の測定 (100 単語)

×	-	0.000000	ぐるなび
	A	4.101445	グルコン
	B	4.249700	すぐくるデリバリー
	A	4.269940	グルメウォーカー
	B	4.963781	楽天デリバリー
×	-	0.000000	タウンページ
	A	4.484693	電話帳
	B	4.891978	郵便番号
	B	5.466439	住所コード
	A	5.900284	ハローページ
○	-	0.000000	気象庁
	A	3.753015	国土地理院
	A	5.035044	海上保安庁
	B	5.339471	人事院
	B	5.622509	会計検査院
×	-	0.000000	ホテル
	A	5.645719	民宿
	B	5.767600	クルーズ
	B	6.372484	ハネムーン
	A	6.864294	旅館
×	-	0.000000	ロト 6
	A	1.129339	ナンバーズ
	B	1.931759	toto
	B	1.990779	パチスロ
	A	2.823674	ミニロト
×	-	0.000000	ぷらら
	A	3.403788	OCN
	B	3.414333	JWord
	A	3.781296	So-net
	B	4.806769	Google

表 4.11: 名詞間類似度の測定 (50 単語)

○	-	0.000000	三井住友銀行
	A	1.229728	UFJ
	A	1.658917	みずほ銀行
	B	2.409083	野村證券
	B	3.654577	大和證券
○	-	0.000000	干支
	A	2.103797	鳥
	A	2.227682	犬
	B	2.684472	猫
	B	3.894113	象
×	-	0.000000	バレーボール
	A	0.616168	バスケットボール
	B	0.849693	剣道
	B	1.570118	柔道
	A	1.934592	サッカー
×	-	0.000000	JR 東日本
	A	0.953318	JR 西日本
	B	1.764031	ANA
	B	1.804343	JAL
	A	1.984022	JR 東海
×	-	0.000000	朝日新聞
	A	2.007885	読売新聞
	B	2.065542	スポーツ報知
	A	2.849678	毎日新聞
	B	2.952551	日刊スポーツ
×	-	0.000000	NHK
	B	3.069409	J-WAVE
	A	3.209518	TBS
	A	3.931467	フジテレビ
	B	5.273891	bayfm

表 4.12: 名詞間類似度の測定 (50 単語)

○	-	0.000000	オンラインゲーム
	A	1.514735	ファイナルファンタジー
	A	2.135205	ハンゲーム
	B	4.327585	ベー駒
	B	4.484154	けん球
○	-	0.000000	集英社
	A	0.391155	小学館
	A	0.713283	講談社
	B	2.948984	凸版印刷
	B	2.959609	大日本印刷
○	-	0.000000	セブンイレブン
	A	1.430660	ローソン
	A	2.177967	サンクス
	B	2.371626	西友
	B	2.406526	イオン
○	-	0.000000	ダイエー
	A	1.258654	パルコ
	A	1.709909	イトーヨーカ堂
	B	3.141937	セシール
	B	4.345162	ディノス
×	-	0.000000	年賀状
	A	1.914851	招待状
	B	2.622611	宅急便
	B	3.580560	小包
	A	4.798586	書中見舞い
○	-	0.000000	浜崎あゆみ
	A	1.163036	中島美嘉
	A	2.147873	大塚愛
	B	2.865956	太田光
	B	3.833594	原田泰造

表 4.13: 名詞間類似度の測定 (50 単語)

○	-	0.000000	チャングムの誓い
	A	0.794894	美しき日々
	A	1.500589	冬のソナタ
	B	1.809520	あずみ
	B	2.100398	座頭市
○	-	0.000000	バイオハザード4
	A	0.833682	ドラゴンクエスト
	A	1.132978	スーパーマリオ
	B	1.435751	ムシキング
	B	2.001732	ビートマニア
×	-	0.000000	ドラゴンボール
	A	0.916069	NARUTO
	B	1.200604	美女と野獣
	A	1.281497	ワンピース
	B	1.365310	ライオンキング
×	-	0.000000	シャープ
	A	1.542517	パナソニック
	B	1.688419	ヨドバシカメラ
	A	2.696262	ソニー
	B	3.638531	ヤマダ電器
○	-	0.000000	あいのうた
	A	0.705783	花より男子
	A	1.250721	木更津キャッツアイ
	B	1.477595	ハリーポッターと炎のゴブレット
	B	3.955443	エンパイア・オブ・ザ・ウルフ
○	-	0.000000	B'Z
	A	0.873768	ゆず
	A	1.281187	WaT
	B	1.868820	ロバート
	B	1.994391	おぎやはぎ

表 4.14: 名詞間類似度の測定 (50 単語)

×	-	0.000000	仮面ライダー響鬼
	B	1.293369	ケロロ軍曹
	B	1.616044	トランスフォーマー
	A	2.580632	ギャバン
	A	3.049679	シャイダー
×	-	0.000000	電車男
	A	1.421727	バカみたい
	B	1.445226	ジョーカー
	B	3.299089	夏のレプリカ
	A	4.492425	祖母の肖像
×	-	0.000000	スタッドレスタイヤ
	B	1.728633	ウェットスーツ
	B	1.871854	サーフボード
	A	2.057671	ホイール
	A	2.261263	ワイパー
×	-	0.000000	リュ・シュウォン
	A	2.667730	イ・ビョンホン
	B	2.877346	オダギリジョー
	A	2.884606	ペ・ヨンジュン
	B	2.964211	細川茂樹
×	-	0.000000	F1
	A	2.412221	ラリー
	B	2.932208	モトクロス
	B	3.008030	オートレース
	A	3.696144	フォーミュラニッポン
×	-	0.000000	ハローワーク
	B	2.652504	九州厚生局
	B	2.664657	東北厚生局
	A	2.799212	労働基準監督署
	A	4.675253	人材銀行

表 4.15: 名詞間類似度の測定 (50 単語)

×	-	0.000000	ぐるなび
	B	3.230754	楽天デリバリー
	A	2.425307	グルメウォーカー
	A	2.444376	グルコン
	B	2.678579	すぐくるデリバリー
○	-	0.000000	タウンページ
	A	2.928749	電話帳
	A	3.508017	ハローページ
	B	4.057238	郵便番号
	B	4.714992	住所コード
○	-	0.000000	気象庁
	A	2.339543	国土地理院
	A	3.832960	海上保安庁
	B	4.325440	人事院
	B	4.352377	会計検査院
×	-	0.000000	ホテル
	A	3.845670	民宿
	B	4.080513	クルーズ
	B	4.585249	ハネムーン
	A	4.791737	旅館
×	-	0.000000	ロト 6
	A	0.872761	ナンバーズ
	B	1.537134	パチスロ
	B	1.613986	toto
	A	1.830044	ミニロト
○	-	0.000000	ぷらら
	A	2.476470	OCN
	A	2.630523	So-net
	B	2.772685	JWord
	B	3.047690	Google

