修 士 論 文

コーパスからの単語の意味の発見

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

九岡 佑介

2008 年 3 月

修 士 論 文

コーパスからの単語の意味の発見

指導教官　白井 清昭 准教授

審査委員主査　白井 清昭 准教授
審査委員　島津 明 教授
審査委員　東条 敏 教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

0610032 九岡 佑介

提出年月: 2008 年 2 月

概 要

本論文では, 辞書に依存せず単語の意味を識別する語義識別の手法について研究する. 語義識別は単語の出現 (インスタンス) を同じ意味で使われたものがまとまるようにクラスタリングするという問題であり, 既存の辞書に依存せずに単語の意味を判別できる.

この提案手法の特徴は次の 2 つである. まず, インスタンスを様々な素性に基づく複数の特徴ベクトルで表現する. 具体的には, インスタンスの周辺語やその関連語を素性とする文脈ベクトル, 対象語を含む連語を素性とする隣接ベクトル, インスタンスの周辺語を他の単語や PLSI・LDA により推定したトピックとの関連度で特徴付ける連想ベクトル, インスタンスの出現した文書のトピックを PLSI で推定し素性とするトピックベクトルなどを提案する. もう 1 つの特徴として, 本研究では, インスタンスを複数の特徴ベクトルで表現し, 様々なタイプの素性を同時に考慮に入れてクラスタリングを行う. 組み合わせ方式は次の 2 通りである. 1 つは, 異なる特徴ベクトルにおける類似度の重み付き和によってクラスタ間の類似度を測り, クラスタリングする手法である. もう 1 つは, クラスタリングの良さを測る評価関数を導入し, クラスタリング結果が最も良いと思われる特徴ベクトルを単語毎に選択する手法である. クラスタリングの良さは, クラスタの要素が互いにどれだけ似ているか, 異なるクラスタがどれだけ互いに似ていないか, などの観点から測る. また, クラスタ内の要素間の類似度やクラスタ同士の非類似度を相対的に測る評価関数も提案した.

特徴ベクトルを単独で用いてクラスタリングを行ったところ, 隣接・連想・トピック・文脈ベクトルの順にクラスタリングの結果が良いことがわかった. さらに, 隣接・連想・トピックベクトルによるクラスタリング結果の良さの順位が対象語によって異なることがわかった. また, 文脈・隣接・連想ベクトル, 文脈・隣接・トピックベクトル, 文脈・隣接ベクトルを組み合わせ, これらのベクトル間の類似度の重み付き和を 2 つのベクトル間の類似度としてクラスタリングを行うことで, 特徴ベクトルを単独で用いる手法よりクラスタリングの結果が改善した. さらに, 文脈・隣接・連想・トピックベクトルの中から単語毎に最適な特徴ベクトルを選択することで, さらにクラスタリング結果が改善することがわかった. また, クラスタリングの良さを測る評価関数としては, クラスタ内の要素間の類似度を相対的に測る関数が最も有効であった. ただし, Purity や Inverse Purity が最も高い特徴ベクトルを常に選択できたわけではない. そのため, 評価関数の改善によりクラスタリング結果のさらなる改善が期待できる.

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　研究の背景

文中の単語の意味を判別する語義曖昧性解消 (Word Sense Disambiguation; WSD) は, 機械翻訳を始めとする自然言語処理に必要となる基盤技術である [7]. 例えば, 機械翻訳においては, 多義語の意味を WSD により特定して適切な単語に翻訳しなければならない. また情報検索においては, キーワードが多義語であれば, その意味を WSD により特定し, ユーザが意図した語義でキーワードが使われている文書を抽出しなければならない.

しかしながら, 従来の WSD の手法は, 一般に, 岩波国語辞典などの辞書においてあらかじめ定義された語義のいずれかを選択するという問題設定を前提とする. ところが, 語義は時を経るに従って変化するため, 実際に WSD を適用する時点において, 単語が辞書にない意味で使われている場合もありうる. このような場合, 従来の語義曖昧性解消の手法では正しい語義を決めることができない. 頻繁に辞書を改訂するという対処法もあるが, 辞書の作成は人手で行う必要があり, 多くのコストを要する.

1.2　研究の目的

本論文では辞書に依存せず単語の意味を判別する語義識別 (Word Sense Discrimination) の手法について研究する. 語義識別は単語の個々の出現を同じ意味で使われたものがまとまるようにクラスタリングするという問題である. 以降の説明では単語の出現をインスタンスと呼ぶ.

語義識別は WSD とは違い具体的な語義の定義があることを前提としない. 例えば, Schütze は WSD を次の 2 ステップから構成されていると説明している [10].

1. インスタンスを語義ごとに分類してクラスタを構築する
2. クラスタに語義ラベルをつける

WSD は 2. においてインスタンスの語義にラベルを付けなければならず, そのラベルの定義は辞書に依存する. したがってコーパスや辞書の語釈文を教師データとした教師あり学習の手法を利用することが一般的である. しかし, 語義識別は WSD のステップ 1. のみを行う問題であり, 同じ語義で使われたインスタンスをまとめることのみを行う. そして, 構築されたクラスタが既存の辞書のいずれの語義に該当するかということは考慮しないため, 辞書に依存することはない. また, クラスタリングなどの教師無し学習の手法が適用される.

以上で述べたとおり, 語義識別は語義ラベルの定義を必要としない. しかしながら, コーパスにおける複数のインスタンスの語義が同じであるかを判別できれば次のような応用が可能となる.

1. 情報検索において, 自動作成した語義クラスタに基づいて検索文中のキーワードと文書中のキーワードの意味の一致・不一致を判断
2. あらかじめ定義された語義に該当する語義ではないインスタンスの発見
3. 同じ意味で使われたインスタンスを含む用例を自動収集することによる辞書編纂作業の補助
4. 2 に基づく, WSD における辞書にない語義を判別できないという問題の解消

1.3 本論文の構成

本論文の構成は次の通りである.

2 章では, 語義識別, その基盤であるクラスタリング手法, 語義曖昧性解消に関する関連研究を調査し, 本研究の特色を述べる.

3 章では, 本研究で用いる特徴ベクトル, クラスタリング手法, そして複数の特徴ベクトルを組み合わせてクラスタリングを行う手法について述べる.

4 章では, 評価実験を行い, 提案するクラスタリング手法の有効性について述べる.

5 章では, 本研究のまとめ, および今後の展望について述べる.

第2章　関連研究

本論文では日本語の単語のインスタンスを語義別にまとめるという語義識別の手法を提案する. ここでは関連する過去の研究について述べる. まず, 過去に提案された代表的なクラスタリング手法を概観する. また, 文書中の単語の出現頻度の情報から関連のある文書や単語のクラスタ (トピック) を確率的に推定するトピックモデルに関する研究のうち, 本研究で利用するものについて述べる. また, インスタンスのクラスタリングに適切な素性を検討するため, 教師あり学習により語義曖昧性解消を行っている研究についても述べる. 最後に語義識別に関する先行研究についてまとめ, 本研究との相違点を述べる.

2.1　クラスタリング手法の概観

ここでは, 対象データを特に単語のインスタンスと限定せず, 多様なデータに対して一般的に用いられているクラスタリング手法を概観する.

2.1.1　階層的凝集型クラスタリング

階層的凝集型クラスタリングは, N 個のデータに対してそれぞれデータを 1 つずつ含む N 個のクラスターを初期状態として, 最も類似度の高い (もしくは距離の近い) クラスタを併合していく手法である. 併合を繰り返すことで結果的に階層構造が得られる.

また, クラスタ間の類似度の与え方により次の 4 つの手法がある [5].

single linkage clustering

クラスタ間の類似度を計算する際, 互いのクラスタから 1 つずつ要素を選び組をつくる. あらゆる組について類似度を計算し, その最大値をクラスタ間の類似度とする.

complete linkage clustering

クラスタ間の類似度を計算する際, 互いのクラスタから 1 つずつ要素を選び組をつくる. あらゆる組について類似度を計算し, その最小値をクラスタ間の類似度とする.

group-average clustering (UPGMA)

クラスタ間の類似度を計算する際, 互いのクラスタから 1 つずつ要素を選び組にする. あらゆる組について類似度を計算し, その平均をクラスタ間の類似度とする.

セントロイド法 (centroid clustering)

クラスタ間の類似度を計算する際, 互いのクラスタにおけるクラスタ中のベクトルの平均, すなわち重心ベクトル (セントロイド) を計算する. クラスタの重心ベクトル間の類似度をクラスタ間の類似度とする.

2.1.2 分割型クラスタリング

分割型クラスタリングはデータを k 個のクラスタにまず割り当て, クラスタの質が高まるように各データのクラスタへの割り当てを繰り返す手法である. 一般に初期のクラスタの割り当てはランダムであり, そのためにクラスタリング結果は毎回異なる. クラスタ間の類似度や重心の定義によって次のような手法がある.

k-means

k-means は対象データをクラスタの重心とその要素間の距離が最小となるようにクラスタリングするアルゴリズムである. k-means は次のような手順により実行される.

1. 対象データをランダムに k 個のクラスタに分割する
2. クラスタの重心を計算する
3. データを最も重心との距離が近い (類似度が高い) が近いクラスタに割り当てる
4. クラスタの重心やデータの割り当てが収束するまで 2, 3 を繰り返す

k-means 法は「クラスタとその要素間の距離」を表す評価関数を最小化するようなクラスタを求めるように設計されている. ただし, 初期化におけるランダムなクラスタの割り当てによって毎回異なるクラスタリング結果が得られる.

Spherical k-means

Spherical k-means[4] は k-means と類似しているが, 前処理として対象となるベクトルをノルムが 1 になるように正規化する. また通常の k-means 法ではベクトル間の距離は任意のものを用いてよいが, Spherical k-means ではベクトルの内積を用いる. そして目的関数は「クラスタとその要素間の内積」となり, これを最大化するクラスタを反復的に計算する.

fuzzy k-means

k-means 法では各ベクトルは単一のクラスタにのみ割り当てられる. しかしながら, fuzzy k-means では, あるベクトルと各クラスタとの類似度を計算する際に, 類似度の総和が 1 になるように正規化を行う. この正規化された類似度を, クラスタへのベクトルの帰属度と見なす. そしてクラスタの重心は, 全てのベクトルに対するそのクラスタへの帰属度の重み付平均として計算される. fuzzy k-means はソフトクラスタリングの手法であり, 帰属度によって 1 つの要素が複数のクラスタに属する.

混合正規分布+EM アルゴリズム

k 個の多次元正規分布 (ベクトルの正規分布) のパラメータ (多次元正規分布のモードを表す平均ベクトルと共分散行列) を最尤推定することにより, fuzzy k-means のようなクラスタリングを行う手法である. fuzzy k-means におけるベクトルのクラスタへの帰属度に相当するものは, 分布からベクトルが生起する確率である. パラメータの最尤推定には EM Algorithm[3] がよく用いられる.

2.1.3 分割型クラスタリングの比較

前項で分割型クラスタリングの手法を概観した. これらは類似度の与え方やクラスタの代表点の計算方法, またクラスタとベクトル間の類似度の計算する際の次元の重みにより次のように分類できる.

Spherical k-means

- 次元の重みを考慮しない
- データとクラスタとの類似度=内積
- クラスタの代表点=クラスタ内要素の平均
- 実験的に高次元ベクトルに対して高い性能が得られる [4]

fuzzy k-means

- 次元の重みを考慮しない
- データとクラスタとの類似度=内積
- クラスタの代表点=クラスタ内要素の, クラスタ重心との類似度による重み付き平均

混合正規分布 + EM アルゴリズム

- 次元の重みを考慮する
- クラスタへの類似度=多次元正規分布からの生起確率 (正規分布の平均と分散から計算可能)
- クラスタの代表点=正規分布の平均 (パラメータ)

2.2 トピックモデル

トピックモデルとは, 文書のトピックやトピックと関連のある単語をコーパスから教師なし学習するための枠組みである.

一般にトピックモデルの学習では単語を行, 文書を列とする共起行列 $\mathbf{M}$ を学習データとする. $\mathbf{M}$ の要素は列や行に対応する単語 w と文書 d の共起頻度 $n(d, w)$ である. そして, トピック z における単語 w の出現確率の分布 $P(w|z)$ と表現されるトピックと単語の関連度などの確率パラメータを学習する.

トピックモデルの応用としては, ベクトル空間モデルに基づく情報検索において, 空間の次元縮約や文書内の単語出現確率のスムージングにより検索性能を向上させる手法が提案されている. 本研究においても特徴ベクトルの計算に次の 2 つの手法を利用している.

Probabilistic Latent Semantic Indexing

Probabilistic Latent Semantic Indexing(PLSI) は文書をトピックの出現確率で索引付けするための手法である. 具体的にはコーパスから計算した単語 w と文書 d の共起行列 $\mathbf{M}$ を学習データとして, w, d, トピック z 間の関連度を表す確率パラメータを EM アルゴリズムで教師無し学習する. 索引付けや検索には d に関するパラメータが用いられる. トピックを考慮することで, キーワードの多義性に対応でき, 情報検索の性能が向上することが報告されている [6].

PLSI で学習されるパラメータを以下に示す.

- コーパスにおいて z が出現したときに d が出現する確率分布 $P(d|z)$
- コーパスにおいて z が出現したときに w が出現する確率分布 $P(w|z)$
- コーパスにおける z の出現確率分布 $P(z)$

これらのパラメータから計算した $P(w|d)$ や $P(z|d)$ といった確率分布が文書の索引として用いられる. 学習データに無い検索クエリなどの未知の文書 d_q に対しては, Folding-in と呼ばれる EM アルゴリズムにより $P(z|d_q)$ を学習する. 検索クエリと文書の照合はそれぞれの分布の類似度を計算することで行われる.

Latent Dirichlet Allocation

Latent Dirichlet Allocation(LDA) は PLSI の拡張である. PLSI では文書 d 毎にトピックの分布を仮定していたが, LDA ではトピックの分布が Dirichlet 分布に従うと仮定して, Dirichlet 分布のパラメータと, トピック上の単語の分布のパラメータを最尤推定する [1]. これは訓練文書毎にトピックの分布を仮定する PLSI とは異なっている.

2.3 語義曖昧性解消

Yarowsky は対象語のインスタンスの直前・直後の内容語[1]または前方・後方で一番近い内容語という素性のいずれかひとつだけ利用した単純なパターンマッチによる分類器を提案した [12]. テストデータとして人手によりタグ付けした同形異義語, 同音異義語, フラン

[1]原文では content word

ス語における対訳が区別されている語, 同音異義語, OCR で間違えやすい語, 擬似語 (いずれも 2 語義) を用いて行った実験では, 90% から 99% の精度が得られたと報告されている.

玉垣らは岩波国語辞典の語釈文から, 語義別の用例中の格に出現しやすい単語や語義の出現条件に関する情報を基にした分類器を作成し, 対象語のインスタンスの前後の文脈に出現した単語の表記や品詞を素性にした SVM と組み合わせて語義曖昧性解消を行っている [15].

菊田らは辞書において定義された n 個の語義に「未定義語義」を加えた n+1 の語義の中から一つを, 対象語のインスタンスの前後 2 語の基本形や品詞, 前後 10 単語以内の自立語の基本形という素性を基に選択する Naive Bayes モデルを構築して, EM アルゴリズムによる学習を行った [14].

2.4 単語のインスタンスの語義の自動分類

Schütze は単語のインスタンスをクラスタリングすることにより単語の意味を自動的に弁別する Word Sense Discrimination に関する研究を行っている [10]. Schütze の提案した手法では, まずコーパス全体から単語の共起頻度を計算し, 単語 w ごとに, 各次元の値がその次元に対応する単語 v との共起頻度であるような Word Vector と呼ばれるベクトルを計算する. そして, 単語のインスタンス w_i を, その周辺語の Word Vector の重み付き足し算によって計算した Context Vector で表現する. 重みはその単語の idf に似た指標で与える. Context Vector の素性は周辺語と共起しやすい単語の分布である. そして, Context Vector に対し, 階層的凝集型クラスタリングと混合正規分布についての EM アルゴリズムを組み合わせた Buckshot[2] によりクラスタリングを行う.

Purandare らは, Schütze の提案したような間接的な共起を素性とする Second Order Context Vector と, 対象語のインスタンスと直接共起した単語や対象語の前後のウインドウ内における単語 bi-gram を素性とする First Order Context Vector を用いてクラスタリングを行った結果を比較している [9]. クラスタリング手法としては, 凝集型の代表として UPGMA, また分割型の代表として k-means 法を k=2 で n-1 回繰り返して n 個のクラスタを得る Repeated Bisection[13] という手法を試した. Purandare らは, 訓練データが多い場合は First Order Context Vector と UPGMA, また訓練データが少ない場合は Second Order Context Vector と Repeated Bisection がよいと報告している.

また, Pantel らはインスタンスではなく単語をクラスタリングして同義語の集合を発見する研究を行っている [8]. Pantel らはコーパスにおいて構文的関係が成立する単語との

共起頻度を素性とするベクトルで単語を表現し, Clustering By Committee(CBC) によりクラスタリングを行う. CBC は, 単語毎に類似度トップ k 単語の集合を計算し, その中から互いに類似度の低いものをクラスタの代表 (Committee) としてあらかじめ決める. そして, ベクトルを Committee のセントロイドとの類似度が閾値を超えるようなクラスタに割り当てる. 一般に単語は多義であり, 複数の意味クラス (ここではクラスタ) を持つので, ソフトクラスタリングを行っている.

2.5 本研究との関連

本論文では Schütze らと同様に語義の自動識別について研究する. クラスタリング手法については関連研究で述べた Spherical k-means やセントロイド法を用いる. 先行研究に対する提案手法の主な特徴は以下の 2 点である.

- より多くの素性を用いる. 単語の 1 次・2 次共起や, WSD で一般的に使われる直近の単語の表記・品詞, PLSI・LDA により推定したトピックに関する情報などを用いる.
- 従来研究の多くが単語インスタンスを 1 つの特徴ベクトルで表現しクラスタリングを行っていた. これに対し, 本研究では, インスタンスを複数の特徴ベクトルで表現し, 様々なタイプの素性を考慮に入れてクラスタリングを行う.

対象語毎に語義識別に有効な特徴ベクトルは異なる (詳細は 4 章で述べる). そこで, 複数の特徴ベクトルを組み合わせて用いてクラスタリングを行う. 組み合わせ方式は次の 2 通りである.

- 異なる特徴ベクトルにおける類似度の重み付き和によってクラスタ間の類似度を測り, クラスタリングする.
- 単語毎にクラスタリング結果が最もよくなる特徴ベクトルを選択する.

第3章　提案手法

語義識別は単語のインスタンスを意味ごとに分類する問題である. 以後では単語のインスタンスを w_i とする. 本研究における語義識別は, 次の手順で行う.