表 4.16: 名詞間類似度の測定 (25 単語)

○	-	0.000000	三井住友銀行
	A	0.754896	UFJ
	A	1.127518	みずほ銀行
	B	1.422747	野村證券
	B	2.704673	大和證券
○	-	0.000000	干支
	A	1.372614	犬
	A	1.629553	鳥
	B	1.965547	猫
	B	2.640293	象
×	-	0.000000	バレーボール
	A	0.336419	バスケットボール
	B	0.551911	剣道
	B	1.291781	柔道
	A	1.335864	サッカー
×	-	0.000000	JR 東日本
	A	0.414844	JR 西日本
	B	1.330984	ANA
	B	1.366766	JAL
	A	1.596791	JR 東海
×	-	0.000000	朝日新聞
	B	1.188794	スポーツ報知
	B	1.554780	日刊スポーツ
	A	1.808377	毎日新聞
	A	1.843610	読売新聞
×	-	0.000000	NHK
	B	2.036536	J-WAVE
	A	2.452431	TBS
	A	2.689370	フジテレビ
	B	3.702546	bayfm

表 4.17: 名詞間類似度の測定 (25 単語)

○	-	0.000000	オンラインゲーム
	A	1.429725	ファイナルファンタジー
	A	1.556905	ハンゲーム
	B	3.294512	ベー駒
	B	3.696198	けん球
○	-	0.000000	集英社
	A	0.224430	小学館
	A	0.452422	講談社
	B	2.019647	凸版印刷
	B	2.331837	大日本印刷
×	-	0.000000	セブンイレブン
	A	0.969577	ローソン
	B	1.206120	西友
	A	1.383143	サンクス
	B	1.397555	イオン
○	-	0.000000	ダイエー
	A	0.893930	パルコ
	A	0.918326	イトーヨーカ堂
	B	2.155994	セシール
	B	3.386163	ディノス
×	-	0.000000	年賀状
	A	1.238239	招待状
	B	1.751378	宅急便
	B	2.561884	小包
	A	3.270968	書中見舞い
○	-	0.000000	浜崎あゆみ
	A	0.456362	大塚愛
	A	0.734937	中島美嘉
	B	2.029991	太田光
	B	2.819666	原田泰造

表 4.18: 名詞間類似度の測定 (25 単語)

×	-	0.000000	チャングムの誓い
	A	1.021629	美しき日々
	B	1.044921	座頭市
	A	1.103130	冬のソナタ
	B	1.258484	あずみ
○	-	0.000000	バイオハザード 4
	A	0.448663	ドラゴンクエスト
	A	0.715338	スーパーマリオ
	B	0.909418	ムシキング
	B	1.188341	ビートマニア
×	-	0.000000	ドラゴンボール
	A	0.604446	NARUTO
	B	0.781406	美女と野獣
	B	0.811894	ライオンキング
	A	0.855304	ワンピース
×	-	0.000000	シャープ
	A	0.819878	パナソニック
	B	1.421839	ヨドバシカメラ
	A	1.797924	ソニー
	B	2.474017	ヤマダ電器
○	-	0.000000	あいのうた
	A	0.600052	花より男子
	A	1.205072	木更津キャッツアイ
	B	1.371768	ハリーポッターと炎のゴブレット
	B	2.775976	エンパイア・オブ・ザ・ウルフ
○	-	0.000000	B'Z
	A	0.582473	ゆず
	A	0.903439	WaT
	B	1.585550	ロバート
	B	1.667075	おぎやはぎ

表 4.19: 名詞間類似度の測定 (25 単語)

×	-	0.000000	仮面ライダー響鬼
	B	0.698455	ケロロ軍曹
	B	0.811054	トランスフォーマー
	A	1.837499	ギャバン
	A	2.241770	シャイダー
×	-	0.000000	電車男
	B	0.778211	ジョーカー
	A	0.794835	バカみたい
	B	1.811998	夏のレプリカ
	A	3.608931	祖母の肖像
×	-	0.000000	スタッドレスタイヤ
	B	1.009789	サーフボード
	B	1.042043	ウェットスーツ
	A	1.690266	ホイール
	A	1.811485	ワイパー
×	-	0.000000	リュ・シュウォン
	B	2.154798	オダギリジョー
	A	2.180561	イ・ビョンホン
	B	2.281013	細川茂樹
	A	2.379928	ペ・ヨンジュン
×	-	0.000000	F1
	A	1.610007	ラリー
	B	1.917878	オートレース
	B	1.994162	モトクロス
	A	3.186440	フォーミュラニッポン
×	-	0.000000	ハローワーク
	B	1.473448	九州厚生局
	B	1.473833	東北厚生局
	A	1.952557	労働基準監督署
	A	3.189496	人材銀行

表 4.20: 名詞間類似度の測定 (25 単語)

×	-	0.000000	ぐるなび
	A	1.314608	グルメウォーカー
	B	1.620449	楽天デリバリー
	A	1.645472	グルコン
	B	1.842401	すぐくるデリバリー
○	-	0.000000	タウンページ
	A	1.648766	電話帳
	A	1.972608	ハローページ
	B	3.221638	郵便番号
	B	3.690966	住所コード
○	-	0.000000	気象庁
	A	1.278204	国土地理院
	A	2.768942	海上保安庁
	B	3.335158	人事院
	B	3.434110	会計検査院
×	-	0.000000	ホテル
	A	3.027109	民宿
	B	3.086491	クルーズ
	B	3.608627	ハネムーン
	A	3.712037	旅館
×	-	0.000000	ロト 6
	A	0.506233	ナンバーズ
	B	1.048019	パチスロ
	A	1.116831	ミニロト
	B	1.222747	toto
×	-	0.000000	ぷらら
	B	1.427338	JWord
	A	1.553525	OCN
	B	1.953271	Google
	A	2.025762	So-net