1. コーパスを用意する.
2. 対象語のインスタンス w_i を抽出する.
3. 対象語のインスタンス w_i を特徴ベクトル $\vec{v}_i$ で表現する.
4. $\vec{v}_i$ をクラスタリングする.
5. 同じクラスタに属するインスタンスは同じ語義であるとみなして w_i の意味を判別する.

以降, 3.1 節では, w_i を表現するための特徴ベクトルについて述べる. また, 3.2 節では, w_i の特徴ベクトルをクラスタリングするための手法について述べる. 最後に, 3.3 節では, 個々の w_i について作成された複数の特徴ベクトルを組み合わせてクラスタリングを行う手法について述べる.

3.1　特徴ベクトル

本研究では, 対象語のインスタンスをそれぞれ異なる素性に基づく特徴ベクトルで表現する. 本章では各種の特徴ベクトルについて, 次の観点から説明する.

- 何を素性とするか
- ベクトルの各要素の重みはどのように決定するか
- その素性が語義識別に有効であることの考察

3.1.1 隣接ベクトル

隣接ベクトル $\vec{n}_i$ は, w_i の直前または直後に現れる単語で w_i を特徴づけるベクトルである. より具体的には, w_i の前後 s 語の単語の基本形ならびに品詞をベクトルの要素とする. すなわち, ある 2 つのインスタンス w_i と w_j に対し, それらの前後 s 語に同じ単語や品詞が出現していれば, w_i と w_j が同じ意味を持つとみなす.

インスタンス w_i の隣接ベクトル $\vec{n}_i$ は次のように作成する. まず w_i の前後 s 語に出現する単語の基本形や品詞という素性にそれぞれベクトル空間の次元をひとつずつ割り当てる. このとき, 同じ基本形や品詞であっても w_i より 2 単語前であるといったような出現位置が異なれば, それぞれに異なる次元を割り当てる. 具体的には w_i の前後 s 語に出現する次のような素性により w_i を特徴づける.

$s = 1$ のとき

- 直前の単語の基本形
- 直前の単語の品詞
- 直前・直後の単語の基本形の組
- 直前・直後の単語の品詞の組

$s = 2$ のとき

- $w = 1$ と同じ素性
- 二つ前の単語の基本形
- 二つ前の単語の品詞
- 前 2 単語の基本形の組
- 後 2 単語の基本形の組

以上の前処理を終えたら, w_i の前後 s 語に出現する単語の基本形や品詞という素性を調べ, 出現した素性それぞれに等しい重みを与える. 最後に $\vec{n}_i$ の要素の和が 1 になるように正規化する.

隣接ベクトル $\vec{n}_i$ は w_i と w_j がそれぞれ同じ連語を構成していれば同じ意味を持つという仮定に基づいた特徴ベクトルである.

3.1.2 文脈ベクトル

文脈ベクトル $\vec{c}_i$ は, w_i の周辺に現れる自立語で w_i を特徴づけるベクトルである. より具体的には, w_i の周辺に現れる自立語の基本形を要素とする. すなわち, ある 2 つのインスタンス w_i と w_j に対し, それらの周辺に同じ自立語が出現していれば, w_i と w_j が同じ意味を持つとみなす. ここでの自立語とは名詞, 動詞, 形容詞, 副詞を指す.

インスタンス w_i の文脈ベクトル $\vec{c}_i$ は次のように作成する. まず w_i の前後 s 語に出現する単語の基本形にそれぞれベクトル空間の次元をひとつずつ割り当てる. このとき, 単語の出現位置が異なっても同じ基本形であれば同じ次元に割り当てる. すなわち, $\vec{c}_i$ は w_i を前後 s 語の bag of words で表現する特徴ベクトルである. 4 章の実験では s は 10 以上に設定した.

以上の前処理を終えたら, w_i の前後 s 語以内に出現する単語の基本形を調べ, それぞれに均等な重みを与える. このとき, 文脈内での単語の出現頻度は重みづけに用いない. 最後に $\vec{c}_i$ の要素の和が 1 になるように正規化する. 正規化を行う理由は, クラスタの重心ベクトルを計算するときに, 前後 s 語内の自立語の数が多い w_i の文脈ベクトルにバイアスがかからないようにするためである.

文脈ベクトルは, w_i と w_j がそれぞれ同じ単語と共起していれば同じ意味を持つという仮定に基づいた特徴ベクトルである.

3.1.3 拡張文脈ベクトル

3.1.2 項で説明した文脈ベクトルは一般にスパースになることが予想される. そのため, w_i と w_j の周辺に出現する自立語に重なりがなく, クラスタリングの手がかりとなる情報が得られない場合がある.

拡張文脈ベクトルは, 文脈ベクトルと同様に w_i の周辺に現れる自立語で w_i を特徴づけるベクトルである. ただし, ベクトルの過疎性を補完するために, w_i の周辺に出現する自立語だけでなく, その関連語も要素とする. すなわち, ある 2 つのインスタンス w_i と w_j に対し, それらの周辺に同じような自立語が出現しているか, あるいは同じような関連語をもつ自立語が出現していれば, w_i と w_j が同じ意味を持つとみなす.

関連語はトピックモデルを利用してコーパスからあらかじめ獲得する. 拡張文脈ベクトルは, 関連語の抽出に利用するトピックモデルの違いにより, 次の 2 種類がある.

PLSI 拡張文脈ベクトル

PLSI 拡張文脈ベクトル $\vec{c^p}_i$ における関連語は次のように獲得する. まず, コーパスから単語 c_j を行, 文書 d_m を列とする共起行列 $\mathbf{A_d}$ を作成する. $\mathbf{A_d}$ の要素 $a_{j,m}$ は, 単語 c_j が文書 d_m に出現した回数とする. 本研究ではコーパスとして文書数が数万程度のものを利用した. また, 単語 c_j として, コーパスにおいて出現する文書数の多いものから順に N_t 個の自立語を選定する[1] . ただし, コーパス全体の 1 割以上の文書に出現する自立語は一般的過ぎるため, クラスタリングに有用でないとみなして除いた. 次に, 共起行列 $\mathbf{A_d}$ に対して PLSI(Probabilistic Latent Semantic Indexing) を適用する. ここでは, PLSI により学習されたパラメタ $P(c_j|z_k)$ を利用する. z_k は隠れ変数であり, 直感的には文書のトピックを表す. $P(c_j|z_k)$ はトピック z_k から単語 c_j が生起する確率を意味しており, あるトピックに関する文書に c_j がどれだけ出現しやすいかを表している.

以上の処理により学習したパラメタから, あるトピック z_k において際立って出現しやすい T 個の単語の集合 Z_k を作成する. 具体的には, 式 (3.1) の値が高い上位 T 個の単語を z_k 毎に選定し, Z_k の要素とする.

$$\log \frac{P(c_j|z_k)}{P(c_j)} \tag{3.1}$$

ただし, $P(c_j)$ はコーパスにおける c_j の出現確率であり, 式 (3.2) のように c_j の出現頻度 $n(c_j)$ から最尤推定で求める.

$$P(c_j) = \frac{n(c_j)}{\sum_{j=1}^{N_t} n(c_j)} \tag{3.2}$$

以上の前処理により推定された関連語の集合 Z_k を利用し, インスタンス w_i の PLSI 拡張文脈ベクトルを次のように作成する. w_i の前後 s 単語中の自立語 c_{lj} に対して重み 1 を与える. また, c_{lj} が Z_k に含まれているなら, Z_k に含まれる c_{lj} 以外の単語に 0.5 の重みを与える.

文脈ベクトルは w_i と w_j の周辺に同じ自立語が出現していなければ類似しているとはみなされない. しかし, PLSI 拡張文脈ベクトルは w_i と w_j の周辺に同じ自立語が出現していなかったとしても, それぞれの周辺語の関連語集合の要素が一致すれば類似度が高く見積もられる. すなわち, 文脈ベクトルの過疎性を克服できる.

[1]実験では $N_t = 20000$ と設定した.

LDA 拡張文脈ベクトル

LDA 拡張文脈ベクトル $\vec{cl}_i$ における関連語は次のように獲得する. まず, コーパスから, 先ほどと全く同じように, 単語 c_j を行, 文書 d_m を列とする共起行列 $\mathbf{A_d}$ を作成する. 次に, 共起行列 $\mathbf{A_d}$ に対して LDA(Latent Dirichlet Allocation) を適用し, LDA により学習されたパラメタ $P(c_j|z_k)$ を利用する. トピック z_k 毎に式 (3.1) の値が最大となる単語を選定し, 関連語の集合 Z_k の要素とする. 式 (3.1) 中のパラメータ $P(c_j|z_k)$ は LDA により学習されたものを使う.

以上の前処理により推定された関連語集合 Z_k を利用し, インスタンス w_i の LDA 拡張ベクトルを PLSI 拡張ベクトルと同じように作成する. つまり, w_i の前後 s 単語のうち自立語 c_{lj} に対して重み 1 を与え, c_{lj} が Z_k に含まれている場合はさらに Z_k 中の c_{lj} 以外の単語に重み 0.5 を与える.

LDA 拡張文脈ベクトルは PLSI 拡張文脈ベクトルと同様に推定されたトピックから関連語集合を抽出する. しかし, PLSI と比較して, その拡張である LDA はトピックの推定精度が改善されている. したがって, LDA によって学習されたトピックから推定した関連語集合は, PLSI によって定義されたものよりも正確であり, クラスタリング性能の向上が期待できる.

3.1.4 連想ベクトル

連想ベクトル $\vec{a}_i$ は, 文脈ベクトルと同様に w_i の周辺に現れる前後 s 個の単語により w_i を特徴付けるベクトルである. ただし, ベクトルの過疎性を克服するために, w_i の周辺語そのものではなく, それと共起する単語を要素とする. すなわち, ある 2 つのインスタンス w_i と w_j において, それらの周辺語がコーパスにおいて似たような単語と共起すれば, w_i と w_j が同じ意味を持つとみなす.

$\vec{a}_i$ の計算を行うには, あらかじめコーパスから行と列をともに単語とする共起行列 $\mathbf{A_w}$ を作成する. $\mathbf{A_w}$ の要素 $a_{l,j}$ は, 単語 c_l と単語 c_j が同じ文書に共起した回数 $n(c_l, c_j)$ とする. 単語 c_l や c_j としては, コーパスにおいて出現する文書数の多いものから順に N_t 個, N_f 個の自立語を選定する[2]. このとき, コーパス全体の 1 割以上の文書に出現する自立語は一般的すぎるためクラスタリングに有用でないとみなして除く. 次に, $\mathbf{A_w}$ の列 j を単語 c_j の共起ベクトル $\vec{o}(c_j)$ とみなす. これは式 (3.3) のように表される.

$$\vec{o}(c_j) = (\ a_{1,j}, ..., a_{N_f,j}\ )^T \tag{3.3}$$

[2]実験では, 文書数が数万以上のコーパスを用いて, N_t は 1,0000, N_f は 1,000 または 10,000 とした.

つまり, $\vec{o}(c_j)$ は c_j と N_f 個の単語の共起頻度を要素とするベクトルである.

以上の処理により得られた c_j の共起ベクトルを使って, 連想ベクトル $\vec{a}_i$ は式 (3.4) と定義する.

$$\vec{a}_i = \sum_{c_j \in context} \vec{o}(c_j) \tag{3.4}$$

式 (3.4) における $context$ は w_i の文脈, c_j はその文脈内に出現する自立語である. つまり, $\vec{a}_i$ は w_i の周辺に現れる自立語 c_j についての共起ベクトル $\vec{o}(c_j)$ の和と定義する. ただし, 出現頻度の上位 N_f 個に含まれていない自立語は, たとえ文脈内に出現していても, 共起ベクトルが存在しないため無視する. 最後に, 連想ベクトルは長さが 1 となるように正規化する.

連想ベクトルは, 文脈ベクトルと同様に前後 s 単語以内に出現する単語 (ただし, 単語の出現位置は考慮しない) を素性とする. つまり, w_i と w_j がそれぞれ同じ単語と共起していれば, 同じ意味を持つという仮定に基づいた特徴ベクトルである. ただし, 文脈ベクトルは周辺に同じ単語が出現していないと類似度を計算できないが, 連想ベクトルでは, コーパスにおいて似たような単語と共起するような単語が周辺に出現していれば類似度が高く見積もられる. すなわち, 特徴ベクトルの過疎性を克服できる.

連想ベクトルは Schütze の Context Vector と類似している. ただし, Context Vector は Word Vector(共起ベクトル $\vec{o}(c_j)$ に相当する) の inverse document frequency(idf) による重み付き和と定義されているが, 連想ベクトルは idf による重みは考慮しない. これは, 評価実験において idf による重みがクラスタリングの質の改善につながらなかったためである. その詳細は 4.3.3 項で述べる. また, 連想ベクトルは長さが 1 となるように正規化を行う. これは, 高次元のベクトルを正規化して, ベクトル間の類似度をコサイン類似度と定義した k-means 法の方が, 正規化を行わずにコサイン類似度以外の距離尺度を利用してクラスタリングした場合と比べて, クラスタリングの性能が向上することが Dhillon によって示されたことを参考にしている [4].

3.1.5 拡張連想ベクトル

3.1.4 項で説明した連想ベクトル $\vec{a}_i$ は, w_i の周辺語 c_j にそれぞれ対応する共起ベクトル $\vec{o}(c_j)$ の和として計算される. このとき, $\vec{o}(c_j)$ は要素が単語 c_l, 重みが c_l と c_j の共起頻度 $a_{l,j}$ であるようなベクトルであった. つまり, w_i の連想ベクトル $\vec{a}_i$ の l 番目の要素 $\vec{a}_i(l)$ は w_i の周辺語 c_j と単語 c_l の共起頻度 $a_{l,j} = n(c_l, c_j)$ の c_j についての和であるともいえる. これは式 (3.5) で表される. ただし, 式 (3.5) 中の j は共起行列 $\mathbf{A_w}$ において単語 c_j に対応す

る列とする.

$$\vec{a}_i(l) = \sum_{c_j \in context} a_{l,j} = \sum_{c_j \in context} n(c_l, c_j) \tag{3.5}$$

このとき, $a_{l,j}$ は c_l と c_j の関連度のようなものを表していると考えられる. したがって, $\vec{a}_i(l)$ は w_i と c_l の間接的な関連度を表しているともいえる.

しかし, 式 (3.5) において $a_{l,j}$ が適切に c_l と c_j の関連度を表しているとはいえない場合もある.

- 例えば, c_l が非常に一般的な単語で, あらゆる文書に出現するような場合, c_l は c_j と必ずしも意味的な関連はない. しかし, この場合でも c_l と c_j の共起頻度 $a_{l,j}$ は高く見積もられることが予想される.

- 逆に, コーパスにおける単語の使われ方に偏りがある場合, 実際には関連のある c_l と c_j があまり同時に使われないということが考えられる. この場合には逆に共起頻度 $a_{l,j}$ は低くなってしまうことが予想される.

- さらに, 選択した N_f 個の単語の中に, 似たような文書で使われる傾向のある単語が複数含まれていた場合, $a_{l,j}$ はそれらの文書に出現しやすい c_l に対してバイアスがかかってしまう. この場合には, $a_{l,j}$ によって c_j の特徴を表すのは適当ではないと考えられる.

したがって, 式 (3.4) を使って, $a_{l,j}$ の周辺語 c_j についての平均として $\vec{a}_i(l)$ を計算しても, それは必ずしも w_i と c_l の関連度を表しているとはいえない.

そこで, 式 (3.5) における共起頻度 $a_{l,j}$ 以外の情報を使って特徴ベクトルの重みを定義することで, ベクトルの過疎性の問題を克服しながら, w_i の周辺語の情報をより適切に表す特徴ベクトルをいくつか定義する.

TWP 連想ベクトル

TWP 連想ベクトル $a^{\vec{twp}_i}$ は, 連想ベクトルと同じく, w_i の周辺に現れる単語で w_i を特徴づけるベクトルである[3]. ただし, ベクトルの過疎性を補完するために, 単語の共起頻度でベクトルの要素の値を決定するのではなく, 単語と単語の共起頻度の情報から PLSI によって推定される単語とトピックの関連度を用いる.

[3]TWP の意味は後ほど説明する.

$a^{\vec{twp}_i}$ の計算を行う前に, 3.1.4 項と同じようにあらかじめコーパスから行と列がそれぞれ単語となる共起行列 $\mathbf{A_w}$ を作成する. 次に, $\mathbf{A_w}$ に対して PLSI を適用し, 単語 c_j やトピック z_k に関する確率パラメタを学習する. このとき, $\mathbf{A_w}$ の列 j で表される単語 c_j を仮想的に文書とみなして PLSI を適用する. 結果として, PLSI により $P(c_j|z_k), P(c_l|z_k), P(z_k)$ というパラメタが得られる. 次に, c_j が出現したときの z_k の出現確率 $P(z_k|c_j)$ をベイズの定理を用いて式 (3.6) によって推定する. ここで, $P(c_j|z_k)$, $P(z_k)$ は PLSI によって得られたパラメタである.

$$P(z_k|c_j) = \frac{P(c_j|z_k)P(z_k)}{P(c_j)} \tag{3.6}$$

また, $P(c_j)$ は式 (3.7) で推定する.

$$P(c_j) = \sum_{z_k} P(c_j|z_k)P(z_k) \tag{3.7}$$

式 (3.6) により計算された $P(z_k|c_j)$ を c_j と z_k の関連度とみなす.

以上の前処理を経て, TWP 連想ベクトルを以下のように定義する. まず, $a^{\vec{twp}_i}$ の素性は PLSI によって得られたトピックとする. そして, $a^{\vec{twp}_i}$ の k 番目の要素 $a^{\vec{twp}_i}(k)$ を式 (3.8) のように定義する.

$$a^{\vec{twp}_i}(k) = \sum_{c_j \in context} P(z_k|c_j) \tag{3.8}$$

すなわち, $a^{\vec{twp}_i}(k)$ は w_i の周辺に現れる単語 c_j について, 確率 $P(z_k|c_j)$ を足した値と定義する. 直感的には $a^{\vec{twp}_i}(k)$ は w_i とトピック z_k の間接的な関連度を表していると考えられる.

TWP 連想ベクトルは, 連想ベクトルと同様に特徴ベクトルの過疎性を克服しながら, 周辺に出現した単語によって w_i を特徴づける. しかし, TWP 連想ベクトルは, ベクトルの要素にトピックとの関連度を使うことで, 先ほど述べたような同じような文書に共起しやすい単語にバイアスがかかってしまうという問題を解消することができると考えられる. これは, PLSI が文書中の単語の多義性に考慮したトピックを学習できることが Hofmann によって示されていることを参考としている [6].

WWP 連想ベクトル

WWP 連想ベクトル $a^{\vec{wwp}_i}$ は, 連想ベクトルと同じく, w_i の周辺に現れる単語で w_i を特徴づけるベクトルである[4]. ただし, ベクトルの過疎性を補完するために, 単語と単語の共起頻度の情報から PLSI によって推定される単語と単語の関連度を使う.