第5章　考察

5.1　クラスタリング

新聞記事から得た基底単語とWebをコーパスにした場合のクラスタリング結果をみると，ward法で行ったクラスタリングでは，適切にクラスタリングされているクラスはなく,「地名」の概念は同じクラスに分類されてはいるが，同じクラス内に「動物」の概念の単語と「食べ物」の概念の単語とが入り混じっている．また，k-means法で行ったクラスタリング結果を見ても，ward法と同様に他の概念の単語が入り混じっている．しかし，Webから得た基底単語とWebをコーパスに用いた場合のクラスタリング結果をみると，ward法で行ったクラスタリング結果では,「動物」,「食べ物」,「感情・人生観」のクラスの単語が入り混じっているが,「地域」,「繁栄しているもの」のクラスは適切にクラスタリングされている．また，k-means法で行った場合は，適切なクラスがなく，他の概念の単語が入り混じっている結果になった．よって，新聞記事から基底単語を得るよりも，Webから基底単語を得る方が良いと考えられる．新聞記事から得た基底単語と新聞記事をコーパスに用いた場合，プードル，チワワ，スパゲッティー，今帰仁，ブリスベンの単語にはスパース性の影響があるため，適切にクラスタリングされないことが報告されている[2]．しかし今回の実験では，ward法では今帰仁，ブリスベンを含む「地名」の概念が適切にクラスタリングされており，プードル，チワワ，犬という関連性の深い単語が同クラスにクラスタリングされている．k-means法の場合でも同様に，プードル，チワワ，犬は同クラスにクラスタリングされている．このことから，スパース性の影響は解消されたと考えられる．

しかし，クラスタリング結果は新聞記事から得た基底単語と新聞記事をコーパスに用いた場合の方が良い．これは，Webから得た基底単語に問題があると考える．今回の実験では，あらかじめ選出されている922単語からWebを利用して100単語の基底単語を選出した．しかし，この922単語は新聞記事を利用して選出されたものである．よって，この922単語をWebを利用して選出したとき，正しいクラスタリング結果に近づくと考える．また，今回基底単語を得るために用いたWebデータの量にも問題があると考える．今回用いたデータは，Webデータの一部のみを用いた．データ量を増やし，基底単語を修正することにより，より正確な名詞間類似度が測定できると考える．

基底単語数を100個から50個，25個と変化させた場合，クラスタリング結果が異なった．しかし，どちらの場合もプードル，チワワが同クラスにクラスタリン

グされている．同様に今帰仁，ブリスベンも同クラスにクラスタリングされており，スパゲッティーも食べ物の概念とクラスタリングされている．よって，スパース性の影響は解消されたと考えられる．また，クラスタリング結果が異なったことから名詞間類似度には基底単語の個数も重要であると考える．つまりより最適な特徴数を選べば，名詞間類似度が正しく測定できると考える．

5.2 名詞間類似度の測定

30個中正解数は12個となった．ランダムに並べた場合に正解する確率は1/6なので，本手法は未知語に関する名詞間類似度の測定において効果があることがわかる．また，基底単語数を50個，25個と変更した場合も正解率が14個，11個となっていることからランダムに並べた場合よりも正解率が高くなっている．単語クラスタリング同様，基底単語数を変更した場合に結果が異なったことから，名詞間類似度の測定では基底単語の個数が重要であると考える．

基底単語数を変更した場合に，結果が異なった単語に注目する．その単語とは「JR東日本」,「セブンイレブン」,「チャングムの誓い」,「ドラゴンボール」,「B'Z」,「タウンページ」,「ぷらら」の7単語である．これらの単語とその名詞間類似度を測定した単語を調べることにより，正解率が向上がすると考える．基底単語数の変化により結果が異なった原因として，ある単語が複数の意味を持っているためであると考える．例として,「セブンイレブン」との類似度を測る対象の単語として「イオン」があるが，これは店名と電子イオンの2通りの意味がある．そのため，基底単語の変化によって結果が異なったと考える．また，正解したデータを見ると，A群の距離とB群の距離に大きな差が見られる．よって、これらの距離の関連性も影響があると考える。

第6章　結論

本論文では，Webサーチエンジン用いた名詞間類似度の測定方法を提案した．評価方法として，単語クラスタリングと名詞間類似度の測定を行った．単語クラスタリングでは，Webを利用した基底単語が有効であるということが証明された．しかし，最適なクラスタリング結果にはならなかったため，改善の余地がある．また，名詞間類似度の測定では，Webサーチエンジンを用いた場合の正解率が向上したため本手法は名詞間類似度の測定において有効であると考える．以上の実験の結果，本手法は未知語に対して有効であると言えるが，一般の単語については有効であるとはいえない．これは，基底の単語選出の際に用いた単語が新聞記事を利用して得られたためであると考える．また，Webを利用して基底単語を選出する際に用いたデータも，Web文書全体ではなく一部のデータであったことも問題であると考える．また，今回，基底の単語数を100単語として実験を行った．単語クラスタリング，未知語との類似度測定の実験において，基底単語の数を変化させた場合，正解率に変化が生じた．このことから，名詞間類似度は基底単語数によって結果が異なると考えられる．そこで，最適な基底単語及び基底単語数を選出することができれば，さらに精度が向上できると考える．基底単語の最適な作成方法及び基底単語数を検討することが今後の課題である．

第7章　謝辞

本研究の遂行及び論文の作成において多大な御助言及び御指導を賜わった 新納 浩幸 教官（茨城大学工学部システム工学科）に深い感謝の意を表します．また，本研究で利用した文書データは，毎日新聞 CD-ROM'95 版です．利用を許可していただいた毎日新聞社に深く感謝します．最期に，本研究を進めるにあたり助言，協力を頂きました， 岩崎 唯史 教官（茨城大学工学部システム工学科），同研究室の 谷津 哲平 氏（茨城大学大学院修士課程），正木 裕一 氏（茨城大学大学院修士課程）及び，岩崎研究室の 児玉 光太郎 氏（茨城大学工学部システム工学科 4 回生），高橋 辰弘 氏（茨城大学工学部システム工学科 4 回生）に深く感謝致します．

参考文献

[1] 藤井丈明，新納浩幸，佐々木稔："語義識別の誤り原因の調査とオンザフライの類似語判定"，言語処理学会第10回年次大会,pp753-756，(2004).

[2] 大城亜里沙，新納浩幸，佐々木稔："検索エンジンを利用した単語クラスタリング"，言語処理学会第10回年次大会,pp17-20，(2004).

[3] 新納浩幸："複合語からの証拠に重みをつけた決定リストによる同音異義語判別"，情報処理学会論文誌, Vol.39, No.12, pp.3200-3206 (1998).

[4] 神嶌敏弘："データマイニング分野のクラスタリング手法(1)ークラスタリングを使ってみよう!ー"，人工知能学会誌，Vol18,No.1，pp.59-65, (2003).

[5] A.K.Jain,M.N.Murty,and P.J.Flynn. Data Clustering: A Review. ACM Computing Surueys, Vol.31, No.3, 1999.

[6] Minoru Sasaki and Hiroyuki Shinnou. Automatic thesaurus construction using word clustering., In PACLING-2003,pp.55-62,2003.

[7] 白井清昭："SENSEVAL-2 日本語辞書タスク"，自然言語処理, Vol.10, No.3, pp.3-24 (2003).

付 録A　プログラムソースリスト

```
//WEBデータからタグをとりのぞく

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define LINESIZE 256
#define WORDSIZE 100

void quit(char *);

int main(int argc, char *argv[])
{
FILE *f1;
char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE],str[LINESIZE],
int r,counter,flag,i,j,k,l,flagh,flagh2;
   counter=flagh=flagh2=l=0;
strcpy(str,"");
i=j=l=0;
if(argc != 2)quit("引数が違う prog datafile");
if((f1 = fopen(argv[1],"r")) == NULL)quit("ファイルが開けない");

  //処理開始

while(fgets(buf,LINESIZE,f1) != NULL){

i=j=k=l=0;
        strcpy(hantei,"");
        strcpy(copy,"");
        strcpy(copy2,"");
```

```
        flagh2=0;
while(buf[i]!='\0'){
//JAVA スクリプトの部分を削除
////////////////////////////////////////////////////////////////////////
hantei[k]=buf[i];
if(k>0)copy[k-1]=hantei[k];

         if(k==7)
          {
         hantei[8]='\0';
copy[7]='\0';

     if(strcmp(hantei,"<script>")==0)
{
flagh=1;


}
if(strcmp(hantei,"</script")==0)
{
flagh=0;



}
 strcpy(copy2,"");
strcpy(copy2,copy);
strcpy(hantei,"");
strcpy(hantei,copy);
strcpy(copy,"");
for(l=1;l<7;l++)copy[l-1]=copy2[l];
        copy[7]='\0';

 k=6;
                }


////////////////////////////////////////////////////////////////////////
```

```
//<…>でくくられている部分を削除
////////////////////////////////////////////////////////////////////////////
if(buf[i]=='<')flag=1;
if(flag==0 && flagh==0 )
{

 str[j]=buf[i];
 j++;


}
if(buf[i]=='>')flag=0;

          i++;
 k++;

////////////////////////////////////////////////////////////////////////////
}

str[j]='\0';



          printf("%s\n",str);


}

if((r = fclose(f1))== -1)quit("ファイルが閉じれない");

}

void quit(char *s)
{
puts(s); exit(1);
```

```
//WEB データから文章のみを抽出

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define LINESIZE 300
#define WORDSIZE 100
#define HANTEISIZE 20

void quit(char *);

int main(int argc, char *argv[])
{
FILE *f1;
char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE],str[LINESIZE],

int r,counter,flag,i,j,k,l,m,n,flagh,flagh2,flagh3;
   counter=flagh=flagh2=flagh3=l=0;
strcpy(str,"");
i=j=l=m=n=0;
if(argc != 2)quit("引数が違う prog datafile");
if((f1 = fopen(argv[1],"r")) == NULL)quit("ファイルが開けない");

  //処理開始

while(fgets(buf,LINESIZE,f1) != NULL){

i=j=k=l=m=n=0;
        strcpy(hantei,"");
        strcpy(copy,"");
        strcpy(copy2,"");
        strcpy(hantei2,"");
        strcpy(hozon,"");
        strcpy(hozon2,"");
        strcpy(url,"http://");
        flagh2=0;
if((strstr(buf,". ")!=NULL) || (strstr(buf,", ")!=NULL)) flagh2=1;
//strstr:buf 内に', 'か'. 'があったら値をかえす
```

```
while(buf[i]!='\0'){

//<…>でくくられている部分を削除
////////////////////////////////////////////////////////////////////////////
if(buf[i]=='<')flag=1;
if(flag==0 && flagh2==1 )
{
 if(isspace(buf[i])==0)
  {

str[j]=buf[i];

 //. が来たら改行
//////////////////////////////////////////

hantei2[m]=buf[i];
if(m>0)copy2[m-1]=hantei2[m];
         if(m==1)
          {

         hantei2[2]='\0';
copy2[1]='\0';

     if(strcmp(hantei2,". ")==0)
{
j++;
str[j]='\n';
}
 strcpy(hozon2,"");
strcpy(hozon2,copy2);
strcpy(hantei2,"");
strcpy(hantei2,copy2);
strcpy(copy2,"");
for(n=1;n<2;n++)copy[n-1]=hozon2[n];
        copy2[1]='\0';

 m=0;
                }
 j++;
```

```
            m++;
      }
////////////////////////////////////////////


}
if(buf[i]=='>')flag=0;


////////////////////////////////////////////////////////////////////////////

            i++;

         }
str[j]='\0';
if(str[j-1]=='\n')str[j-1]='\0';

if(strcmp(str,"\0")!=0)
{

   printf("%s\n",str);
}
counter++;
}

if((r = fclose(f1))== -1)quit("ファイルが閉じれない");

}

void quit(char *s)
{
puts(s); exit(1);
}
```

```
//Google検索結果から，URLのみの抽出

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define LINESIZE 300
#define WORDSIZE 100