[4]WWP の意味は後ほど説明する.

$a^{\vec{wwp}_i}$ を作成する前に, 3.1.5 項と全く同じように単語と単語の共起行列 $\mathbf{A_w}$ に対して PLSI を適用して, パラメタ $P(c_j|z_k), P(c_l|z_k), P(z_k)$ を得る. また, 単語 c_j が出現したときのトピック z_k の出現確率 $P(z_k|c_j)$ を式 (3.6) で計算する. 次に, 式 (3.9) により単語 c_j が出現したときの c_l の出現確率を計算する.

$$P(c_l|c_j) = \sum_{z_k} P(c_l|z_k)P(z_k|c_j) \tag{3.9}$$

以上の前処理を経て, WWP 連想ベクトルを以下のように定義する. まず,$a^{\vec{wwp}_i}$ の素性は単語とする[5]. そして, $a^{\vec{wwp}_i}$ の l 番目の要素 $a^{\vec{wwp}_i}(l)$ を式 (3.10) のように定義する.

$$a^{\vec{wwp}_i}(l) = \sum_{c_j \in context} P(c_l|c_j) \tag{3.10}$$

すなわち, $a^{\vec{wwp}_i}(l)$ は w_i の周辺に現れる単語 c_j について $P(c_l|c_j)$ を足した値と定義する. 直感的には, $a^{\vec{wwp}_i}(l)$ は w_i と c_l の間接的な関連度を表している.

単語間の関連度を使う点では 3.1.4 項の連想ベクトルと同じである. ただし, 先ほど述べたように, 連想ベクトルの要素の重みを式 (3.5) のように計算すると, 単語の使われ方に偏りがある場合に w_i の特徴を適切に表現できない可能性がある.

式 (3.5) における $n(c_l, c_j)$ は, 式 (3.11) に示す最尤推定された $P(c_l|c_j)$ とほぼ同じ意味をもつと考えられる.

$$P(c_l|c_j) = \frac{n(c_l, c_j)}{n(c_j)} \tag{3.11}$$

WWP 連想ベクトルは, 式 (3.9) が式 (3.11) より良い確率の推定値を与えるならば, 式 (3.10) が式 (3.5) によって計算された w_i と c_l の関連度のより良い推定値を与えるだろうという考えに基づいている. なお式 (3.9) が式 (3.11) より良い推定値を与えることは, Hofmann が情報検索システムにおける実験により示している [6].

TDP 連想ベクトル

TDP 連想ベクトル $a^{\vec{tdp}_i}$ は, TWP 連想ベクトルと同様に, 単語とトピックの関連度の情報を使って特徴ベクトルの過疎性を克服する[6]. ただし, 単語と単語の共起行列 $\mathbf{A_w}$ ではなく, 文書と単語の共起行列 $\mathbf{A_d}$ に対して PLSI を適用する.

$a^{\vec{tdp}_i}$ の計算を行う前に, 3.1.3 項と同じように, あらかじめコーパスから行と列がそれぞれ単語と文書であるような共起行列 $\mathbf{A_d}$ を作成する. 次に, $\mathbf{A_d}$ に対して PLSI を適用

[5]WWP の意味は後ほど説明する
[6]TDP の意味は後ほど説明する

し, 単語 c_j やトピック z_k に関する確率パラメタを学習する. その結果として, $P(d_j|z_k)$, $P(c_j|z_k)$, $P(z_k)$ というパラメタが得られる. このうち, $P(c_j|z_k), P(z_k)$ を用いて, $P(z_k|c_j)$ を式 (3.12) により計算する.

$$P(z_k|c_j) = \frac{P(c_j|z_k)P(z_k)}{P(c_j)} \tag{3.12}$$

ただし, $P(c_j)$ は式 (3.2) により最尤推定する. 式 (3.12) により計算された $P(z_k|c_j)$ を c_j と z_k の関連度とする.

以上の前処理を経て, $a^{\vec{tdp}}{}_i$ を以下のように定義する. まず, $a^{\vec{tdp}}{}_i$ の要素は PLSI によって推定されたトピック z_k とする. そして, $a^{\vec{tdp}}{}_i$ の k 番目の要素 $a^{\vec{tdp}}{}_i(k)$ を式 (3.13) のように定義する.

$$a^{\vec{tdp}}{}_i(k) = \sum_{c_j \in context} P(z_k|c_j) \tag{3.13}$$

つまり, $a^{\vec{tdp}}{}_i$ は, w_i の周辺に現れる自立語 c_j について確率 $P(z_k|c_j)$ を足した値と定義する.

TDP 連想ベクトルは TWP 連想ベクトルと同様に, w_i の周辺語 c_j とトピック z_k の関連度から, w_i と z_k の関連度を推定する. しかし, TWP 連想ベクトルは単語と単語の共起頻度の行列 $\mathbf{A_w}$ から推定されるトピックを用いるのに対して, TDP 連想ベクトルは文書と単語の共起頻度の行列 $\mathbf{A_d}$ から推定されるトピックを用いる. $\mathbf{A_d}$ を用いる方が PLSI が本来仮定するモデルとは近い [6].

TDL 連想ベクトル

TDL 連想ベクトル $a^{\vec{tdl}}{}_i$ は, TDP 連想ベクトルと同様に, 単語とトピックの関連度の情報を使って特徴ベクトルの過疎性を克服する[7]. トピックを学習する元となるのも TDP 連想ベクトルと同様に文書と単語の共起行列 $\mathbf{A_d}$ である. しかし, 学習には PLSI ではなく LDA を用いる.

$a^{\vec{tdl}}{}_i$ の計算を行う前に, あらかじめ 3.1.3 項と同じように, $\mathbf{A_d}$ に対して LDA を適用する. LDA によりトピック z_k が出現したときの c_j の出現確率 $P(c_j|z_k)$ が得られる. $P(c_j|z_k)$ を c_j と z_k の関連度とする.

以上の前処理を経て, $a^{\vec{tdl}}{}_i$ を以下のように定義する. まず, $a^{\vec{tdl}}{}_i$ の要素は LDA によって推定されたトピック z_k とする. そして, $a^{\vec{tdl}}{}_i$ の k 番目の要素 $a^{\vec{tdl}}{}_i(k)$ を式 (3.14) のように定義する.

$$a^{\vec{tdl}}{}_i(k) = \sum_{c_j \in context} P(c_j|z_k) \tag{3.14}$$

[7]TDL の意味は後ほど説明する

つまり, $\vec{a^{tdl}}_i(k)$ は, w_i の周辺に現れる自立語 c_j について c_j と z_k の関連度を足した値と定義する.

TDL 連想ベクトルは, TDP 連想ベクトルと同様に, w_i の周辺語 c_j とトピック z_k の関連度から, w_i と z_k の関連度を推定する. その際に PLSI の拡張である LDA を利用する. LDA は PLSI とは異なる仮定に基づいたトピックモデルであり, 学習されるトピックも PLSI とは異なることが予想される. したがって, TDL 連想ベクトルは TDP 連想ベクトルとは異なる性質を持つことが考えられる.

連想ベクトルのまとめ

3.1.4 項および 3.1.5 項で説明した連想ベクトルとその拡張について表 3.1 にまとめる. $\mathbf{A_w}$ は単語と単語の共起行列で, その要素は単語の共起頻度である. また, $\mathbf{A_d}$ は文書と単語の共起行列で, その要素は文書と単語の共起頻度である. これらの共起行列に対して, PLSI や LDA を適用してパラメタを学習した.

表 3.1 において, 「要素」は w_i の特徴ベクトルの要素である. また, 「重み」は特徴ベクトルの要素に対する計算式を表す. いずれのベクトルの重みも, w_i の周辺に現れる c_j について c_j と単語 (c_l) またはトピック (z_k) との関連度を計算し, その和として与えられる. すなわち, $n(c_l, c_j)$ や $P(c_l|c_j)$ は単語 c_j と c_l の関連度, また $P(z_k|c_j)$ と $P(c_j|z_k)$ は単語 c_j とトピック z_k の関連度と見なしている. これにより, 特徴ベクトルの過疎性を補完している.

なお, 本項で提案した 4 つの拡張連想ベクトルの名称で, 連想ベクトルの前にある 3 つのアルファベット, TWP, WWP, TDP, TDL は表 3.1 の違いを表す. 具体的には, 最初のアルファベットは表 3.1 の「要素」(T はトピック, W は単語), 2 番目のアルファベットは「共起行列」(W は単語×単語の行列 $\mathbf{A_w}$, D は文書×単語の行列 $\mathbf{A_d}$), 3 番目のアルファベットは「トピックモデル」(P は PLSI, L は LDA) を表す.

表 3.1: 連想ベクトルとその拡張の相違点

名前	要素	重み	共起行列	トピックモデル
連想ベクトル	単語 c_l	$\sum_{c_j \in context} n(c_l, c_j)$	$\mathbf{A_w}$	なし
TWP 連想ベクトル	トピック z_k	$\sum_{c_j \in context} P(z_k\|c_j)$	$\mathbf{A_w}$	PLSI
WWP 連想ベクトル	単語 c_l	$\sum_{c_j \in context} P(c_l\|c_j)$	$\mathbf{A_w}$	PLSI
TDP 連想ベクトル	トピック z_k	$\sum_{c_j \in context} P(z_k\|c_j)$	$\mathbf{A_d}$	PLSI
TDL 連想ベクトル	トピック z_k	$\sum_{c_j \in context} P(c_j\|z_k)$	$\mathbf{A_d}$	LDA

3.1.6 トピックベクトル

トピックベクトル $\vec{t_i}$ は, PLSI によって推定されるトピックによって w_i を特徴づけるベクトルである. ここでのトピックは, w_i が出現する文書のトピックを表している. つまり, ある 2 つのインスタンス w_i と w_j に対して, それらの出現した文書が同じようなトピックに関するものであれば, w_i と w_j が同じ意味を持つとみなす. $\vec{t_i}$ の計算を行う前に, あらかじめ 3.1.3 項と同じように, 文書と単語の共起頻度を要素とする共起行列 $\mathbf{A_d}$ を作成する. そして, $\mathbf{A_d}$ に対して PLSI を適用する. ここでは, PLSI より得られたパラメタ $P(c_j|z_k)$ を利用する.

以上の前処理を経て, トピックベクトル $\vec{t_i}$ を以下のように定義する. まず, $\vec{t_i}$ の要素は PLSI によって得られたトピック z_k とする. そして, $\vec{t_i}$ の k 番目の要素 $\vec{t_i}(k)$ を式 (3.15) のように定義する. ただし, d_i は w_i が出現する文書とする.

$$\vec{t_i}(k) = P(z_k|d_i) \tag{3.15}$$

式 (3.15) 中の $P(z_k|d_i)$ は, d_i を学習データに含まれない未知の文書と見なして, PLSI の Folding-in[6] と呼ばれる方法で推定する. 具体的には, 式 (3.16) の E ステップと式 (3.17) の M ステップによる EM アルゴリズムを実行する. ただし, 式において $n(d_i)$ は d_i に含まれる自立語ののべ数, $n(d_i, c_j)$ は d_i における c_j の出現回数とする. $P(c_j|z_k)$ はあらかじめコーパスから学習しておいたものを用い, Folding-in のための EM アルゴリズムでは変更しない.

E-step

$$P(z_k|c_j, d_i) = \frac{P(c_j|z_k)P(z_k|d_i)}{\sum_{z_{k'}} P(c_j|z_{k'})P(z_{k'}|d_i)} \tag{3.16}$$

M-step

$$P(z_k|d_i) = \frac{\sum_{c_j} n(d_i, c_j)P(z_k|c_j, d_i)}{n(d_i)} \tag{3.17}$$

$\vec{t_i}$ は, トピック z_k を要素とし, $P(z_k|d_i)$ を重みとする特徴ベクトルである. つまり, $\vec{t_i}$ は w_i そのものではなく, w_i の出現する文書 d_i の特徴を表している. したがって, トピックベクトルは, w_i と w_j が同じようなトピックを持つ文書に現れていれば, w_i と w_j が同じ意味であるという仮定に基づく特徴ベクトルであるといえる.

3.2 クラスタリング

本研究における語義識別では, 単語インスタンスをクラスタリングする. 本研究では, ベクトル空間モデルに基づくクラスタリングアルゴリズムを適用する. ベクトル空間モデルに基づくクラスタリングアルゴリズムは, ベクトル間の類似度が計算できることを前提としている. そして, 類似度が高いベクトル同士が同じクラスタにまとまるようにクラスタリングが行われる. 本節では, 本研究で用いるベクトル間の類似度の定義とクラスタリングアルゴリズムについて説明する.

3.2.1 クラスタリング手法

語義識別では, 対象語のインスタンス w_i を特徴ベクトル $\vec{v}_i$ で表現し, 特徴ベクトルを用いて w_i をクラスタリングする. 対象語のインスタンス w_i と w_j に対して, それぞれの特徴ベクトルを $\vec{v}_i$ と $\vec{v}_j$ とする. ここで用いるクラスタリングアルゴリズムは, $\vec{v}_i$ と $\vec{v}_j$ の類似度 sim が高くなるような w_i と w_j がまとまるようにクラスタを作成する. したがって, 類似度 sim の計算は必須である. 本研究では, sim として式 (3.18) のようなコサイン類似度を用いる.

$$sim(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{|\vec{v}_i| \cdot |\vec{v}_j|} \tag{3.18}$$

コサイン類似度とは, ベクトルの内積をベクトルの長さで正規化したものである. また, コサイン類似度は, 直感的には二つのベクトルの方向の一致度を表している. ベクトルの方向がまったく一致していなければ 0, 完全に一致していれば 1 となる.

コサイン類似度は, 特徴ベクトルの次元数が一般に高い, ベクトル空間モデルに基づく情報検索でよく使われている類似度である. また, コサイン類似度を用いた k-means 法は他の類似度を用いた k-means 法と比較して高次元な特徴ベクトルのクラスタリングに有利であることが実験的に示されている [4]. 本研究では単語を要素とした高次元の特徴ベクトルを用いる. これは, ベクトル空間モデルに基づく情報検索システムにおいて単語を要素とした特徴ベクトルで文書を表すことと同等であると考えられる. したがって, 情報検索システムで良い結果が得られるとされるコサイン類似度をベクトル間の類似度としてクラスタリングを行う.

次に, 本研究で用いるクラスタリングアルゴリズムについて説明する.

凝集型クラスタリング (セントロイド法)

凝集型クラスタリングは, 最も類似度の高いクラスタを併合するという処理の繰り返してクラスタリングを行う手法である. 凝集型クラスタリングでは併合の繰り返しにより階層構造が生成されるが, 本研究では特に階層構造を必要としない. そのため, あらかじめクラスタ数 k を指定する. そして, 併合が進みクラスタ数が k 個となった時点でクラスタリングを終了する.

ここで用いるセントロイド法は, 凝集型クラスタリングの一種である. セントロイド法では, クラスタ間の距離が各クラスタの重心ベクトル間の距離と定義されている. セントロイド法のアルゴリズムを図 3.1 に示す. 1 行目から 5 行目では, クラスタリング対象となる

アルゴリズム 1 セントロイド法

Input: クラスタの個数 k, クラスタリング対象のベクトルのリスト V
Output: クラスタのリスト C

1: C は空のクラスタのリスト
2: **for** $\vec{v}_i \in V$ **do**
3: $\quad \pi_j \leftarrow \{\ \vec{v}_i\ \}$
4: $\quad \pi_j$ を C に加える
5: **end for**
6: **repeat**
7: $\quad (\pi_j, \pi_k) \leftarrow \underset{(\pi_j,\pi_k)}{\operatorname{argmax}}\, sim(\vec{g}_j, \vec{g}_k)$
8: $\quad \pi_j$ と π_k をマージする
9: **until** $|C| > k$
10: **return** C

図 3.1: セントロイド法のアルゴリズム

るベクトル $\vec{v}_i$ を一つずつ含むクラスタ π_j を作成し, クラスタのリスト C に追加している. つまり, $\vec{v}_i$ が N 個あれば, N 個の π_j が C に作成される. そして, 6 行目から 9 行目では, C から類似度が最大となるクラスタのペア π_j と π_k を検索し, それぞれのクラスタの要素を全て含む新規のクラスタを作成し C に追加する. π_j の重心ベクトルを $\vec{g}_j$, π_k の重心ベクトルを $\vec{g}_k$ とすると, π_j と π_k の類似度は $\vec{g}_j$ と $\vec{g}_k$ のコサイン類似度として計算する. π_j の重心ベクトル $\vec{g}_j$ は式 (3.19) と定義する. ただし, 式 (3.19) において $|\pi_j|$ はクラスタ π_j に

含まれるベクトルの数を表す.

$$\vec{g}_j = \frac{1}{|\pi_j|} \sum_{\vec{v}_i \in \pi_j} \vec{v}_i \tag{3.19}$$

すなわち, $\vec{g}_j$ は π_j に含まれるベクトルの平均と定義する.

その後, π_j と π_k は C から削除する. 以上のステップを C に含まれるクラスタの数 $|C|$ が指定されたクラスタ数 k に到達するまで繰り返し, 最終的に k 個のクラスタのリスト C を出力する.

Spherical k-means

k-means 法は一般的にはクラスタ数 k を指定して, 目的関数を最適化するクラスタを反復的に計算する手法である. その中でも Spherical k-means[4] は, あらかじめベクトルの長さが 1 となるように正規化しておく. そして, 「あらゆるクラスタにおける, クラスタとその要素間のコサイン類似度の和」を表す目的関数を最大化するようなクラスタを反復的に計算する. 目的関数は式 (3.20) となる. 同式において, k はクラスタ数, π_j は j 番目のクラスタ, $\vec{v}_i$ はクラスタの要素, $\vec{g}_j$ は j 番目のクラスタの重心を表す.

$$\mathcal{Q} = \sum_{j=1}^{k} \sum_{\vec{v}_i \in \pi_j} \vec{v}_i \cdot \vec{g}_j \tag{3.20}$$

また, 式 (3.20) において, $\vec{v}_i \cdot \vec{g}_j$ は $\vec{v}_i$ と $\vec{g}_j$ の内積を表している. 本研究においては, ベクトル間の類似度にコサイン類似度を使うと述べたが, Spherical k-means では, 式 (3.20) 中の内積は式 (3.18) で示したコサイン類似度と同じである. なぜなら, Spherical k-means ではベクトルがあらかじめ長さ 1 に正規化されているため, 式 (3.18) 中の分母が 1 となるからである.

Spherical k-means のアルゴリズムを図 3.2 に示す. 1 行目から 4 行目は初期化処理である. 1 行目では, k 個の空のクラスタを作成している. 2 行目から 4 行目では, クラスタリング対象の各ベクトル $\vec{v}_i$ を k 個のクラスタのいずれかにランダムに割り当てている. そして, 5 行目から 13 行目で式 (3.20) を最適化するクラスタの割り当てを反復的に計算する. 6 行目から 8 行目では, 各クラスタ π_j に割り当てられている各ベクトル $\vec{v}_i$ の重心ベクトルを計算する. 9 行目から 12 行目では, 各 $\vec{v}_i$ を最も近い重心ベクトル $\vec{g}_j$ に対応するクラスタ π_j に割り当てる. $\vec{g}_j$ は式 (3.19) により計算する. 9 行目から 12 行目で, もし π_j に割り当てられた割り当てが変化しなかった場合, 式 (3.20) が収束したとみなして, π_j のリスト C を出力して終了する. そうでなければ, 再度 6 行目以降を繰り返す.

アルゴリズム 2 Spherical k-means

Input: クラスタ数 k, クラスタリング対象語のベクトルのリスト V
Output: クラスタのリスト C

C は k 個の空のクラスタのリスト
for $\vec{v}_i \in V$ **do**
  $\vec{v}$ をランダムにクラスタに割り当てる
end for
repeat
  for $\pi_j \in C$ **do**
    π_j の重心ベクトル $\vec{g}_j$ を計算する
  end for
  for $\vec{v}_i \in V$ **do**
    $\pi_j \leftarrow \underset{\pi_j}{\operatorname{argmax}}\, sim(\vec{g}_j, \vec{v}_i)$
    $\vec{v}_i$ を π_j に割り当てる
  end for
until ベクトルのクラスタへの割り当てが変化しない
return C

図 3.2: Spherical k-means のアルゴリズム

3.3 複数の特徴ベクトルの組み合わせ

クラスタリングに有効な特徴ベクトルは対象語毎に異なると考えられる[8]. そこで, それぞれ異なる素性に基づく複数の特徴ベクトルを組み合わせてクラスタリングを行う. これにより, いずれの対象語についても, その対象語に有効な特徴ベクトルを生かすことができると考えられる. 本研究では特徴ベクトルの組み合わせ方として 2 つの方法を用いる.

3.3.1 類似度の組み合わせ

ここでは, インスタンス w_i やクラスタ π_j に関する類似度を, 異なる素性に基づくそれぞれの特徴ベクトルの類似度の重み付き和として定義してクラスタリングに用いる. 本節では, 類似度の組み合わせによるセントロイド法と k-means 法について説明する.

両手法に共通するポイントは次の通りである.

[8] このことに関する実験的考察については, 4.3.2 項で述べる.

1. クラスタリング対象となる単語インスタンス w_i をそれぞれ異なる素性に基づく N 通りの特徴ベクトル $\vec{v}_i^{(n)}$ $(1 \leq n \leq N)$ で表現する.

2. k-means またはセントロイド法によりクラスタリングを行う. そのとき, それぞれ異なる特徴ベクトル $\vec{v}_i^{(n)}$ についてクラスタの重心ベクトル $\vec{g}_j^{(n)}$ は 1 つずつ, 計 N 個存在することになる. クラスタ間の類似度は, $\vec{g}_j^{(n)}$ $(1 \leq n \leq N)$ による類似度の重み付き和と定義する. また, w_i とクラスタ π_j の類似度は, それぞれ N 個の $\vec{v}_i^{(n)}$ と $\vec{g}_j^{(n)}$ との類似度の重み付き和と定義する. ただし, 重み $weight_n$ は実験的に適切なものを決める.

以降では, それぞれの手法について説明する.

類似度の組み合わせによるセントロイド法

類似度の組み合わせによるセントロイド法は, クラスタ π_j と π_k の類似度を式 (3.21) のように定義する. ただし, N は特徴ベクトルの個数である.

$$sim(\pi_j, \pi_k) = \sum_{n=1}^{N} weight_n \cdot sim(\vec{g}_j^{(n)}, \vec{g}_k^{(n)}) \tag{3.21}$$

ここで, $weight_n$ は特徴ベクトル $\vec{v}_i^{(n)}$ に対する重みを表す. 実際には $weight_n$ は実験的に適切な値を決める. また, $\vec{g}_j^{(n)}$ は π_j の特徴ベクトル $\vec{v}_i^{(n)}$ に関する重心ベクトルを表し, 式 (3.22) のように定義する.

$$\vec{g}_j^{(n)} \leftarrow \frac{1}{|\pi_j|} \sum_{\vec{v}_i^{(n)} \in \pi_j} \vec{v}_i^{(n)} \tag{3.22}$$

つまり, 類似度の組み合わせによるセントロイド法において, π_j と π_k の類似度は, n 番目の特徴ベクトル $\vec{v}_i^{(n)}$ に関するクラスタの重心 $\vec{g}_j^{(n)}$ と $\vec{g}_k^{(n)}$ の類似度の重み付き和と定義する. これは, 複数の特徴ベクトルにおいて類似度が高くなるような場合, π_j と π_k が同じ単語の意味を表すクラスタである可能性が高いという仮定に基づいている.

類似度の組み合わせによるセントロイド法の流れを図 3.3 に示す.

アルゴリズム 3 類似度の組み合わせによるセントロイド法

Input: インスタンスのリスト W, クラスタ数 k
Output: クラスタのリスト C

1: $C \leftarrow k$ 個の空のクラスタのリスト
2: **for** $w_i \in W$ **do**
3: **for** $n = 1$ to N **do**
4: w_i の特徴ベクトル $\vec{v}_i^{(n)}$ を計算する
5: **end for**
6: **end for**
7: **for** $w_i \in W$ **do**
8: $\pi_j \leftarrow \{\ w_i\ \}$
9: π_j を C に加える
10: **end for**
11: **repeat**
12: $(\pi_j, \pi_k) \leftarrow \underset{(\pi_j,\pi_k)}{\operatorname{argmax}}\, sim(\pi_j, \pi_k)$
13: π_j と π_k をマージする
14: **until** $|C| > k$
15: **return** C

図 3.3: 類似度の組み合わせによるセントロイド法

1 行目から 11 行目は初期化を行っている. まず, 1 行目から 7 行目では, 各 w_i について N 個の特徴ベクトル $\vec{v}_i^{(n)}$ を計算して, 後にクラスタの重心ベクトルの計算に利用するために保持している. 次に, 8 行目から 11 行目では, 各 w_i を一つずつ含むクラスタを作り, C に追加している. したがって, クラスタリング対象のインスタンスの数を $|W|$ と表すと, 11 行目の時点で $|W|$ 個のクラスタが作成される. そして, 12 行目から 15 行目では, 最も類似度の高いクラスタを繰り返し併合している. ただし, クラスタ間の類似度は, 式 (3.21) により計算される. 併合の結果, C に含まれるクラスタの数が k となったら処理を完了する.

類似度の組み合わせによる k-means 法

類似度の組み合わせによる k-means 法では, クラスタ π_j とインスタンス w_i の類似度は式 (3.23) で定義する.

$$sim(\pi_j, w_i) = \sum_{n=1}^{N} weight_n \cdot sim(\vec{g}_j^{(n)}, \vec{v}_i^{(n)}) \tag{3.23}$$

ただし, $\vec{g}_j^{(n)}$ は π_j の n 番目の特徴ベクトルに関する重心ベクトルを表し, 式 (3.22) で定義する. つまり, 類似度の組み合わせによる k-means 法では, π_j と w_i の類似度はそれぞれの特徴ベクトルに関する重心ベクトル $\vec{g}_j^{(n)}$ と特徴ベクトル $\vec{v}_i^{(n)}$ の類似度の重み付きと定義される. これは, 複数の特徴ベクトルにおいて類似度が高くなるような場合, π_j は w_i と同じ意味に関するクラスタである可能性が高いという仮定に基づいている.

類似度の組み合わせによる k-means 法の流れを図 3.4 に示す. 1 行目から 10 行目は初期化処理を行う. まず, 1 行目から 7 行目では, 各 w_i について, N 個の特徴ベクトル $\vec{v}_i^{(n)}$ を作成している. $\vec{v}_i^{(n)}$ は後のステップで, 重心ベクトル $\vec{g}_j^{(n)}$ の計算や, クラスタ π_j と w_i の類似度の計算に利用される. 次に, 8 行目から 10 行目では, 各 w_i をランダムにクラスタ $\pi_j \in C$ に割り当てる. 以上の初期化を経て, 11 行目から 21 行目では C 内のクラスタを反復的に最適化する. まず, 12 行目から 16 行目では各 π_j について, n 番目の特徴ベクトルに対する重心ベクトル $\vec{g}_j^{(n)}$ を再計算する. そして, 17 行目から 21 行目では, 各 w_i を類似度が最も高くなる π_j に割り当て直す. 類似度は式 (3.23) で計算される. その後, 11 行目からの処理で各 π_j に対する w_i の割り当てが変化していれば 11 行目以降を繰り返す. 割り当てが変化していなければ, C を出力して処理を完了する.

アルゴリズム 4 類似度の組み合わせによる k-means 法

Input: インスタンスのリスト W, クラスタ数 k
Output: クラスタのリスト C

$C \leftarrow k$ 個の空のクラスタのリスト
for $w_i \in W$ **do**
  for $n = 1$ to N **do**
    w_i の特徴ベクトル $\vec{v}_i^{(n)}$ を計算する
  end for
end for
for $w_i \in W$ **do**
  w_i をいずれかのクラスタ $\pi_j \in C$ にランダムに割り当てる
end for
repeat
  for $\pi_j \in C$ **do**
    for $f \in F$ **do**
      式 (3.22) により $\vec{g}_j^{(n)}$ を計算する
    end for
  end for
  for $w_i \in W$ **do**
    $\pi_j \leftarrow \underset{\pi_j}{\operatorname{argmax}}(sim(\pi_j, w_i))$
    w_i を π_j に割り当てる
  end for
until クラスタの割り当てが変化しない
return C

図 3.4: 類似度の組み合わせによる k-means 法

3.3.2 単語毎に特徴ベクトルを選択する手法

クラスタリングの対象となる単語に応じて適切な特徴ベクトルをクラスタリングの評価値に基づいて選択する手法を提案する. このアルゴリズムは次のようになる.

1. 異なる素性に基づく N 通りの特徴ベクトル $\vec{v}_i^{(n)}$ により独立にクラスタリングを行う. それぞれのクラスタリング結果を $C_1, C_2, ..., C_N$ と表す.

2. $\underset{n}{\mathrm{argmax}}(eval(C_n))$ を満たす特徴ベクトル $\vec{v}_i^{(n)}$ を選択する.

3. 特徴ベクトル $\vec{v}_i^{(n)}$ によるクラスタリング結果 C_n を出力する.

以上のアルゴリズムは, 利用可能な特徴ベクトルでひとまずクラスタリングを行い, 評価関数 $eval(C_n)$ が最大となるようなクラスタリング結果 C_n を採用しているといえる. このとき, $eval(C_n)$ の値が高いほど C_n の質もよいとみなしている.

以降では, クラスタリング結果を評価する 4 つの評価関数について説明する.

評価関数 1:クラスタ内類似度

クラスタリング結果 C のクラスタ内類似度 $intra(C)$ を式 (3.24) のように定義する. ただし, π_j は j 番目のクラスタ, $\vec{g}_j^{(n)}$ は π_j の重心ベクトル, $\vec{v}_i^{(n)}$ は π_j の要素である特徴ベクトル, N_j はクラスタ π_j の要素数とする.

$$intra(C) = \sum_{\pi_j \in C} \frac{1}{N_j} \sum_{\vec{v}_i^{(n)} \in \pi_j} sim(\vec{g}_j^{(n)}, \vec{v}_i^{(n)}) \tag{3.24}$$

まず, 個々の $\pi_j \in C$ について, クラスタの要素 $\vec{v}_i^{(n)}$ とクラスタの重心ベクトル $\vec{g}_j^{(n)}$ の類似度の平均値により評価値を算出する. そして, $intra(C)$ は, このように計算した個々の π_j の評価値の和と定義される. つまり, $intra(C)$ は個々のクラスタがその要素と類似しているほど C が良いクラスタリング結果であるという仮定に基づいている.

評価関数 2:クラスタ凝集度

クラスタリング結果 C のクラスタ凝集度 $coh(C)$ を式 (3.25) のように定義する.

$$coh(C) = \frac{intra(C)}{inter(C)} \tag{3.25}$$

$inter(C)$ はクラスタ間類似度を表し, 式 (3.26) のように定義する. ただし, π_j, π_k はクラスタ, $\vec{g}_j$, $\vec{g}_k$ はクラスタの重心ベクトル, N_{pairs} は $\pi_j \neq \pi_k$ であるような π_j と π_k の組み合わせの数を表す.

$$inter(C) = \frac{1}{N_{pairs}} \sum_{\pi_j, \pi_k \in C, \pi_j \neq \pi_k} sim(\vec{g}_j, \vec{g}_k) \tag{3.26}$$

つまり, $inter(C)$ はクラスタ同士の類似度の高さを表している. したがって, クラスタ内における要素間の類似度が高く, かつクラスタ間の類似度が低いほど, $coh(C)$ は高くなる.

これは, クラスタ内における要素は互いに類似度が高いほどよいクラスタであるが, クラスタ同士の類似度は低い方がクラスタリング結果全体としてはよいという仮定に基づいている.

評価関数 3:相対的クラスタ内類似度

一般に, 特徴ベクトルによって典型的な類似度の値は異なっている. 例えば, 4 章の評価実験では, トピックベクトルの 2 ベクトル間の類似度は 0.1 程度であるのに対し, LDA 拡張文脈ベクトルの 2 ベクトル間の類似度は 0.0003 程度であった. そのため, クラスタ内類似度に基づいた評価値 $intra(C_n)$ では常に類似度の大きい特徴ベクトルが選択されてしまうという問題がある. そこで, クラスタ内の要素間の類似度を相対的に評価し, ベクトル間の類似度の大きさに影響されにくい評価関数を定義する.

クラスタリング結果 C の相対的クラスタ内類似度 $rel\_intra(C)$ を式 (3.27) のように定義する. ただし, π_j は j 番目のクラスタ, $\vec{g}_j$ は π_j の重心ベクトル, $\vec{v}_i$ は π_j の要素である特徴ベクトル, N_j はクラスタ π_j の要素数とする.

$$rel\_intra(C) = \sum_{\pi_j \in C} \frac{1}{N_j} \sum_{\vec{v}_i \in \pi_j} \frac{sim(\vec{g}_j, \vec{v}_i)}{\max_{\vec{v}_i}(sim(\vec{g}_j, \vec{v}_i))} \tag{3.27}$$

まず, 式 (3.24) で定義した $intra$ と同様にクラスタ π_j 毎に評価値を計算する. π_j の評価値は, π_j の重心とその要素 $\vec{v}_i$ の類似度を $\max_{\vec{v}_i}(sim(\vec{g}_j, \vec{v}_i))$ との比を取ることで相対化したうえで, それらの平均と定義する. そして, $rel\_intra(C)$ は, このように計算した個々の π_j の評価値の和と定義する. つまり, $rel\_intra(C)$ は個々のクラスタにおいて, クラスタの重心とクラスタ内の要素との類似度がその最大値から外れているほど低くなり, 逆にクラスタ内の類似度が互いに似ているほど高くなるような評価関数である. 直感的には, $rel\_intra(C)$ は, クラスタ内類似度の分散のようなものを計算しているといえる.

評価関数 4:相対的クラスタ凝集度

クラスタリング結果 C の相対的クラスタ凝集度 $rel\_coh(C)$ を式 (3.28) のように定義する.

$$rel\_coh(C) = \frac{rel\_intra}{rel\_inter} \tag{3.28}$$

rel_inter は相対的クラスタ間類似度を表し, 式 (3.29) のように定義する. ただし, π_j は j 番目のクラスタ, $\vec{g}_j$ は j 番目のクラスタの重心ベクトル, k はクラスタの個数, $\vec{G}$ は $\vec{g}_j$ の各

π_j についての平均を表す.

$$rel\_inter(C) = \sum_{\pi_j \in C} \frac{sim(\vec{G}, \vec{g}_j)}{\max_{\vec{g}_j}(sim(\vec{G}, \vec{g}_j))} \tag{3.29}$$

$rel\_inter(C)$ は, クラスタ同士の類似度の高さを $\max_{\vec{g}_j}(sim(\vec{G}, \vec{g}_j))$ との比を取ることで相対化して測り, それらの平均値と定義する. 直感的には, $rel\_inter(C)$ はクラスタ間の類似度が似通っているほど高くなり, 逆にクラスタ間の類似度とその最大値との差が大きいほど低くなる. したがって, $rel\_coh(C)$ はクラスタ間の類似度の差が大きいほど, またクラスタ内の要素間の類似度の差が小さいほど, よいクラスタリング結果であるという仮定に基づいている.

第4章　評価実験

本章では, 提案手法の評価実験について述べる. まず, 3.1 節で提案した特徴ベクトルを用いて単語インスタンスのクラスタリングを行い, その結果を評価する. 次に, 3.2 節で示したクラスタリングアルゴリズムの比較を行う. また, 3.3 節で提案した複数の特徴ベクトルを組み合わせる手法の評価も行う.

4.1　テストデータ

ここではテストデータの作成方法について述べる.

まず, 実験に用いるコーパスとして, 毎日新聞と Yahoo! 知恵袋コーパスを用意した. 次に, それぞれのコーパスにおいて実験の対象となる単語を選定した. そして, 対象語のそれぞれに対し, そのインスタンスをコーパスからランダムに選択し, これをクラスタリングした. 選択されたインスタンスに対して人手で語義を付与し, 正解データを作成した. クラスタリングの結果は正解データをもとに評価した.

なお, 毎日新聞を対象とした実験と Yahoo! 知恵袋を対象とした実験では評価した特徴ベクトルやクラスタリング手法が異なる. これは評価実験を行う過程による. 本実験では, まず毎日新聞を用いた実験を行い, 次に Yahoo! 知恵袋コーパスを用いた実験を行った. そして, 毎日新聞における実験で全般的に評価の低い特徴ベクトルの実験は Yahoo! 知恵袋コーパスの実験では省いた. また, 一部の特徴ベクトルについては, 毎日新聞での結果を踏まえて新たに導入したため, Yahoo! 知恵袋コーパスにおける実験のみ行っている. それぞれのテストデータにおいて評価した特徴ベクトルならびにクラスタリング手法を表 4.1 にまとめる.

4.1.1　語義の基準

クラスタリングの結果を評価する際に, 各インスタンスに対して正しい語義を付与して正解データとする. 正解データの作成にあたっては, 岩波国語辞典の中分類を語義の基準とした. 例えば, 岩波国語辞典では「顔 (かお)」の語義は図 4.1 のように定義されている.