#define HANTEISIZE 20

void quit(char *);

int main(int argc, char *argv[])
{
FILE *f1;
char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE],str[LINESIZE],

int r,counter,flag,i,j,k,l,m,n,flagh,flagh2,flagh3;
   counter=flagh=flagh2=flagh3=l=0;
strcpy(str,"");
i=j=l=m=n=0;
if(argc != 2)quit("引数が違う prog datafile");
if((f1 = fopen(argv[1],"r")) == NULL)quit("ファイルが開けない");

  //処理開始

while(fgets(buf,LINESIZE,f1) != NULL){

i=j=k=l=m=n=0;
        strcpy(hantei,"");
        strcpy(copy,"");
        strcpy(copy2,"");
        strcpy(hantei2,"");
        strcpy(hozon,"");
        strcpy(hozon2,"");
        strcpy(url,"http://");
        flagh2=0;
while(buf[i]!='\0'){
```

```
 if(strstr(buf,"<font color=#008000>")!=NULL) flagh2=1;
if(flagh2==1)
{

if(flagh3==1)
{
str[j]=buf[i];
 j++;

}
if(buf[i]==' ')flagh3=0;
//処理
////////////////////////////////////////////////////////////////////////

hantei2[m]=buf[i];
if(m>0)copy2[m-1]=hantei2[m];

   if(m==19)
          {
         hantei2[20]='\0';
copy2[19]='\0';

     if(strcmp(hantei2,"<font color=#008000>")==0)
{
flagh3=1;


}

 strcpy(hozon2,"");
strcpy(hozon2,copy2);
strcpy(hantei2,"");
strcpy(hantei2,copy2);
strcpy(copy2,"");
for(l=1;l<19;l++)copy2[l-1]=hozon2[l];
        copy2[19]='\0';
```

```
 m=18;
                }     //  if(k==20) 終了
 m++;


////////////////////////////////////////////////////////////////////////


}



        i++;

      }
str[j]='\0';
if(str[j-1]=='\n')str[j-1]='\0';

if(strcmp(str,"\0")!=0)
{
  strcat(url,str);
   printf("%s\n",url);


}
counter++;
}

if((r = fclose(f1))== -1)quit("ファイルが閉じれない");

}

void quit(char *s)
{
puts(s); exit(1);
```

```
//名詞と未知語のみとりだす.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define LINESIZE 2048
#define WORDSIZE 100
#define HANTEISIZE 20

void quit(char *);

int main(int argc, char *argv[])
{
FILE *f1;
char buf[LINESIZE],word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE],word4[WORDSIZE

int r,counter,flag,i,j,k,l,m,n,flagh,flagh2,flagh3;
   counter=flag=flagh=flagh2=flagh3=l=0;
strcpy(str,"");
strcpy(tango1,"");
strcpy(tango2,"");
strcpy(tango1,"試合");
strcpy(tango2,"構成");


i=j=l=m=n=0;
if(argc != 2)quit("引数が違う prog datafile");
if((f1 = fopen(argv[1],"r")) == NULL)quit("ファイルが開けない");

  //処理開始

while(fgets(buf,LINESIZE,f1) != NULL){

i=j=k=l=m=n=0;
        strcpy(hantei,"");
        strcpy(copy,"");
        strcpy(copy2,"");
```

```
        strcpy(hantei2,"");
        strcpy(hozon,"");
        strcpy(hozon2,"");
        strcpy(url,"http://");
        flagh2=0;
  strcpy(word1,"");
strcpy(word2,"");
strcpy(word3,"");
strcpy(word4,"");
strcpy(word5,"");
strcpy(word6,"");
strcpy(word7,"");

sscanf(buf,"%s %s %s %s %s %s %s",word1,word2,word3,word4,word5,word6,word7);
if(strcmp(word4,"名詞")==0 || strcmp(word4,"未定義語")==0)
printf("%s ",word1);
if(strcmp(word1,"EOS")==0)
printf("\n");
}


if((r = fclose(f1))== -1)quit("ファイルが閉じれない");

}

void quit(char *s)
{
puts(s); exit(1);
}
```

```
//クラスタリング


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gdbm/ndbm.h>
#include <fcntl.h>
#include <math.h>

#define LINESIZE1 1000
#define LINESIZE2 2000

#define WORDSIZE 20


void quit(char *);
int hantei(int,int,int,char,char *);
void hyouji(int data_number, float min, int gyou, int retu, float mat[][WORDSIZE]
表示関数

int main(int argc, char *argv[])
{
FILE *f1,*f2;
char buf[LINESIZE1],clm1[WORDSIZE],clm2[WORDSIZE],clm3[WORDSIZE],str[WORDSIZE],co
int r,i,j,k,l,gyou,retu,size,kioku1,kioku2,data_number,number,length,change,kioku
float max,real_number,maximam,kioku4_2,kioku4_1;
int flag[2000];
  number=0;
char tango[2000][1000];
float mat[2000][2000];
DBM *mydb;
  datum key,val;


strcpy(copy,"");
strcpy(copy2,"");
```

```
strcpy(copy3,"");
strcpy(str2,"");
int counter=1000;
float count1=0.0;
float count2=0.0;
int count3=0;

i=j=k=gyou=retu=m=data_number=l=kioku1=kioku2=length=change=0;

kioku1_2=kioku2_2=kioku1_3=kioku2_3=0;
data_number=1000;


max=real_number=maximam=kioku4_1=kioku4_2=0.0;
for(i=0;i<2000;i++){
for(j=0;j<2000;j++) mat[i][j]=0.0;

strcpy(tango[i],"");
}

   strcpy(str,"");
if(argc != 3) quit("引数の数が違う"); /* prog datafile  db-filename*/

  if((f1 = fopen(argv[1],"r"))==NULL) quit("ファイルが開けない");
  mydb = dbm_open(argv[2],O_RDONLY, 0640);

retu=-1;
//処理開始