表 4.1: 各テストデータにおいて実験した特徴ベクトル

	新聞記事	Yahoo! 知恵袋
隣接ベクトル	o	o
文脈ベクトル	o	o
PLSI 拡張文脈ベクトル	o	x
LDA 拡張文脈ベクトル	o	o
連想ベクトル	o	o
TWP 連想ベクトル	o	x
WWP 連想ベクトル	o	x
TDP 連想ベクトル	o	x
TDL 連想ベクトル	o	x
トピックベクトル	x	o
類似度組み合わせ	o	o
特徴ベクトル選択	x	o

各語釈文の前の数値は, 右端が語義の小分類, 右から 2 つ目が語義の中分類を表しており, それぞれの語義が階層構造を持つ. この岩波国語辞典における小分類や中分類のような語義の粒度のうち, いずれが適切かということはタスクによって異なると考えられる. 本研究では仮に岩波国語辞典に掲載されている語義の中分類を基準とした.

4.1.2 毎日新聞のテストデータについて

ここでは毎日新聞のテストデータの作成方法について述べる. コーパスとして, 毎日新聞の 1991 年から 2004 年の 14 年分を用意した. 次に, コーパスにおいて出現数が多い順に単語を調査し, 出現数が少なくとも 70 件あり, 2 つ以上の語義で使われているような 10 単語を対象語として選定した. 得られた対象語のそれぞれに対し, 70 個のインスタンスをランダムに抽出した. こうして得られた対象語の各インスタンスに対して, 著者が人手で語義を付与した. なお, 語義は岩波国語辞典の中分類を基準とした.

これら 10 単語の詳細を表 4.2 に示す. 表において「第 1 語義」は最多の語義の割合,「第 2 語義」はその次に多い語義の割合を表す. 単語によって語義数の分布がかなり異なることがわかる. また, これら 10 単語に付与したそれぞれの語義の定義を付録 A に示す.

図 4.1: 岩波国語辞典における「顔」の語釈

7272-0-0-1-0 頭部の前面。目や鼻や口がある所。「―を洗う」「―から火が出る」（非常に恥ずかしい思いをする）「―にもみじを散らす」（婦人などが恥ずかしがって赤面した美しい様子の形容）「（人の）―に泥を塗る」（他人の体面を傷つける。↓（3）（イ））「―を合わせる」（会う。対面する）「―がそろう」（一同が皆集まる）「大きな―をする」（いばる）▽↓つら（面）
7272-0-0-2-0 顔（1）の様子。
7272-0-0-2-1 顔つき。表情。「変な―をする」
7272-0-0-2-2 容貌。「きれいな―」
7272-0-0-2-3 《接尾語的に》…の様子。「得意―」「―」「わけ知り―」▽人の態度に言う。
7272-0-0-3-0 顔（1）は人を見分ける目立つ部分だから、次のようにも使う。
7272-0-0-3-1 人によく知られていること。「―が広い」「彼はこの辺で―だ」（顔を見せるだけでも、様様の便宜をはかってもらえるほどだ）「―が売れる」（有名だ）「―を利かす」（有名さや勢力を利用して特別扱いをさせる）
7272-0-0-3-2 面目。体面。「―が立つ」「―をつぶす」
7272-0-0-3-3 『―がさす』人に見られては具合の悪い状態になる。＜関連＞おもて・つら・温顔・紅顔・厚顔・尊顔・童顔・笑顔・幼な顔・素顔・泣き顔・似顔・寝顔・真顔・横顔・赤ら顔・瓜ざね顔・恵比須顔・心得顔・したり顔・手柄顔・我が物顔・顔面・満面・面長・細面・顔立ち・しかめっ面・ふくれっ面・仏頂面・吠え面・横っ面・相貌・容貌・容色・美貌・人相・面相・形相・相好・プロフィール

4.1.3 Yahoo! 知恵袋のテストデータについて

ここでは Yahoo! 知恵袋のテストデータについて述べる．なお, Yahoo!知恵袋をコーパスとして用いたのは, 新聞記事のコーパスよりも日常的に使われている自然な日本語文が得られ, 多様な語義が出現すると考えられたためである．

まず, Yahoo! 知恵袋のテストデータの作成方法について述べる．Yahoo! 知恵袋コーパスは質問とベストアンサーの組 45,725 件からなるコーパスである．このコーパスにおいて, 出現数が多い順に単語を調査し, 出現数が少なくとも 100 件あり, 2 つ以上の語義で使われているような 23 単語を対象語として選定した．得られた対象語のそれぞれに対し, 100 個のインスタンスをランダムに抽出した．こうして得られた対象語の 100 個のインスタンスに対して, 作業者 2 名が 60 個ずつ語義を付与した．なお, 語義については, 岩波国語辞典で対象語の見出しを引いて, 語義の中分類の中に該当すると思われる語義があればそのラベルを振った．該当するものが存在しなければ, 作業者 2 名が中分類に相当すると思われる粒度で語義を新たに定義して, そのラベルを付与した．

100 個のうち 20 個のインスタンスについては, 2 名の作業者がそれぞれ独立に語義を付与した．その 20 個のインスタンスについて語義の一致率を調べた結果を表 4.3 に示す．表 4.3 は, 単語によっては語義の判定が人によってかなり異なることを示している．例えば,「場合」という単語では 20 インスタンス中の語義の一致率が 0.5 となっている一方,「ア

表 4.2: 新聞記事における対象単語の語義の数と分布

	語義数	第 1 語義	第 2 語義
頭	5	51.4%	30.0%
場所	2	81.4%	18.6%
ケース	3	87.1%	11.4%
核	2	82.9%	17.1%
記録	2	74.3%	25.7%
目	5	58.6%	35.7%
ポイント	4	42.9%	37.1%
線	4	37.1%	35.7%
自然	3	61.4%	37.1%
運動	3	75.7%	20.0%

具体的な語義の定義については付録 A に記載した

ルバム」という単語では 1.0 である.

23 個の対象語のそれぞれの 100 インスタンスに付与された語義の分布とそのエントロピーを表 4.4 に示す. また, 付与された語義の定義を付録 B に示す. 表 4.4 において, エ

表 4.3: Yahoo! 知恵袋における対象語の語義の一致率

モデル	ネタ	カバー	ウイルス	ソース	肉	サービス	地方
0.90	0.80	0.80	1.00	0.95	0.95	0.30	0.65
アルバム	コード	自分	場合	時間	意味	電話	一緒
1.00	1.00	1.00	0.50	0.65	0.80	0.85	1.00
目	以前	代	顔	系	郵便	反応	
0.85	0.95	0.90	0.75	0.85	0.70	0.80	

平均一致率: 0.82

ントロピーとは後に述べるクラスタ内の語義の分布のエントロピーではなく, 語義全体おける出現確率のエントロピーである. これは式 (4.1) のように定義される, ただし, 式 (4.1) において s_i は i 番目の語義を表す.

$$Entropy = -\sum_{s_i} P(s_i) \log P(s_i) \tag{4.1}$$

エントロピーが高ければ高いほど, 複数の語義が均等に現れることを表す. つまり, エントロピーが高い単語はそれだけクラスタリングによる語義の自動識別が難しいといえる.

表 4.4: Yahoo! 知恵袋における対象語の語義の分布

	語義の分布						Entropy
	1	2	3	4	5	6	
モデル	45	44	7	2	2		1.06
ネタ	58	21	14	6	1		1.13
カバー	44	22	22	12			1.28
ウイルス	85	15					0.42
ソース	74	18	8	1			0.78
肉	96	2	1	1			0.21
サービス	67	18	9	6	1		1.01
地方	63	37					0.66
アルバム	91	9					0.30
コード	46	34	19	1			1.09
自分	64	35	1				0.70
場合	55	41	3	1			0.85
時間	58	30	9	3			1.00
意味	69	20	11				0.82
電話	68	30	2				0.70
一緒	82	18					0.47
目	45	44	5	2	2	2	1.11
以前	87	13					0.39
代	47	47	3	1	1	1	0.95
顔	60	38	2				0.75
系	86	7	6	1			0.53
郵便	90	7	3				0.39
反応	78	12	10				0.68

具体的な語義の定義については付録 B に記載した

例えば「以前」という単語ではエントロピーは低くなっており, 語義の自動判別は簡単であるといえる. 一方, 「カバー」という単語では語義が 4 つ存在している. そして, 4 つの語義が比較的均等に出現していることを反映して, エントロピーは高くなっており, 語義の自動判別は難しいといえる.

4.2 評価基準

本節では, 対象語のインスタンスを同じ語義を持つものでまとめたクラスタリング結果の評価指標について述べる. なお, 評価指標の説明には表 4.5 に示すような記号を用いる.

表 4.5: 評価指標の説明に用いる記号とその意味

記号	意味
s_i	i 番目の語義
π_j	j 番目のクラスタ
$W(\pi_j, s_i)$	クラスタ π_j の中で語義が s_i であるようなインスタンスの数
N_{π_j}	クラスタ π_j のインスタンスの総数
N_{s_i}	語義 s_i を持つインスタンスの総数
N	インスタンスの総数

Purity

Purity はクラスタ内の語義の純度の指標であり, 式 (4.2) のように定義される.

$$Purity = \frac{1}{N}\sum_{\pi_j}\max_{s_i}(W(\pi_j, s_i)) \tag{4.2}$$

つまり, Purity は各クラスタ π_j において最多の語義を持つインスタンスの数を求め, その値をインスタンスの総数 N で割ったものと定義する. 直感的には, 大きなクラスタにおいて語義の純度が高ければ Purity は高くなる.

Entropy

Entropy はクラスタ内の語義の出現確率のエントロピーの指標であり, 式 (4.3) のように定義される.

$$Entropy = \frac{1}{N}\sum_{\pi_j} N_{\pi_j}\{-\sum_{s_i} P(s_i|\pi_j)\log P(s_i|\pi_j)\} \tag{4.3}$$

ここで, $P(s_i|\pi_j)$ はクラスタ π_j の中に語義 s_i を持つインスタンスが存在する確率であり, 式 (4.4) で求める.

$$P(s_i|\pi_j) = \frac{W(\pi_j, s_i)}{N_{\pi_j}} \tag{4.4}$$

つまり, Entropy は各クラスタ π_j において語義の出現確率のエントロピーを計算し, それをクラスタの要素数 N_{π_j} で重み付き平均したものと定義する. 直感的には, 大きなクラス

タにおいて語義の曖昧性が低い, つまりある語義の出現確率が突出して高いほど, Entropy は低くなる.

Inverse Purity

Inverse Purity は情報検索における再現率と似た指標であり, 式 (4.5) のように定義される.

$$InversePurity = \sum_{s_i} \frac{N_{s_i}}{N} \times \frac{\max_{\pi_j}(W(\pi_j, s_i))}{N_{s_i}} \tag{4.5}$$

つまり, Inverse Purity は各語義 s_i 毎に再現率を計算し, それを語義数 N_{s_i} により重み付き平均したものと定義する. 直感的には, クラスタリング対象となる単語インスタンスにおいて数多く出現している語義が少数のクラスタにまとまって出現しているほど, Inverse Purity は高くなる.

評価基準のまとめ

本項で述べたクラスタリングの評価基準についてまとめる. Purity は, その値が高いほど, 1 つのクラスタ内には同じような語義を持つインスタンスがまとまっていると解釈できる. Entropy は, その値が低いほど, 各クラスタにおける語義の出現確率のエントロピーが低いと解釈できる. Inverse Purity は, その値が高いほど, 同じ語義を持つインスタンスが 1 つのクラスタにまとめられていると解釈できる. つまり, Purity や Inverse Purity は値が高いほどクラスタリング結果が良く, 逆に Entropy は値が低いほどクラスタリング結果が良い.

4.3 毎日新聞を対象とした実験

本節では, 毎日新聞のテストデータを対象に, 単独の特徴ベクトルでクラスタリングする手法, 複数の特徴ベクトルの類似度を組み合わせる手法について評価する.

4.3.1 実験方法

本節の実験ではクラスタリング手法として k-means 法とセントロイド法を用いた. k-means 法については 70 個のインスタンスのみを対象とした実験では評価が非常に低かったため, 語義タグを付与していない 630 個のインスタンスをランダムサンプリングして追

加した計 700 個のインスタンスを対象にクラスタリングを行っている. ただし, クラスタリングの評価は語義タグを付与した 70 個のインスタンスのみで行う. 以後, 特に断りのない限り, 700 インスタンスをクラスタリングした結果を示す. クラスタ数は k-means では 10, セントロイド法では 12 とした. セントロイド法でクラスタ数を 10 ではなく 12 とした理由は, 毎日新聞のテストデータでは対象語によっては非常にスパースな特徴ベクトルが作られることがあり, 既定の数のクラスタが作成される前に全ての組のクラスタの重心ベクトル間の類似度が 0 となり, クラスタリングが続行できなかったことによる. クラスタ数を 12 にすると, 全ての対象語について 12 個のクラスタを作成することができた.

4.3.2 隣接, 文脈, 連想ベクトルによるクラスタリングの評価

隣接ベクトル (3.1.1 項, ウインドウ幅 $w = 2$), 文脈ベクトル (4.3.4 項), 連想ベクトル (4.3.3 項) について k-means 法とセントロイド法を適用してクラスタリングを行った. クラスタリング結果の Purity と Entropy を表 4.6 および表 4.7 に示す. 表中の k-means1, k-means2 はそれぞれ 70 インスタンス, 700 インスタンスを対象に k-means を適用した結果を示す. また, k-means は初期クラスタによってクラスタリング結果が異なるため, 10 回試行した平均の評価値を掲載している.

表 4.6: 連想・文脈・隣接のクラスタリング結果 (Purity)

	k-means1			k-means2			セントロイド法		
	連想	文脈	隣接	連想	文脈	隣接	連想	文脈	隣接
頭	**0.628**	0.580	0.577	**0.626**	0.573	0.579	0.590	0.590	**0.660**
場所	**0.863**	0.813	0.819	0.861	0.863	**0.911**	0.890	0.830	**0.940**
ケース	**0.884**	0.871	0.872	0.866	0.876	**0.893**	0.900	0.930	**0.940**
核	0.850	0.831	**0.867**	0.842	0.837	**0.946**	0.910	0.930	**0.990**
記録	**0.840**	0.751	0.776	**0.833**	0.808	0.777	0.810	**0.860**	0.840
目	0.732	0.639	**0.796**	0.716	0.807	**0.904**	0.730	0.700	**0.960**
ポイント	**0.714**	0.556	0.670	0.737	0.666	**0.805**	0.630	0.670	**0.870**
線	**0.713**	0.537	0.567	**0.714**	0.639	0.571	**0.640**	**0.640**	0.530
自然	**0.815**	0.703	0.767	0.819	0.740	**0.855**	**0.840**	0.740	0.800
運動	**0.805**	0.759	0.764	**0.848**	0.770	0.833	**0.860**	0.830	0.840
avg.	**0.784**	0.704	0.748	0.786	0.758	**0.807**	0.780	0.772	**0.837**

太字はそれぞれのクラスタリング手法において Purity が最も高い特徴ベクトル

まず Purity の実験結果に注目する. k-means1 では, 連想ベクトルの結果がよく, 次に隣接ベクトルが良い. k-means2 では, 連想ベクトルの結果が k-means1 よりさらに良くな

表 4.7: 連想・文脈・隣接のクラスタリング結果 (Entropy)

	k-means1			k-means2			セントロイド法		
	連想	文脈	隣接	連想	文脈	隣接	連想	文脈	隣接
頭	**0.747**	0.887	0.871	**0.777**	0.890	0.858	0.820	0.890	**0.690**
場所	**0.303**	0.410	0.380	0.297	0.284	**0.225**	0.290	0.340	**0.170**
ケース	**0.286**	0.319	0.312	0.341	0.301	**0.228**	0.340	0.240	**0.180**
核	0.290	0.364	**0.281**	0.271	0.349	**0.132**	0.190	0.230	**0.040**
記録	**0.308**	0.504	0.465	**0.307**	0.387	0.449	**0.340**	0.380	0.370
目	0.588	0.723	**0.458**	0.670	0.454	**0.262**	0.710	0.650	**0.170**
ポイント	**0.605**	0.928	0.713	0.594	0.724	**0.453**	0.650	0.780	**0.320**
線	**0.641**	0.925	0.892	**0.677**	0.763	0.875	0.730	**0.720**	0.920
自然	**0.403**	0.574	0.490	0.419	0.529	**0.344**	**0.400**	0.490	0.400
運動	**0.414**	0.546	0.521	**0.334**	0.529	0.415	0.420	0.430	**0.400**
avg.	**0.458**	0.618	0.538	0.469	0.521	**0.424**	0.489	0.515	**0.366**

太字はそれぞれのクラスタリング手法において Entropy が最も低い特徴ベクトル

り, 他の特徴ベクトルによるクラスタリングの質も改善した. ただし, k-means2 では隣接ベクトルの結果が最も良く, 連想ベクトルも上回った. セントロイド法では, k-means2 と同様に隣接ベクトルの結果が最も良く, 全体的にみて k-means2 より結果が良い. 表 4.7 の Entropy についても同様の傾向が見られた.

4.3.3 連想ベクトルの評価

本項では, 3.1.5 項で述べた TWP 連想ベクトル, WWP 連想ベクトル, TDP 連想ベクトル, TDL 連想ベクトルについて実験する. また, 連想ベクトル (4.3.3 項) との比較も行う.

まず, TWP 連想ベクトルおよび WWP 連想ベクトルについて実験する. TWP・WWP 連想ベクトルの作成の際は, 隠れ変数の数 n を 256, 500, 900 と変化させて PLSI を適用した. なお, クラスタリング手法は k-means 法のみを用いた. 結果を表 4.8 に示す.

対象語によって適切なトピック数 n が異なることが表から読み取れる. 例えば, WWP 連想ベクトルに注目すると,「頭」では n が大きいほど良い結果が出ているが,「ケース」や「目」といった単語では n=500 のときの結果が最良である.

また, 表 4.6 に示した連想ベクトルを k-means 法でクラスタリングした結果と比較すると, WWP 連想ベクトルでは n=900 のときの「場所」と n=500 のときの「目」で Purity が改善したほかは悪化している. 同様に TWP 連想ベクトルでは n=500 または 900 のときの「場所」で Purity が改善しているほかは悪化している. 10 語の対象語の平均の結果も

表 4.8: WWP 連想ベクトル, TWP 連想ベクトルによるクラスタリング結果 (Purity)

	WWP 連想			TWP 連想		
n	256	500	900	256	500	900
頭	0.584	0.613	**0.619**	**0.600**	**0.600**	0.598
場所	0.874	0.874	**0.900**	0.869	0.896	**0.898**
ケース	0.861	**0.871**	0.857	0.864	**0.872**	0.863
核	0.826	**0.828**	**0.828**	0.821	0.827	**0.835**
記録	0.801	0.787	**0.810**	**0.786**	0.780	0.780
目	0.712	**0.729**	0.704	0.686	**0.697**	0.681
ポイント	0.602	0.610	**0.627**	0.639	0.649	**0.663**
線	0.648	0.638	**0.678**	0.648	**0.650**	0.649
自然	0.736	0.729	**0.774**	**0.761**	0.726	0.726
運動	0.765	**0.768**	0.766	**0.774**	0.766	0.768
avg.	0.741	0.745	**0.756**	0.745	**0.746**	**0.746**

太字はそれぞれの特徴ベクトルにおいて Purity が最も高い n

これを反映して, 連想ベクトルの Purity は 0.786 であったのに対し, WWP 連想ベクトルでは 0.756, TWP 連想ベクトルでは 0.746 が最大となり, いずれも PLSI を利用しないほうが良いという結果になった.

次に, TDP 連想ベクトルおよび TDL 連想ベクトルと, 連想ベクトルの idf による重み付けについて実験する. TWP 連想ベクトルや WWP 連想ベクトルではクラスタリング性能の向上は見られなかった. そこで, トピックを学習する共起行列を単語×単語の行列から単語×文書の行列に変更した TDP 連想ベクトルや, PLSI のかわりに LDA を用いた TDL 連想ベクトルを用いる実験を行った. トピックを学習するためのデータは, 毎日新聞 1991 年分のコーパス (480,000 段落, 10,000 単語) より作成した文書-単語の共起行列を用いる. なお, ここでは, 毎日新聞の段落を文書の単位とする.

さらに, Schütze が Word Vector に idf による重みを与えている [10] ことを参考に, インスタンス w_i の周辺に含まれる c_j の共起ベクトル $\vec{o}(c_j)$ の和で連想ベクトルを作成する際に, inverse document frequency(idf) の重み付けを試みた. 単語 c_j の idf は式 (4.6) のように定義される. ただし, N はコーパスに含まれる文書数, N_{c_j} は単語 c_j が含まれる文書の数を表す.

$$idf(c_j) = \log \frac{N}{N_{c_j}} \tag{4.6}$$

つまり, idf は c_j を含む文書数が少ないほど高くなる関数である. idf の重みを考慮した

連想ベクトルを式 (4.7) のように定義する.

$$\vec{a}_i = \sum_{c_j \in context} idf(c_j)\ \vec{o}(c_j) \tag{4.7}$$

つまり, idf の重みを考慮した連想ベクトルの要素は, idf の高い周辺語 c_j に関する共起ベクトルにバイアスがかかっている. これは, 限られた文書に出現するような周辺語 c_j がインスタンス w_i と w_j に出現していれば, それらは同じ意味で使われているであろうという考えに基づいている. TDP 連想ベクトル, TDL 連想ベクトルについても同様に idf による重み付けを行う.

以上の条件で実験を行った結果を表 4.9, 表 4.10 に示す.

表 4.9: idf の重みと TDP・TDL 連想ベクトルによるクラスタリング (Purity)

重み	なし			idf		
	連想	TDP 連想	TDL 連想	連想	TDP 連想	TDL 連想
頭	**0.633**	0.618	0.609	**0.627**	0.599	0.597
場所	0.856	0.834	**0.869**	0.850	0.850	**0.874**
ケース	**0.899**	0.870	0.883	**0.903**	0.870	0.882
核	0.841	0.849	**0.876**	0.840	0.846	**0.865**
記録	**0.834**	0.781	0.748	**0.843**	0.788	0.752
目	0.761	0.788	**0.865**	0.766	0.697	**0.843**
ポイント	**0.727**	0.629	0.668	**0.719**	0.606	0.645
線	**0.714**	0.619	0.585	**0.699**	0.595	0.614
自然	**0.782**	0.715	0.731	**0.772**	0.734	0.705
運動	**0.795**	0.794	0.775	**0.796**	0.779	0.758
avg	**0.784**	0.750	0.761	**0.781**	0.736	0.753

太字は重みあり, 重みなしの場合におけるそれぞれの Purity 最大値

まず, idf の重みを用いない場合の結果を考察する. 連想ベクトルを基準として TDP 連想ベクトルの Purity を見ると, 「核」と「目」でそれぞれ 0.009, 0.027 の改善が見られる他は 0.01 から 0.10 低下している. それを受けて, 10 単語の平均も 0.034 低下している. 一方, Entropy はいずれの単語でも上昇している. したがって, 全体的にみれば TDP 連想ベクトルを用いることは有効ではないが, 単語によっては Purity 向上が見込める.

連想ベクトルを基準として TDL 連想ベクトルの Purity を見ると, 「頭」は 0.034, 「場所」は 0.013, 「核」は 0.035, 「目」は 0.104 の向上が見られるが, 他の対象語では 0.016 から 0.086 低下している. 平均では 0.023 の低下である. Entropy を比較すると, 「目」で 0.116 低下している他は総じて上昇している. 平均で見ても, やはり 0.075 の上昇であり全

表 4.10: idf の重みと TDP・TDL 連想ベクトルによるクラスタリング (Entropy)

重み	なし			idf		
	連想	TDP 連想	TDL 連想	連想	TDP 連想	TDL 連想
頭	**0.807**	0.818	0.821	**0.825**	0.864	0.857
場所	**0.304**	0.356	0.326	**0.308**	0.318	0.320
ケース	**0.269**	0.323	0.332	**0.252**	0.313	0.339
核	**0.316**	0.358	**0.316**	0.322	0.349	**0.294**
記録	**0.292**	0.407	0.469	**0.292**	0.422	0.463
目	0.465	0.509	**0.349**	0.458	0.639	**0.411**
ポイント	**0.599**	0.752	0.744	**0.611**	0.800	0.777
線	**0.617**	0.764	0.871	**0.633**	0.816	0.818
自然	**0.420**	0.548	0.549	**0.426**	0.504	0.598
運動	**0.444**	0.450	0.508	**0.461**	0.484	0.548
avg	**0.453**	0.528	0.528	**0.459**	0.551	0.542

太字は重みあり, 重みなしの場合におけるそれぞれの Entropy 最低値

体的に良いとはいえない. 「核」および「目」は TDP 連想ベクトルでも改善していたが, LDA を用いた TDL 連想ベクトルを用いた場合はさらに良い. しかしながら, 「記録」などは PLSI を用いた TDP 連想ベクトルよりもさらに悪化している.

次に, 単語ベクトルの重みを idf にした場合の結果を考察する. idf を重みにすることで Purity が悪化した対象語は「頭」「場所」「核」「ポイント」「線」「自然」で, それぞれ 0.001 から 0.015 低下している. また, Purity が改善された単語は「ケース」「記録」「目」「運動」で, それぞれ 0.001 から 0.009 上昇している. 平均では 0.003 の低下である. Entropy が改善された単語は「ケース」「目」で, それぞれ 0.017 と 0.007 低下している. その他の単語では 0.04 から 0.018 上昇している. 平均的に見れば Purity も Entropy も悪化している.

TDP 連想ベクトルを用いた場合は, Purity が特に改善したのは「場所」「記録」「自然」で, それぞれ 0.016, 0.007, 0.019 向上しているが, その他の単語では低下し, 平均的には悪化している. Entropy で見ても, 特に「目」では 0.130 の上昇となっており全ての組み合わせ中最も idf と相性が悪い. 平均的にも悪化している. TDL 連想ベクトルを用いた場合も, Purity の改善が「線」「場所」「記録」で見られた他は悪化しており, Entropy の改善も「線」「場所」「記録」で見られた他は悪化している. しかしながら, 「線」については Purity で 0.029, Entropy で 0.053 改善しており, 有効であるといえる.

以上の結果から, 連想ベクトルの拡張や idf による重み付けは対象語によって有効性が異なるが, 全体的にはクラスタリングの結果を向上させることはなかった.

4.3.4 文脈ベクトルの評価

前項では, 連想ベクトルの PLSI や LDA によって学習されたトピックを利用した改良が単語によっては効果があることを示した. 本項では, 3.1.2 項で説明した文脈ベクトルを PLSI や LDA で得られたトピック関連語で拡張することによる効果を確認する. ここで, n は PLSI の隠れ変数の数, m は 1 つの隠れ変数 (トピック) に追加する関連語の数とする. そして, パラメータ n, m を変えた場合のクラスタリングの変化に注目する.

まず, PLSI 拡張文脈ベクトルによるクラスタリングの結果を表 4.11, 4.12 に示す.

表 4.11: PLSI 拡張文脈ベクトルによるクラスタリング (Purity)

n	100			150			200		
m	100	200	300	100	200	300	100	200	300
頭	0.598	0.607	0.612	0.614	0.590	0.606	0.622	**0.631**	0.609
場所	0.839	**0.866**	0.829	0.848	0.851	0.853	0.810	0.853	0.849
ケース	0.883	0.869	0.868	0.870	0.877	0.878	0.884	0.889	**0.892**
核	0.837	0.833	0.851	0.845	0.849	0.855	0.837	**0.859**	0.851
記録	0.778	0.768	0.768	0.758	0.771	**0.806**	0.775	0.781	0.797
目	0.659	**0.718**	0.700	0.672	0.659	0.686	0.663	0.683	0.694
ポイント	0.624	0.626	0.605	0.584	0.561	0.587	0.593	0.639	**0.670**
線	0.602	0.595	0.595	0.621	**0.631**	0.588	0.603	0.604	0.588
自然	0.708	0.674	0.718	0.719	**0.758**	0.733	0.728	0.722	0.711
運動	0.772	0.760	0.775	0.770	**0.777**	0.766	0.765	0.765	0.756
avg.	0.730	0.732	0.732	0.730	0.732	0.736	0.728	**0.743**	0.742

太字は Purity の最大値

ここでは, $(n, m) = (x, y)$ でパラメタ $n = x$, $m = y$ の組み合わせを表すものとして説明する. まず, Purity は $(n, m) = (200, 200)$ の場合が最大となっている. また n, m の違いによる変化をみると, Purity は $n = 200$ の場合を除いて m が大きくなるに従い改善しており, 同様に $m = 100$ の場合を除けば n の増加にともない改善している. 対象語毎にみて Purity が最大となった単語数が最大であるのは $(n, m) = (150, 200)$ であり, 「線」「自然」「運動」の 3 つの単語に対して改善がみられた. したがって, 平均的に Purity が高いからといって, どんな対象語においてもそのパラメータが良いとは限らない. しかしながら, n や m を増やすほど Purity が高くなるという傾向は見られる.

Entropy は $(n, m) = (200, 200)$ の場合が最低値となっている. m による変化をみると $n = 100$ のときは $m = 100$ が, $n = 200, 300$ のときは $m = 200$ が最低値を示している. また, 次点でみると $n = 100$ のときは $m = 300$, $n = 200, 300$ のときは $m = 300$ の Entropy が

表 4.12: PLSI 拡張文脈ベクトルによるクラスタリング (Entropy)

n	100			150			200		
m	100	200	300	100	200	300	100	200	300
頭	0.819	0.846	0.840	**0.770**	0.866	0.860	0.837	0.804	0.885
場所	0.362	0.330	0.378	0.340	0.339	0.338	0.396	**0.308**	0.341
ケース	0.292	0.322	0.308	0.319	0.313	0.317	0.288	0.281	**0.279**
核	0.354	0.372	0.324	0.300	0.286	0.298	0.368	**0.277**	0.298
記録	0.425	0.450	0.430	0.460	0.427	**0.391**	0.410	0.442	0.403
目	0.658	**0.596**	**0.596**	0.676	0.656	0.607	0.665	0.666	0.616
ポイント	0.784	0.790	0.811	0.848	0.855	0.831	0.831	0.749	**0.718**
線	0.805	0.790	0.829	0.790	**0.726**	0.802	0.807	0.795	0.825
自然	0.548	0.617	0.567	0.552	**0.493**	0.555	0.511	0.548	0.583
運動	0.484	0.521	0.489	0.501	**0.462**	0.506	0.507	0.523	0.515
avg.	0.553	0.563	0.557	0.556	0.542	0.550	0.562	**0.539**	0.546

太字は Entropy の最低値

低い. 対象語毎にみて Entropy が最低となった単語数が最大であるのは $(n, m) = (150, 200)$ の 3 であり, 10 単語で平均的によかった $(200, 200)$ の場合は 2 つの単語について改善が見られた. また, いずれの対象語でも Entropy が最低値を示すような最適なパラメータというものは見受けられない. したがって, Entropy の観点からすると, $n = 200$, m は 200 または 300 が妥当である.

以上の通り, いずれの対象語においても適切であるような n, m の値は発見できなかった. しかしながら, Purity と Entropy が平均的に高くなるのは n および m を大きく設定した場合である.

次に, LDA 拡張文脈ベクトルによるクラスタリングの結果を表 4.13, 4.14 に示す.

Purity は $(n, m) = (200, 300)$ の場合が最大となっている. また n, m の違いによる変化をみると, Purity は $n = 100$ の場合は m が大きくなるに従い低下しており, $n = 150$ の場合は $m = 150$ が最大となっている. そして $n = 200$ の場合は m が大きくなるに従い上昇している. 対象語毎にみると $(200, 300)$ の場合でも 3 つの対象語で最大となるだけであり, 全ての単語に対して最適なパラメータであるとはいえないが, 平均的な有効性を考慮すれば $(200, 300)$ が最善である.

Entropy は $(n, m) = (200, 300)$ の場合が最低値となっている. m による変化をみると, $n = 100$ のときは m の増加に伴い低下, $n = 150$ のときは $m = 150$ で最大, $n = 200$ のときは m の増加に伴い上昇して $m = 300$ で全体の最低値を示している. 改善・悪化の傾向

表 4.13: LDA 拡張文脈ベクトルによるクラスタリング (Purity)

n	100			150			200		
m	100	200	300	100	200	300	100	200	300
頭	0.621	0.590	0.571	0.624	0.629	0.613	0.585	0.626	**0.685**
場所	0.869	0.828	0.836	0.864	0.864	0.845	0.855	**0.890**	0.846
ケース	0.877	0.878	0.875	**0.916**	0.893	0.889	0.902	0.909	0.889
核	0.842	0.862	**0.874**	0.839	0.846	0.837	0.856	0.852	0.869
記録	0.817	0.831	0.808	0.844	0.790	0.821	0.810	0.819	**0.860**
目	0.758	0.745	0.794	0.844	**0.866**	0.831	0.843	0.825	0.818
ポイント	0.699	0.696	0.640	0.688	0.733	**0.734**	0.643	0.652	0.682
線	**0.689**	0.655	0.667	0.606	0.651	0.647	0.602	0.654	0.675
自然	**0.786**	0.774	0.736	0.766	0.739	0.755	0.715	0.689	0.717
運動	0.785	0.762	0.782	0.770	0.771	0.772	0.765	0.780	**0.798**
avg.	0.774	0.762	0.758	0.776	0.778	0.774	0.758	0.770	**0.784**

太字は Purity の最大値

は Purity と類似している. 対象語毎にみても $(n, m) = (200, 300)$ が 4 つの対象語で最低となるだけであり, $(n, m) = (200, 300)$ が全ての単語に対し最適なパラメータであるとはいえない.

以上の通り, PLSI の場合と同様に全ての対象語について最適な n, m の値は発見できなかった. しかしながら, 平均的な Purity と Entropy が共に最大となるのは $(n, m) = (200, 300)$ の場合であった.

表 4.14: LDA 拡張文脈ベクトルによるクラスタリング (Entropy)

n	100			150			200		
m	100	200	300	100	200	300	100	200	300
頭	0.807	0.815	0.813	0.813	0.733	0.771	0.867	0.776	**0.661**
場所	0.283	0.352	0.364	0.272	0.275	0.305	0.312	**0.256**	0.335
ケース	0.290	0.273	0.315	**0.229**	0.229	0.276	0.250	0.248	0.293
核	0.328	0.305	0.285	0.342	0.345	0.359	0.325	0.279	**0.267**
記録	0.339	0.325	0.392	0.320	0.391	0.342	0.374	0.366	**0.281**
目	0.495	0.542	0.493	0.392	**0.350**	0.384	0.412	0.388	0.409
ポイント	0.650	0.643	0.748	0.647	0.602	**0.581**	0.755	0.711	0.689
線	**0.651**	0.702	0.696	0.762	0.723	0.748	0.783	0.732	0.698
自然	**0.447**	0.475	0.531	0.494	0.482	0.496	0.567	0.580	0.556
運動	0.441	0.489	0.475	0.496	0.512	0.500	0.528	0.448	**0.415**
avg.	0.473	0.492	0.511	0.477	0.464	0.476	0.517	0.478	**0.460**

太字は Entropy の最低値

4.3.5 連想・文脈・隣接ベクトルの類似度の組み合わせによるクラスタリングの評価

4.3.2 項の実験で, 対象語によってクラスタリングに有効な素性が異なることを示した. そこで, 異なる特徴ベクトルによる類似度の重み付き和を取ることで, その対象語に関係のないベクトルの影響を排除することを意図して, 連想・文脈・隣接ベクトルの類似度を 3.3.1 項で述べた方法で組み合わせてクラスタリングを行った. すなわち, 3 つのベクトルによる類似度の重み付き和でインスタンス間の類似度を求め, クラスタリングを行った. 類似度の重みは, 連想ベクトルに対する重みを 0.2, 文脈・隣接ベクトルに対する重みをそれぞれ 0.4 と設定した. その結果を表 4.15 に示す.

表 4.15: 連想・文脈・隣接の類似度組み合わせによるクラスタリング結果

	k-means1		k-means2		セントロイド法	
	purity	entropy	purity	entropy	purity	entropy
頭	0.590	0.869	0.594	0.807	**0.670**	**0.680**
場所	0.815	0.396	0.911	**0.203**	0.930	0.210
ケース	0.875	0.302	0.880	0.276	**0.960**	**0.170**
核	0.829	0.378	**0.948**	**0.124**	0.990	0.040
記録	0.763	0.477	0.778	0.413	**0.900**	**0.270**
目	0.645	0.703	**0.929**	**0.225**	**0.970**	**0.130**
ポイント	0.534	0.925	**0.825**	**0.407**	0.870	0.360
線	0.536	0.918	0.676	0.702	0.530	0.860
自然	0.735	0.537	0.838	0.362	0.700	0.520
運動	0.757	0.554	**0.856**	**0.367**	0.830	0.460
avg.	0.708	0.606	**0.824**	0.389	0.835	0.370

太字は連想・文脈・隣接単体より改善したもの

表 4.15 より, k-means2 とセントロイド法については, 3 つの特徴ベクトルを組み合わせると, 特徴ベクトルを単独で利用した場合と比較して, Purity は向上し, Entropy は低下することがわかる. また, 単独の特徴ベクトルによりクラスタリングを行った場合と同様に, k-means 法よりセントロイド法のクラスタリングの方が評価が良い. k-means 法についてはクラスタリング時に利用する単語インスタンスの数を 700 からさらに増やすことで精度が向上する可能性もある.

4.3.6 拡張文脈ベクトルと隣接ベクトルの類似度組み合わせによるクラスタリングの評価

4.3.4 項では, PLSI 拡張文脈ベクトル・LDA 拡張文脈ベクトルのパラメータの隠れ変数の数 n は 200, ベクトルに追加する関連語の数 m は 300 が適切であることを示した. 関連語を考慮しない文脈ベクトルによるクラスタリングの Purity は 0.76(表 4.6), Entropy は 0.52 であった (4.7) が, 関連語を追加した拡張文脈ベクトルはこれを上回った.