while(fgets(buf,LINESIZE1,f1) != NULL){

 sscanf(buf,"%s %s %s",clm1,clm2,clm3);
//printf(" %s %s %s %daaaaaaaaaaaaaaaaaaaaaaa\n",clm1,clm2,clm3,number);
if(max==0.0) max=atof(clm3);

if(strcmp(str,clm1)!=0)
{
strcpy(tango[number],clm1);
strcpy(str,"");
```

```
strcpy(str,clm1);
//printf("%s  %d\n",tango[number],number);
number++;


}//if(strcmp(str,clm1)!=0)end

real_number=atof(clm3);
if(real_number>max)
{
max=real_number;
strcpy(copy,"");
strcpy(copy2,"");
strcpy(copy3,"");

strcpy(copy,clm1);
strcpy(copy2,clm2);
//printf("%s %sfafdfsssssssssssssss\n",copy,copy2);
}//if(real_number>max)end

}// while(fgets(buf,LINESIZE,f1) != NULL){ end

strcpy(tango[number],clm2);//最後の単語は入らないから clm2 からコピー
//printf("%s %d\n",tango[number],number);

strcpy(clm1,"");
strcpy(clm2,"");
strcpy(clm3,"");

if((r = fclose(f1)) == -1) quit("ファイル 1 が閉じれない");
  if((f1 = fopen(argv[1],"r"))==NULL) quit("ファイルが開けない");

while(fgets(buf,LINESIZE1,f1) != NULL){
 sscanf(buf,"%s %s %s",clm1,clm2,clm3);

if(strcmp(str,clm1)!=0)
{
retu++;
gyou=retu+1;
```

```
strcpy(str,"");
strcpy(str,clm1);

}//if(strcmp(str,clm1)!=0)end

while(1){
if(strcmp(tango[retu],clm1)!=0 || strcmp(tango[gyou],clm2)!=0){

mat[gyou][retu]=0.0;
//printf("%s %s %lf\n",tango[retu],tango[gyou],mat[gyou][retu]);
gyou++;


}//if(strcmp(tango[retu],clm1)!=0 || strcmp(tango[gyou],clm2)!=0){end

else
{
mat[gyou][retu]=atof(clm3);

break;
}//else end
}//while(1){end

//printf("%s %s %s %s %s %lf %d %d\n",clm1,clm2,clm3,tango[retu],tango[gyou],mat[

gyou++;

}//while(fgets(buf,LINESIZE,f1) != NULL){ end

for(j=0;j<number;j++)
{
if(strcmp(copy,tango[j])==0){
//printf("%s %d\n",tango[j],j);
kioku1=j;
}//if(strcmp(copy,tango[j])==0){ end

if(strcmp(copy2,tango[j])==0)
{
//printf("%s %d\n",tango[j],j);
```

```
kioku2=j;
}//if(strcmp(copy2,tango[j])==0) end
}//for(j=0;j<=number;j++)end
maximam=max;
////////////////////////////////////////////////////////
//ループ処理

while(data_number>=100){
//printf("ただいまクラス%d\n",data_number);

//printf("%d %d\n",flag[kioku1],flag[kioku2]);
//データ入力
if(data_number!=1000){
max=0.0;
for(i=0;i<=number;i++)
{
for(j=0;j<=number;j++)
{
//if(i<=j)continue;
//printf("%lf %lf\n",max,maximam);

//if(i==number-1)mat[i][j]=0.0;
//printf("%lf mat[%d][%d]\n",mat[i][j],i,j);

if(mat[i][j]>0.0 && max<mat[i][j] && flag[i]==0 && flag[j]==0){
//printf("%lf\n",mat[i][kioku2]);
max=mat[i][j];
kioku1_2=i;
kioku2_2=j;
//printf("%lf %d %d\n",max,i,j);
}//if(max<mat[i][j]){end

if(kioku1_2>kioku2_2){
change=0;
change=kioku1_2;
kioku1_2=kioku2_2;
kioku2_2=change;
```

```
}//if(kioku1>kioku2){end


}//for(j=0;j<=999;j++)end


}//for(i=0;i<number;i++)end

strcpy(copy,"");
strcpy(copy2,"");
kioku1=kioku1_2;
kioku2=kioku2_2;

strcpy(copy,tango[kioku1]);
strcpy(copy2,tango[kioku2]);

}//if(data_number!=1000){end


//}

//printf("%d クラス\n",data_number);
//printf("%s %s %lf\n",tango[kioku1],tango[kioku2],max);

//データ入力終了

////////////////////////////////////////////////////////////////
//データ検索して計算


number++;
count1=count2=0;

l=0;
m=0;
k=0;
l=strlen(copy);
while(m<=l)
```

```
{
str2[k]=copy[m];
if(str2[k]==',' || str2[k]=='\0'){
count1++;
str2[k]='\0';
k=0;

}//if(str2[k]==',' || str2[k]=='\0'){end
k++;
m++;
}//while(m<=l)end


l=0;
k=0;
m=0;
l=strlen(copy2);

while(m<=l)
{
str2[k]=copy2[m];
if(str2[k]==',' || str2[k]=='\0'){
count2++;
str2[k]='\0';
k=0;

}//if(str2[k]==',' || str2[k]=='\0'){ end

k++;
m++;
}//while end

for(j=0;j<=number;j++)
{
if(j==kioku1)
{
mat[number][j]=0.0;
continue;
}
```

```
if(j==kioku2)
{
mat[number][j]=0.0;
continue;
}
if(j==number)
{
mat[number][j]=0.0;
continue;
}
mat[number][j]=0.0;
kioku4_1=0.0;
kioku4_1=mat[kioku1][j];
if(mat[kioku1][j]==0.0 ){
kioku4_1=mat[j][kioku1];
}
//printf("%lf\n",kioku4_1);
//printf("%lf %d %lf %lf %d\n",count1,kioku1,mat[kioku1][j],mat[j][kioku1],j);
kioku4_2=0.0;
kioku4_2=mat[kioku2][j];
if(mat[kioku2][j]==0.0 ){
kioku4_2=mat[j][kioku2];
}
//printf("%lf %d %lf %lf %d %lf\n",count2,kioku2,mat[kioku2][j],mat[j][kioku2],j,
if(count1==1.0 && count2==1.0){
mat[number][j]=(kioku4_1+kioku4_2)/2;
}
else
{
mat[number][j]=((count1*kioku4_1)+(count2*kioku4_2))/(count1+count2);
}
//printf("%d:%lf %d\n",data_number,mat[number][j],j);
//printf("%lf %lf %lf\n",count1,count2,count1*count2);