本項では, 関連語を追加した拡張文脈ベクトルと隣接ベクトルの類似度の組み合わせによるクラスタリングを評価する. 比較対象として, 関連語を追加しない文脈ベクトルと隣接ベクトルの組み合わせについても調査した. また, パラメータは $(n, m) = (200, 300)$ とした. 実験結果を表 4.16, 表 4.17 に示す. なお, 隣接ベクトルによるクラスタリングの Purity は 0.81(表 4.6), Entropy は 0.42 であった (表 4.7).

実験によると Purity, Entropy の両指標において関連語を追加しない文脈ベクトルと隣接ベクトルの組み合わせが最も良いクラスタリング結果になった. 関連語を追加した文脈ベクトルとの組み合わせでは Entropy が隣接ベクトル単独の結果より悪くなっている場合もある. しかしながら, 例えば「自然」では LDA 拡張文脈ベクトルと隣接ベクトルの組み合わせが最も評価がよい. 対象語によっては関連語を追加した拡張文脈ベクトルとの組み合わせが良いものの, 単独で精度の高い特徴ベクトルを組み合わせることが良い結果になるとは限らないことがわかる.

表 4.16: PLSI 拡張・LDA 拡張文脈ベクトル文脈と隣接ベクトルの類似度組み合わせ (Purity)

組み合わせ	PLSI 拡張文脈 + 隣接			LDA 拡張文脈 + 隣接			文脈 + 隣接		
重み	0.7,0.3	0.5,0.5	0.3,0.7	0.7,0.3	0.5,0.5	0.3,0.7	0.7,0.3	0.5,0.5	0.3,0.7
頭	0.605	0.588	0.587	0.595	0.596	0.616	**0.645**	0.596	0.612
場所	0.882	0.887	0.889	0.877	0.883	0.867	**0.891**	0.868	**0.891**
ケース	0.889	0.876	0.891	0.875	**0.892**	0.882	0.884	0.888	0.880
核	0.947	**0.966**	0.941	0.936	0.950	0.959	0.931	0.951	0.938
記録	0.773	0.792	0.795	0.783	0.786	0.788	**0.809**	0.804	0.791
目	0.908	0.924	0.912	0.895	0.922	0.923	0.926	0.910	**0.930**
ポイント	0.829	0.824	0.830	0.827	0.823	0.826	**0.836**	0.826	0.823
線	0.607	0.588	0.585	0.595	0.616	0.613	**0.657**	0.624	0.609
自然	0.834	0.825	0.832	0.831	0.836	**0.844**	0.804	0.821	0.840
運動	**0.834**	0.828	0.820	0.831	0.819	0.822	0.802	0.820	0.822
avg.	0.811	0.810	0.808	0.804	0.812	0.814	**0.819**	0.811	0.814

太字は Purity の最大値

表 4.17: PLSI 拡張・LDA 拡張文脈ベクトルと隣接ベクトルの類似度組み合わせ (Entropy)

組み合わせ	PLSI 拡張文脈 + 隣接			LDA 拡張文脈 + 隣接			文脈 + 隣接		
重み	0.7,0.3	0.5,0.5	0.3,0.7	0.7,0.3	0.5,0.5	0.3,0.7	0.7,0.3	0.5,0.5	0.3,0.7
頭	0.807	0.820	0.850	0.823	0.817	0.791	**0.734**	0.802	0.815
場所	0.268	0.263	0.253	0.271	0.266	0.292	**0.239**	0.287	0.243
ケース	**0.238**	0.260	0.243	0.271	0.245	0.255	0.245	0.242	0.255
核	0.122	**0.091**	0.133	0.135	0.116	0.101	0.149	0.121	0.133
記録	0.453	0.433	0.420	0.447	0.432	0.430	**0.392**	0.396	0.432
目	0.224	**0.183**	0.216	0.257	0.209	0.195	0.194	0.225	0.191
ポイント	0.400	0.414	0.403	0.399	0.416	0.390	**0.365**	0.385	0.404
線	0.799	0.821	0.820	0.803	0.776	0.797	**0.714**	0.756	0.795
自然	0.340	0.364	0.340	0.364	0.354	**0.338**	0.393	0.377	0.340
運動	**0.396**	0.403	0.407	0.405	0.420	0.401	0.433	0.405	0.408
avg.	0.405	0.405	0.408	0.418	0.405	0.399	**0.386**	0.400	0.402

太字は Entropy の最低値

4.3.7 まとめ

毎日新聞コーパスを用い, 10 単語を対象としたクラスタリングによる語義識別の実験について報告した. 特徴ベクトルを単独で利用した場合, それぞれの類似度を組み合わせた場合のクラスタリング結果を考察した. 各手法を 10 単語の平均の評価指標の値が良い順に並べた表を表 4.18 に示す. 全体的に類似度を組み合わせてクラスタリングする提案手法の性能は高かった.

4.4 Yahoo! 知恵袋を対象とした実験

本節では, Yahoo! 知恵袋コーパスから作成したテストデータを対象に, 単独の特徴ベクトルによるクラスタリング, また複数の特徴ベクトルにおける類似度を組み合わせる手法, 対象語毎に適切な特徴ベクトルを選択する手法を評価する.

4.4.1 実験方法

クラスタリング手法は k-means 法のみを用いる. クラスタ数は 10 とした. また, 毎日新聞を対象とした実験では 70 個の語義付きインスタンス (A) と語義無し 630 個のインスタンス (B) をまとめてクラスタリングし, 評価は (A) のみで行っていたが, Yahoo! 知恵袋を

表 4.18: 毎日新聞を対象とした実験のまとめ

手法	Purity	Entropy
隣接+文脈+連想	0.824	0.389
隣接+文脈	0.819	0.386
隣接+LDA 拡張文脈	0.814	0.399
隣接+PLSI 拡張文脈	0.811	0.405
隣接	0.807	0.424
連想	0.786	0.469
LDA 拡張文脈	0.784	0.460
TDL 連想	0.761	0.528
文脈	0.758	0.521
WWP 連想	0.756	-
TDP 連想	0.750	0.528
TWP 連想	0.746	-

対象とした実験ではクラスタリング・評価ともに語義つきの 100 個のインスタンスを用いる．評価尺度は Purity と Inverse Purity を用いる．Entropy による評価は省略した．これは毎日新聞を対象とした実験において Entropy と Purity の実験結果がほぼ同じ傾向を示していたからである．

4.4.2 単独の特徴ベクトルによるクラスタリングの評価

単独の特徴ベクトルを用いたクラスタリングの結果を比較する．比較したのは, 連想ベクトル (3.1.4 項), 文脈ベクトル (3.1.2 項), 隣接ベクトル (3.1.1 項), LDA 拡張文脈ベクトル (3.1.3 項), トピックベクトル (3.1.6 項) である．このうち, 連想ベクトル, 文脈ベクトル, LDA 拡張文脈ベクトル, トピックベクトルについてはウインドウ幅 w を 10 またはインスタンスが出現した文書全体に広げた場合 (ここではウインドウ幅 $w = all$ と表す) について実験した．また, 隣接ベクトルについてはウインドウ幅 w を 1 と 2 とした場合について実験した．

なお, 連想ベクトルの作成には, Yahoo! 知恵袋コーパスから作成した 10000 語-10000 語の共起行列を用いる．また, LDA 拡張文脈ベクトルやトピックベクトルの作成にあたって PLSI や LDA を適用する文書-単語の共起行列は, Yahoo! 知恵袋コーパスの質問と回答の組を 1 文書と見なして作成した共起行列を用いる．最後に, LDA 拡張文脈ベクトルのパラメータは, 毎日新聞における実験結果を踏まえて $n = 200, m = 300$ と設定する．

以上で述べた単独の特徴ベクトルのクラスタリング結果を表 4.19, 表 4.20 に示す．

Purity を基準にすると連想ベクトル ($w = all$) が平均的に良い. しかしながら, Purity が最大となった対象語の数は隣接ベクトル ($w = 1$) が 9 個と最も多く, ついで連想ベクトル ($w = all$) の 6 個, トピックベクトル ($w = 10$) の 3 個と続く. Inverse Purity でみても連想ベクトルが平均的に良く, Inverse Purity の値が最も大きい対象語も 6 個で最多となっている. 隣接ベクトル ($w = 1$), トピックベクトルがそれに次いで結果がよい. いずれの評価尺度においても, 文脈ベクトルや LDA 拡張文脈ベクトルについては平均的にも, 対象語毎にみてもクラスタリングの質は低い. しかしながら, LDA 拡張文脈ベクトルは文脈ベクトルと比較すると良い結果が得られている.

表 4.19: Yahoo! 知恵袋における単独の特徴ベクトルによるクラスタリング (Purity)

	連想		文脈		隣接		LDA 拡張文脈		トピック	
w	10	all	10	all	1	2	10	all	10	all
モデル	0.805	**0.846**	0.581	0.548	0.656	0.581	0.644	0.706	0.720	0.786
ネタ	0.615	0.623	0.584	0.590	**0.691**	0.611	0.609	0.608	0.589	0.611
カバー	0.656	0.710	0.478	0.481	0.634	0.571	0.519	0.529	0.614	**0.716**
ウイルス	0.923	**0.985**	0.850	0.850	0.867	0.850	0.931	0.873	0.877	0.931
ソース	0.889	0.891	0.739	0.733	0.745	0.734	0.759	0.777	**0.903**	0.898
肉	0.960	0.960	0.960	0.960	0.960	0.960	0.960	0.960	0.960	**0.964**
サービス	0.693	**0.694**	0.670	0.673	0.676	0.677	0.673	0.688	0.687	0.682
地方	0.743	0.710	0.662	0.646	**0.755**	0.697	0.713	0.699	0.681	0.706
アルバム	0.910	0.910	0.911	0.910	0.913	0.910	0.910	0.910	**0.916**	0.912
コード	0.710	**0.797**	0.528	0.533	0.568	0.543	0.584	0.628	0.717	0.734
自分	0.656	0.654	0.656	0.659	**0.688**	0.677	0.670	0.678	0.662	0.666
場合	0.628	0.608	0.595	0.602	**0.705**	0.692	0.612	0.615	0.642	0.657
時間	0.621	0.633	0.593	0.607	**0.805**	0.675	0.612	0.601	0.620	0.616
意味	0.689	0.705	0.697	0.696	0.706	0.703	0.696	0.690	**0.717**	0.714
電話	0.767	0.763	0.690	0.686	**0.785**	0.750	0.712	0.737	0.718	0.739
一緒	0.830	0.822	0.820	0.820	0.907	**0.911**	0.822	0.822	0.822	0.822
目	0.639	0.566	0.560	0.554	0.766	**0.798**	0.629	0.606	0.648	0.576
以前	0.867	0.870	0.870	0.870	**0.876**	0.872	0.870	0.870	0.870	0.870
代	0.840	0.788	0.600	0.618	**0.883**	0.872	0.779	0.734	0.783	0.729
顔	0.697	**0.707**	0.642	0.642	0.679	0.648	0.647	0.665	0.676	0.683
系	0.865	0.862	0.860	0.860	**0.866**	0.860	0.860	0.860	0.860	0.862
郵便	0.923	**0.928**	0.901	0.900	0.900	0.900	0.901	0.900	0.923	0.904
反応	**0.841**	0.840	0.781	0.780	0.784	0.781	0.784	0.786	0.785	0.799
avg.	0.773	**0.777**	0.706	0.705	0.775	0.751	0.735	0.737	0.756	0.764

太字は対象語毎に Purity が最大となる特徴ベクトル
w=all はウインドウ幅を文書全体とした場合.

比較対象として, 「Schütze's Context Vector」「ランダム」によるクラスタリング結果を表 4.21 に示す. 表 4.21 において「Schütze's Context Vector」は, Schütze の提案した Context Vector[10] でインスタンスを表現し, それをクラスタリングする手法を表す.

表 4.20: Yahoo! 知恵袋における単独の特徴ベクトルによるクラスタリング (Inverse Purity)

	連想		文脈		隣接		LDA 拡張文脈		トピック	
w	10	all	10	all	1	2	10	all	10	all
モデル	**0.420**	0.406	0.199	0.172	0.241	0.207	0.247	0.287	0.324	0.309
ネタ	0.321	**0.361**	0.195	0.184	0.322	0.211	0.249	0.273	0.226	0.215
カバー	0.337	0.367	0.192	0.180	0.348	0.265	0.227	0.242	0.325	**0.371**
ウイルス	0.270	0.307	0.161	0.133	0.225	0.163	0.255	0.209	**0.378**	0.338
ソース	0.391	0.415	0.185	0.158	0.274	0.201	0.278	0.226	**0.607**	0.576
肉	0.259	0.241	0.131	0.130	0.212	0.175	0.192	0.190	0.421	**0.463**
サービス	**0.284**	0.269	0.169	0.169	0.221	0.184	0.212	0.216	0.218	0.225
地方	0.245	0.228	0.147	0.142	**0.258**	0.187	0.208	0.207	0.209	0.203
アルバム	0.243	0.244	0.195	0.128	0.183	0.158	0.180	0.185	0.360	**0.384**
コード	0.346	**0.405**	0.195	0.176	0.251	0.193	0.249	0.274	0.396	0.400
自分	0.214	**0.257**	0.155	0.159	0.237	0.207	0.177	0.188	0.183	0.194
場合	0.257	0.246	0.172	0.168	**0.294**	0.258	0.195	0.202	0.208	0.214
時間	0.267	0.282	0.165	0.175	**0.324**	0.232	0.204	0.200	0.237	0.247
意味	**0.293**	0.288	0.219	0.173	0.211	0.206	0.195	0.211	0.270	0.246
電話	**0.271**	0.259	0.154	0.153	0.251	0.218	0.217	0.229	0.219	0.206
一緒	0.280	0.270	0.131	0.140	**0.365**	0.310	0.180	0.203	0.173	0.200
目	0.316	0.239	0.190	0.190	0.337	**0.341**	0.255	0.230	0.260	0.215
以前	**0.229**	0.202	0.149	0.134	0.207	0.202	0.167	0.184	0.173	0.173
代	**0.389**	0.366	0.178	0.185	0.363	0.335	0.291	0.278	0.321	0.293
顔	**0.286**	0.247	0.156	0.160	0.224	0.172	0.185	0.207	0.226	0.215
系	0.229	0.234	0.149	0.155	**0.293**	0.213	0.189	0.213	0.206	0.213
郵便	0.257	0.240	0.179	0.137	0.341	0.257	0.194	0.193	**0.429**	0.327
反応	0.327	**0.361**	0.157	0.152	0.241	0.217	0.196	0.219	0.200	0.267
avg.	**0.293**	**0.293**	0.171	0.159	0.271	0.222	0.215	0.220	0.286	0.282

太字は対象語毎に Inverse Purity が最大値となる特徴ベクトル.
w=all はウインドウ幅を文書全体とした場合.

Context Vector の作成にあたっては, まず連想ベクトル (3.1.4 項) と同じように 10,000 語 × 10,000 語の共起行列 $\mathbf{A_w}$ を作成し, その列ベクトルを Word Vector [1] とみなし, インスタンスをその周辺語 c_j の Word Vector $\vec{o}(c_j)$ の和として計算する. 和を計算する際, $\vec{o}(c_j)$ を $idf(c_j)$ により重み付けする. つまり, Context Vector は idf で重み付けした連想ベクトルと同様に式 (4.7) で計算される. ただし, 連想ベクトルの作成時に用いる共起ベクトル $\vec{o}(c_j)$(3.1.4 項) はノルムが 1 となるように正規化を行うのに対して, Word Vector は正規化を行わない点が異なる. さらに, Schütze は Context Vector のクラスタリングに Buckshot を用いているが, ここでは特徴ベクトルの違いによるクラスタリング結果をみるため, Spherical k-means を用いる. 一方, 表 4.21 の「ランダム」はベースラインであり, k 個のクラスタにランダムに要素を割り当てることによりクラスタリングを行う手法

[1] 3.1.4 項で述べた共起ベクトル $\vec{o}(c_j)$ に相当する.

を表す.

まず, Schütze's Context Vector と, それと同様に単語の 2 次共起を素性とする連想ベクトルを比較する. 表 4.21 の Schütze's Context Vector(w=10) と表 4.19, 4.20 の連想ベクトル (w=10) を比較すると, Purity は連想ベクトルの方が高く, 逆に Inverse Purity は Schütze's Context Vector の方が高い. また, Schütze's Context Vector(w=all) と連想ベクトル (w=all) を比較すると, Purity, Inverse Purity のどちらについても連想ベクトルの方が高い. また, Schütze's Context Vector を他の特徴ベクトルと比較すると, Purity は連想ベクトル, 隣接ベクトル, トピックベクトルの次に良く, Inverse Purity は Schütze's Context Vector(w=10) が最大となった.

「ランダム」と比較すると, 連想ベクトル, 隣接ベクトル, LDA 拡張文脈ベクトル, トピックベクトルの Purity および Inverse Purity は高い. しかし, 文脈ベクトルは Purity, Inverse Purity ともに「ランダム」とほぼ変わらない. しかし, 文脈ベクトルについては, 毎日新聞を対象とした 4.3.2 項の実験と同様に, クラスタリング対象とするインスタンス数を増やすことでクラスタリングの結果が改善する可能性がある.

表 4.21: Schütze's Context Vector, ランダムクラスタリングによるクラスタリング結果

	Schutze's Context Vector(w=10)		Schutze's Context Vector(w=999)		ランダム	
	Purity	Inverse Purity	Purity	Inverse Purity	Purity	Inverse Purity
モデル	0.796	0.411	0.798	0.407	0.562	0.176
ネタ	0.600	0.357	0.609	0.402	0.587	0.188
カバー	0.587	0.301	0.620	0.297	0.483	0.181
ウイルス	0.881	0.259	0.978	0.337	0.850	0.136
ソース	0.866	0.346	0.887	0.345	0.734	0.163
肉	0.960	0.247	0.960	0.239	0.960	0.131
サービス	0.682	0.263	0.696	0.301	0.675	0.169
地方	0.679	0.274	0.682	0.219	0.670	0.144
アルバム	0.911	0.311	0.910	0.275	0.910	0.122
コード	0.616	0.337	0.701	0.380	0.542	0.175
自分	0.655	0.245	0.667	0.295	0.665	0.162
場合	0.612	0.283	0.629	0.276	0.601	0.173
時間	0.627	0.274	0.635	0.269	0.596	0.172
意味	0.678	0.378	0.706	0.334	0.695	0.169
電話	0.786	0.253	0.756	0.245	0.686	0.152
一緒	0.840	0.258	0.828	0.282	0.820	0.133
目	0.700	0.374	0.589	0.240	0.544	0.187
以前	0.867	0.247	0.871	0.209	0.870	0.130
代	0.815	0.375	0.775	0.298	0.605	0.179
顔	0.656	0.327	0.660	0.265	0.636	0.148
系	0.862	0.280	0.867	0.264	0.860	0.152
郵便	0.902	0.250	0.916	0.238	0.900	0.139
反応	0.799	0.306	0.813	0.294	0.780	0.153
avg.	0.756	0.302	0.763	0.292	0.706	0.158

4.4.3 類似度の組み合わせによるクラスタリングの評価

本項では異なる特徴ベクトルにおける類似度を組み合わせる手法 (3.3.1 項) によるクラスタリングを評価する. 今回の実験では単独で性能の高い隣接ベクトル ($w = 1$)・トピックベクトル ($w = all$), そしてこれとは大きく異なる素性を用いた LDA 拡張文脈ベクトル ($w = all$) を組み合わせる. 実験結果を表 4.22 に示す. なお, ここでは (0.6, 0.3, 0.1) は左からそれぞれ隣接・トピック・LDA 拡張文脈における類似度に対する重みを表す.