kioku4_1=0.0;
kioku4_2=0.0;
}

//printf("%s\n",str2);
//printf("%lf %lf %lf %lf\n",count1,count2,kioku4_2,kioku4_1);
```

```
//printf("%d %d\n",count1,count2);
//}
//strcpy(tango[kioku1],"");
//strcpy(tango[kioku2],"");
strcpy(tango[number],"");
strcat(tango[number],copy);
strcat(tango[number],",");
strcat(tango[number],copy2);
maximam=max;

flag[number]=0;

//printf("%s %lf\n",tango[number],max);
data_number--;
counter--;
//printf("%d %d\n",kioku1,kioku2);
//for(i=0;i<number;i++)
//{
for(j=0;j<=number;j++)
{
//if(i<=j)continue;
/*
mat[kioku1][j]=0.0;
mat[kioku2][j]=0.0;
mat[i][kioku1]=0.0;
mat[i][kioku2]=0.0;
mat[kioku1][kioku2]=0.0;
mat[kioku2][kioku1]=0.0;
*/
}
//}
mat[kioku1][kioku2]=0.0;
mat[kioku2][kioku1]=0.0;
flag[kioku1]=1;
flag[kioku2]=1;
//printf("%d %s %10.8lf %d %d\n",data_number,tango[number],max,kioku1,kioku2);
```

```
//printf("%s\n",tango[number]);
//printf("%s %d %d %lf\n",tango[number],kioku1,kioku2,mat[kioku1][kioku2]);
}//while(data_number>=100){end

if((r = fclose(f1)) == -1) quit("ファイル１が閉じれない");

// f2 = fopen("kekka3.data","w");
count3=0;

 for(j=0;j<2000;j++)
{
if(flag[j]==0 && strcmp(tango[j],"\0")!=0){
printf("クラス%d:%s \n",count3,tango[j]);
count3++;
}
}

//if((r = fclose(f2)) == -1) quit("ファイル２が閉じれない");

}

void hyouji(int data_number, float min, int gyou, int retu, float mat[][WORDSIZE]
{
int i,j,kekka_number;
char henkou[WORDSIZE];
strcpy(henkou,"");

kekka_number=0;
for(i=0;i<=data_number-1;i++)//行数
{


  for(j=0;j<=data_number-1;j++)//列数
   {

    printf("%lf ",mat[i][j]);//タブだと綺麗になる
    }
printf("\n");
}
```

```
printf(" 最小は%lf\n 削除は mat[%d][%d]\n\n",min,gyou+1,retu+1);

}


void quit(char *s)
{
puts(s); exit(1);
}
```

```
//k-means クラスタリング


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define LINESIZE 512
#define WORDSIZE 100

void quit(char *);

int main(int argc, char *argv[])
{

FILE *f1;
char buf[LINESIZE],copy[WORDSIZE][WORDSIZE],copy2[WORDSIZE],str[WORDSIZE],clm1[WO
int r,counter,flag,i,j,k,l,m,n,flagh,flagh2,gyou,retu,data_number,rand,daihyou[10
   counter=flagh=flagh2=mat_gyou=mat_retu=daihyou_number=0;
flag=0;
flagh=0;

double mat[WORDSIZE][WORDSIZE],max,min,kyori,keisan[WORDSIZE],juusinn[100][100],h

strcpy(str,"");
strcpy(clm1,"");
strcpy(clm2,"");
strcpy(clm3,"");
strcpy(clm4,"");
strcpy(clm5,"");
seikika=0.0;
for(i=0;i<=WORDSIZE;i++)
{
strcpy(copy[i],"");
}

retu_count=gyou_count=0;
```

```
ruijido=0.0;
class_gyou=class_retu=0;
min=0.0;
hozon=0.0;
i=j=k=gyou=retu=rand=max=data_number=l=m=0;

//
mat_gyou=24;
mat_retu=99;
daihyou_number=4;
//
ruijido=0.0;
kyori=0.0;
for(i=0;i<=mat_gyou;i++)//行数
{
  for(j=0;j<=mat_retu;j++)//列数
   {
keisan[i]=0.0;
mat[i][j]=0.0;
daihyou[i]=0;

    }
}


//初期値＝０にしてる
for(i=0;i<=100;i++)
{
syokiti[i]=0;

}

for(i=0;i<=100;i++)
for(j=0;j<=100;j++)
{{
class[i][j]=100;
juusinn[i][j]=0.0;
}}
```

```
//初期値＝0にしてるend


if(argc != 2) quit("引数の数が違う"); /* prog datafile  db-filename*/
if((f1 = fopen(argv[1],"r")) == NULL)quit("ファイルが開けない");

while(fgets(buf,LINESIZE,f1) != NULL){
sscanf(buf,"%s  %s  %s  %s   %s",clm1,clm2,clm3,clm4,clm5);
//printf("%s\n",clm5);
if(strcmp(copy[0],"")==0) strcpy(copy[0],clm1);
if(strcmp(clm1,str)!=0 && strcmp(copy[0],clm1)!=0)
{
gyou_count++;
strcpy(copy[gyou_count],clm1);
retu_count=0;
}

ruijido=atof(clm5);
mat[gyou_count][retu_count]=ruijido/1000;
//printf("%lf\n",mat[gyou_count][retu_count]);

strcpy(str,"");
strcpy(str,clm1);
strcpy(clm1,"");
strcpy(clm2,"");
strcpy(clm3,"");
strcpy(clm4,"");
strcpy(clm5,"");
ruijido=0.0;

retu_count++;

}//while(fgets(buf,LINESIZE,f1) != NULL){end

for(i=0;i<=24;i++)
{
for(j=0;j<=99;j++)
{
for(k=0;k<=99;k++)
```

```
{
seikika=seikika+mat[i][k]*mat[i][k];
}

mat[i][j]=(mat[i][j])/sqrt(seikika);
}
printf("%lf\n",seikika);
seikika=0.0;
}



retu_count--;

/*
for(i=0;i<=gyou_count;i++)
{
printf("%s\n",copy[i]);
}
*/


for(i=0;i<25;i++)
{
for(j=0;j<100;j++)
{
//printf("%lf\n",mat[i][j]);
}
}


/*
mat[0][0]=2;
mat[0][1]=5;
mat[1][0]=3;
mat[1][1]=0;
mat[2][0]=4;
mat[2][1]=2;
```

```
mat[3][0]=0;
mat[3][1]=5;
mat[4][0]=3;
mat[4][1]=7;
*/


for(i=0;i<=mat_gyou;i++)//行数
{
  for(j=0;j<=mat_retu;j++)//列数
   {
//printf("%lf\n",mat[i][j]);
    }
}


srandom(time(0));

for(i=0;i<=daihyou_number;i++)//代表点の個数
{
rand=random()%25;//行数

 for(j=0;j<=daihyou_number;j++)//代表点の個数
 {
if(rand==daihyou[j])
{
flag=1;
break;
}
  }
 if(flag==1)
 {
 flag=0;
 i--;
 continue;
 }

daihyou[i]=rand;
printf("代表点%d\n",daihyou[i]);
```

```
}//for(i=0;i<=4;i++)//行数 end

class_gyou=0;

for(i=0;i<=daihyou_number;i++)//代表点の個数
 {
 class[class_gyou][0]=daihyou[i];
 class_gyou++;
 }
class_gyou=0;
flag=0;


for(i=0;i<=daihyou_number;i++)//代表点の数
{
  for(j=0;j<=mat_retu;j++)//列数
   {
//printf("mat[%d][%d]=",daihyou[i],j);//ランダムで選んだ行 (ベクトル)
を表示
//printf("%lf \n",mat[daihyou[i]][j]);
    }
//printf("\n");
}
//printf("\n");
//ランダムで選んだ代表点以外を総和
for(i=0;i<=mat_gyou;i++)//行数
{
class_gyou=0;
flag=0;
   for(j=0;j<=daihyou_number;j++)//代表点の数
  {

if(i==daihyou[j]) flag=1;
  }//   for(j=0;j<=1;j++)//列数 end
min=0.0;
 if(flag==0){
for(j=0;j<=daihyou_number;j++){//代表点個数
```

```
kyori=0.0;
for(k=0;k<=mat_retu;k++)//列数
    {

keisan[k]=keisan[k]+mat[i][k];
//printf("mat[%d][%d]=",i,k);
//printf("%lf\n",mat[i][k]);
//printf("%d %d %d %d\n",i,k,daihyou[j],k);
kyori=kyori+pow(mat[i][k]-mat[daihyou[j]][k],2.0);
//printf("mat[%d][%d]=%lf mat[%d][%d]=%lf\n",i,k,mat[i][k],daihyou[j],k,mat[daihy

    }//for(k=0;k<=1;k++)//列数 end



kyori=sqrt(kyori);
//printf("%lf\n",kyori);



if(j==0)//min との比較が初めてだったら
 {
  min=kyori;
//printf("kyori=%lf min=%lf class=%d\n",kyori,min,class_gyou);
  continue;
  }

if(j!=0 && min>=kyori)
   {
    min=kyori;

    class_gyou=j;
      }
//printf("kyori=%lf min=%lf class=%d\n",kyori,min,class_gyou);
}//for(j=0;j<=1;j++){end

//printf("クラスの行数%d\n",class_gyou);
```

```
for(l=0;l<=100;l++){//全体の数
if(class[class_gyou][l]==100){//100は適当な値.
   class[class_gyou][l]=i;
   break;
 }
}

       }//    if(flag==0){end



}//for(i=0;i<=4;i++)//行数end
printf("\n");


for(i=0;i<=daihyou_number;i++)//クラス(代表点)の数
{
printf("class[%d]=",i);
for(j=0;j<100;j++)//配列の大きさ
{
if(class[i][j]==100) break;
printf("%d  ",class[i][j]);
}
printf("\n");
}
printf("\n");

counter++;
//重心選出
for(i=0;i<=daihyou_number;i++)//i=クラス数
{
  for(j=0;j<=mat_retu;j++)
  {
  for(k=0;k<=100;k++)//配列の大きさ
  {
 if(class[i][k]==100)break;
juusinn[i][j]=juusinn[i][j]+mat[class[i][k]][j];
   }//  for(k=0;k<=100;k++)end
juusinn[i][j]=juusinn[i][j]/k;
```

```
//printf("%lf\n",juusinn[i][j]);
   }  //for(j=0;j<=1;j++)end


}//for(i=0;i<=1;i++)end
//重心選出 end



for(i=0;i<=100;i++)
{
for(j=0;j<=100;j++)
{
class[i][j]=100;

}
}

flag=0;


//クラスタリングのくり返し
do {
for(i=0;i<=100;i++){
for(j=0;j<=100;j++){
class[i][j]=100;
}
}
flag=0;
for(i=0;i<=mat_gyou;i++)//行数
{
class_gyou=0;

for(j=0;j<=daihyou_number;j++){//代表点個数


kyori=0.0;
for(k=0;k<=mat_retu;k++)//列数
```

```
    {


kyori=kyori+pow(mat[i][k]-juusinn[j][k],2.0);
//printf("mat[%d][%d]=%lf mat[%d][%d]=%lf\n",i,k,mat[i][k],daihyou[j],k,mat[daihy

    }//for(k=0;k<=1;k++)//列数 end

//printf("%lf\n",kyori);

kyori=sqrt(kyori);
//printf("%lf\n",kyori);
if(j==0)
 {
  min=kyori;
  continue;
  }
if(j!=0 && min>=kyori)
   {
    min=kyori;
    class_gyou=j;

    }


}//for(j=0;j<=1;j++){end
 //printf("%d\n",class_gyou);

for(l=0;l<=100;l++){
if(class[class_gyou][l]==100){
   class[class_gyou][l]=i;
   break;
 }
}




}//for(i=0;i<=4;i++)//行数 end
```

```
for(i=0;i<=daihyou_number;i++)
{
//printf("class[%d]=",i); //計算結果表示
for(j=0;j<=100;j++)
{
if(class[i][j]==100) break;
//printf("%d ",class[i][j]); //計算結果表示
}
printf("\n");
}
printf("\n");

counter++;

//重心選出
for(i=0;i<=daihyou_number;i++)//i=クラス数
{
  for(j=0;j<=mat_retu;j++)
  {
hozon=juusinn[i][j];
juusinn[i][j]=0.0;

  for(k=0;k<=100;k++)
  {
 if(class[i][k]==100)break;
juusinn[i][j]=juusinn[i][j]+mat[class[i][k]][j];
   }//  for(k=0;k<=100;k++)end
juusinn[i][j]=juusinn[i][j]/k;
if(hozon!=juusinn[i][j])flag=1;

//printf("%lf\n",juusinn[i][j]); //結果表示
   }  //for(j=0;j<=1;j++)end


}//for(i=0;i<=1;i++)end
//重心選出 end
```

```
}while(flag==1);

//クラスタリングのくり返しend
counter++;

printf("------------結果-------------\n");

for(i=0;i<=daihyou_number;i++)
{
//printf("class[%d]=",i); //計算結果表示
for(j=0;j<=100;j++)
{
if(class[i][j]==100) break;
//printf("%d, ",class[i][j]); //計算結果表示
//printf("%s ",copy[class[i][j]]); //計算結果表示

}
//printf("\n");
}
//printf("\n");
//printf("クラスタリング回数は%d 回\n",counter);

if((r = fclose(f1)) == -1) quit("ファイル1が閉じれない");


}




void quit(char *s)
{
puts(s); exit(1);
}
```