単独で性能の高い隣接ベクトル ($w = 1$) を基準とすると平均的には Purity が向上し, Inverse Purity は低下している. 重みを (0.5, 0.3, 0.2) に設定した場合に Purity が隣接ベクトル ($w = 1$) より改善している対象語が 8 個, 重みを (0.5, 0.4, 0.1) に設定した場合に Inverse Purity が隣接ベクトル ($w = 1$) を上回っている対象語が 6 個あり, 対象語によっては類似度の組み合わせが有効である. しかしながら, 「自分」や「地方」などといった隣接ベクトル ($w = 1$) で良い結果が得られる対象語で Purity, Inverse Purity が低下している. 類似度の組み合わせが有効な対象語であるかどうかをあらかじめ判別できれば, 全体的なクラスタリングの質をさらに向上させることができると考えられる.

4.4.4 特徴ベクトルを対象語毎に選択する手法の評価

本項では対象語毎に適切な特徴ベクトルを選択してクラスタリングする手法 (3.3.2 項) を評価する. 組み合わせる特徴ベクトルは, Purity や Inverse Purity の値からみて相補的であり, 互いに異なる素性を用いた隣接ベクトル (3.1.1 項, $w = 1$), 連想ベクトル (3.1.4 項, $w = all$), LDA 拡張文脈ベクトル (3.1.3 項, $w = all$), トピックベクトル (3.1.6 項, $w = all$) とする. 実験結果を表 4.23 に示す.

クラスタ内類似度 intra(式 (3.24)), クラスタ凝集度 coh(式 (3.25)) で特徴ベクトルを選択した場合は, クラスタリングの評価値は隣接ベクトル ($w = 1$) や連想ベクトル ($w = all$) 単独のときと比べて低く, トピックベクトル ($w = all$) 単独のときとほぼ同等であった. 一方, 相対的クラスタ内類似度 rel_intra(式 (3.27)), 相対的クラスタ凝集度 rel_coh(式 (3.28)) で特徴ベクトルを選択した場合は, 平均的にみて全ての特徴ベクトル単独の結果よりも良いクラスタリング結果となっており, ある程度適切な特徴ベクトルを選択できている. しかしながら, 連想ベクトル ($w = all$) 単独の結果と比較すると「モデル」「カバー」「サービス」「コード」「顔」「反応」において Purity と Inverse Purity が共に低く, 対象語によっては特徴ベクトルの選択が必ずしもうまくいかないことを示している.

表 4.22: 隣接 ($w = 1$), トピック ($w = all$), LDA 拡張文脈 ($w = all$) の類似度組み合わせ

重み	0.6, 0.3, 0.1		0.5,0.4,0.1		0.5,0.3,0.2	
	purity	i-purity	purity	i-purity	purity	i-purity
モデル	**0.762**	**0.281**	**0.791**	**0.304**	**0.770**	**0.293**
ネタ	0.643	0.256	0.623	0.233	0.619	0.219
カバー	**0.712**	0.346	**0.748**	**0.354**	**0.723**	0.337
ウイルス	**0.904**	**0.248**	**0.926**	**0.266**	**0.900**	**0.261**
ソース	**0.874**	**0.319**	**0.888**	**0.358**	**0.884**	**0.310**
肉	0.960	0.205	0.960	0.199	0.960	0.191
サービス	**0.678**	0.201	**0.686**	0.215	**0.686**	0.205
地方	0.709	0.190	0.741	0.214	0.727	0.189
アルバム	0.912	0.179	0.910	0.184	0.910	0.208
コード	**0.693**	**0.318**	**0.741**	**0.361**	**0.740**	**0.360**
自分	0.672	0.199	0.677	0.188	0.680	0.183
場合	0.678	0.220	0.668	0.216	0.652	0.220
時間	0.686	0.254	0.671	0.242	0.703	0.240
意味	0.706	0.195	**0.722**	**0.221**	**0.717**	**0.212**
電話	0.755	0.220	0.743	0.210	0.766	0.214
一緒	0.893	0.248	0.869	0.230	0.884	0.238
目	0.707	0.278	0.652	0.255	0.685	0.273
以前	0.871	0.181	0.871	0.171	0.870	0.179
代	0.845	0.333	0.834	0.352	0.847	0.340
顔	0.676	0.206	0.673	0.207	0.663	0.201
系	0.861	0.200	0.860	0.198	0.860	0.193
郵便	**0.921**	0.304	**0.923**	0.287	**0.927**	0.275
反応	0.784	0.230	0.784	0.238	**0.790**	0.237
avg.	**0.778**	0.244	**0.781**	0.248	**0.781**	0.243

太字は隣接ベクトル ($w = 1$) 単体より改善した場合

以上で述べたように, 対象語によってはクラスタリングに有効な特徴ベクトルが選択されていない可能性がある. そこで, intra, coh, rel_intra, rel_coh をそれぞれ評価関数として利用した場合について, 対象語毎に各特徴ベクトルが選択された回数を調査した. ここでは, 隣接ベクトル ($w = 1$), 連想ベクトル ($w = all$), LDA 拡張文脈ベクトル ($w = all$), トピックベクトル ($w = all$) を選択対象とした. そして, それぞれの特徴ベクトルでクラスタリングを行い評価値の高い特徴ベクトルを選ぶことを各対象語について 10 回ずつ行った.

まず, intra により特徴ベクトルを選択した. その結果, 23 単語全てにおいてトピックベクトルが選択され, 単語毎に異なる特徴ベクトルが選ばれることはなかった.

coh を特徴ベクトルの選択に用いた場合を表 4.24 に示す. ほとんどの対象語でトピック

表 4.23: 特徴ベクトル (隣接・連想・LDA 拡張文脈・トピック) を対象語毎に選択した結果

	intra		coh		rel_intra		rel_coh	
	purity	i-purity	purity	i-purity	purity	i-purity	purity	i-purity
モデル	**0.803**	**0.304**	**0.772**	**0.308**	**0.801**	**0.316**	**0.669**	**0.257**
ネタ	0.615	0.233	0.656	0.288	0.609	0.223	0.675	0.296
カバー	**0.714**	**0.371**	**0.700**	**0.360**	**0.693**	**0.377**	**0.639**	**0.351**
ウイルス	**0.935**	**0.321**	**0.916**	**0.305**	**0.927**	**0.332**	**0.935**	**0.323**
ソース	**0.897**	**0.544**	**0.893**	**0.557**	**0.897**	**0.559**	**0.892**	**0.561**
肉	0.960	**0.435**	0.960	**0.453**	**0.961**	**0.453**	**0.962**	**0.478**
サービス	**0.684**	0.219	**0.687**	**0.229**	**0.688**	0.218	**0.679**	**0.222**
地方	0.708	0.203	0.710	0.210	0.693	0.201	0.721	0.215
アルバム	**0.914**	**0.391**	**0.914**	**0.279**	**0.920**	**0.372**	**0.920**	**0.377**
コード	**0.732**	**0.411**	**0.746**	**0.404**	**0.747**	**0.412**	**0.713**	**0.366**
自分	0.663	0.187	0.676	0.199	0.662	0.201	0.685	**0.250**
場合	0.674	0.209	0.655	0.210	**0.721**	0.278	0.699	0.268
時間	0.619	0.244	0.610	0.250	0.644	0.235	0.786	0.309
意味	**0.719**	**0.250**	**0.709**	**0.242**	0.703	**0.232**	**0.717**	**0.234**
電話	0.731	0.213	0.749	0.212	0.741	0.220	**0.790**	0.251
一緒	0.825	0.199	0.820	0.186	**0.922**	0.359	0.906	0.336
目	0.577	0.216	0.583	0.222	0.596	0.231	0.710	0.288
以前	0.870	0.181	0.872	0.186	0.870	0.182	0.873	**0.210**
代	0.739	0.308	0.733	0.291	**0.891**	0.353	**0.888**	**0.368**
顔	0.671	0.211	**0.685**	0.216	0.662	0.201	**0.680**	**0.242**
系	0.861	0.209	0.860	0.200	0.863	0.289	0.860	0.284
郵便	**0.908**	**0.355**	**0.914**	**0.360**	**0.910**	0.333	**0.902**	0.329
反応	**0.803**	**0.277**	**0.796**	**0.256**	**0.792**	**0.249**	**0.790**	0.216
avg.	0.766	**0.282**	0.766	**0.279**	**0.779**	**0.297**	**0.787**	**0.306**

太字は隣接ベクトル ($w = 1$) 単独より改善した場合

ベクトルが選択され, その他に選択されたのは隣接ベクトルのみであった. 両方の特徴ベクトルが選択された対象語のうち, Purity を基準とすると, 隣接ベクトルが 1 回以上選ばれた単語のうち, 「モデル」「ウイルス」はトピックベクトルの方が良い. Inverse Purity を基準とすると, 「モデル」「ネタ」「ウイルス」「アルバム」はトピックベクトルの方が良い. したがって, coh により選択された特徴ベクトルは必ずしも対象語のクラスタリングに最適であるとはいえない.

rel_intra や rel_coh を特徴ベクトルの選択に用いた場合を表 4.25 に示す. なお, 表中の数値の横のアルファベット (p) は, 選ばれた回数が最も多い特徴ベクトルで, かつそのベクトルが他のベクトルと比べて最も Purity の値が高かった場合を, (i) は同様に Inverse Purity

の値が高かった場合を表す．つまり，(p) または (i) のついている単語は，提案手法によって最適な特徴ベクトルを選択できる可能性が高いことを示している．

まず，rel_intra を特徴ベクトルの選択に用いた結果に注目する．rel_intra により選択された特徴ベクトルは隣接・LDA 拡張文脈・トピックベクトルの 3 つであった．表 4.25 によると，23 単語のうち 8 単語のついては Purity か Inverse Purity が高い特徴ベクトルを選択できることがわかる．しかし，Purity や Inverse Purity が低い LDA 拡張文脈ベクトルが選択される回数が多い対象語もある．したがって，rel_intra による評価値が高いからといって，実際に Purity や Inverse Purity も必ず高いわけではない．

次に，rel_coh を特徴ベクトルの選択に用いた結果に注目する．rel_coh により選択された特徴ベクトルは隣接ベクトルとトピックベクトルのいずれかである．23 単語のうち 10 単語については Purity か Inverse Purity が高い特徴ベクトルを選択できることがわかる．

4 つの評価値を全体的にみれば，トピックベクトル・隣接ベクトル・LDA 拡張文脈ベクトルの順に選択された回数が多い．また，rel_intra, rel_coh は intra や coh より多様な特徴ベクトルを選択できた．Purity かまたは Inverse Purity が最大となるような特徴ベクトルが正しく選択された対象語の数は rel_coh を用いた場合が最多となり，10 であった．ただし，表 4.19, 表 4.20 によれば連想ベクトルの Purity や Inverse Purity が高い対象語もあるが，これが選ばれることはなかった．したがって，クラスタリングの評価関数には改善の余地がある．

表 4.24: coh によって特徴ベクトルが選択された回数

	coh	
	隣接	トピック
モデル	2	8
ネタ	3	7
ウイルス	4	6
アルバム	8	2
以前	6	4

その他の対象語では全てトピックが 10 回選択された

表 4.25: rel_intra, rel_coh によって特徴ベクトルが選択された回数

	rel_intra			rel_coh	
	隣接	LDA 拡張文脈	トピック	隣接	トピック
モデル	0	0	10	5	5
ネタ	0	0	10	$7(p)$	3
カバー	1	0	$9(p,i)$	8	2
ウイルス	0	0	10	0	10
ソース	0	0	10	0	10
肉	0	0	$10(p,i)$	1	$9(p,i)$
サービス	0	1	9	9	1
地方	0	6	4	5	5
アルバム	0	0	$10(i)$	1	$9(i)$
コード	0	0	10	3	7
自分	$4(p)$	3	3	$7(p)$	3
場合	$10(p,i)$	0	0	$10(p,i)$	0
時間	1	6	3	$8(p,i)$	2
意味	8	0	2	10	0
電話	2	0	8	$9(p)$	1
一緒	$10(i)$	0	0	$10(i)$	0
目	0	9	1	7	3
以前	0	0	10	$6(p)$	4
代	$9(p)$	0	1	$10(p)$	0
顔	0	0	10	10	0
系	$10(p,i)$	0	0	$10(p,i)$	0
郵便	0	0	10	3	7
反応	3	2	5	8	2

表に記載されていない特徴ベクトルは一度も選択されなかった
数値の横のアルファベットについては, 選択された回数が多い特徴ベクトルであり,
かつ Purity 最大となるベクトルには p, または Inverse Purity が最大となるベクトルには i を付与した.

4.4.5　クラスタリング結果と評価関数について

前項では intra, coh による特徴ベクトルの選択はうまくいかず, rel_intra, rel_coh による選択は, 連想ベクトル単独に及ばない場合はあるものの, 全体的にはクラスタの質を向上させることができた. 本項では, 評価関数 intra, coh, rel_intra, rel_coh の実際の値を調べ, それらがどの程度クラスタリングの良さを反映しているか調べる.

クラスタ内類似度, クラスタ凝集度による評価

特徴ベクトルを単独で利用してクラスタリングした場合に Purity や Inverse Purity の高かった連想・隣接・トピックベクトルについて, 3.3.2 項で述べた intra, coh によるクラスタリングの評価値が Purity や Inverse Purity と相関しているかを調査した. 結果を表 4.26 に示す.

全ての対象語においてトピックベクトルの評価値が最大となった. 対象語によっては隣接ベクトルや連想ベクトルの方が有効であるため, intra, coh の 2 つの評価関数はクラスタの質を正しく反映しているとはいえない. この原因としては, 各特徴ベクトルにおける類似度の絶対値の差が大きいことが考えられる. 例えば, 今回のテストデータにおいてトピックベクトル間の類似度は 0.1 ± 0.2 程度であったが, LDA 拡張文脈ベクトル間の類似度は 0.0003 ± 0.005 程度であった. その結果, intra や coh の値もトピックベクトルが他のベクトルと比べて大きくなり, トピックベクトルのみが選ばれる結果となった.

相対的クラスタ内類似度・クラスタ凝集度による評価

連想・隣接・トピックベクトルについて, rel_intra, rel_coh によるクラスタリングの評価値と Purity や Inverse Purity との相関を調査した. 結果を表 4.27 に示す.

rel_intra を利用すると, ほとんどの対象語でトピックベクトル ($w = all$) の評価値が最大となる. しかし, 「場合」「意味」「一緒」「代」「系」では隣接ベクトル ($w = 1$) の方が評価値が高い. 表 4.19 によれば, これらのうち「場合」「一緒」「代」「系」の場合は隣接ベクトル ($w = 1$) はトピックベクトル ($w = all$) を Purity で上回っている. また表 4.20 によれば, やはり「場合」「一緒」「代」「系」において隣接ベクトル ($w = 1$) の方が Inverse Purity が高い. したがって, rel_intra によるクラスタリングの評価値はある程度信頼できる.

一方 rel_coh による評価では, 隣接ベクトル ($w = 1$) の評価値が最大となる場合が多い. しかし, 「ウイルス」「肉」「アルバム」「郵便」ではトピックベクトル ($w = 10$) の評価値がそれを上回る. 表 4.19 によればこのうち「ウイルス」「アルバム」「郵便」については

表 4.26: intra, coh によるクラスタリングの評価 (連想, 隣接, トピック)

評価関数	intra						coh					
特徴ベクトル	連想		隣接		トピック		連想		隣接		トピック	
w	10	all	1	2	10	all	10	all	1	2	10	all
モデル	2.68	1.16	2.62	1.23	6.44	**6.97**	2.98	1.26	26.30	5.33	**42.87**	34.76
ネタ	3.79	2.28	2.54	1.19	6.04	**6.39**	4.62	2.63	30.49	5.72	**43.35**	32.17
カバー	2.82	1.04	2.77	1.45	6.37	**7.06**	3.12	1.12	20.81	7.24	**55.89**	39.11
ウイルス	3.07	1.20	2.96	1.48	7.99	**8.20**	3.56	1.33	30.05	7.81	**51.78**	30.12
ソース	2.67	1.23	2.59	1.33	8.13	**8.42**	3.11	1.38	18.57	5.09	**50.37**	35.91
肉	2.69	1.33	2.68	1.37	7.47	**7.87**	3.09	1.45	20.19	6.04	**31.06**	20.67
サービス	2.35	0.72	2.48	1.14	6.63	**6.83**	2.66	0.77	22.03	4.23	**53.95**	35.11
地方	2.86	0.83	2.73	1.37	6.41	**6.92**	3.41	0.89	23.03	7.34	**50.94**	34.07
アルバム	3.76	1.73	2.64	1.32	6.83	**7.58**	4.25	1.92	20.96	6.72	**29.54**	19.01
コード	3.13	1.40	2.57	1.20	7.26	**7.61**	3.80	1.63	17.31	4.30	**52.13**	34.77
自分	3.27	0.81	3.52	1.72	6.18	**6.72**	3.86	0.86	16.85	6.26	**58.03**	39.25
場合	2.55	0.75	3.10	1.69	6.23	**6.70**	2.93	0.84	19.10	8.93	**62.90**	45.11
時間	2.61	0.85	2.73	1.24	**6.54**	6.52	2.85	0.93	19.26	4.81	**54.47**	32.29
意味	4.14	2.68	2.96	1.59	6.05	**6.23**	4.92	3.15	16.77	7.75	**58.35**	38.43
電話	2.56	0.85	2.76	1.46	6.47	**7.04**	2.88	0.93	18.28	7.83	**39.27**	32.51
一緒	2.81	0.65	3.53	1.78	6.65	**7.09**	3.33	0.71	9.81	4.53	**64.88**	35.46
目	2.67	0.69	2.97	1.59	6.01	**6.56**	2.92	0.73	13.61	6.31	**37.76**	31.74
以前	3.00	0.83	3.44	1.77	5.88	**6.50**	3.64	0.91	42.40	14.01	**53.64**	42.31
代	3.30	0.77	2.88	1.62	6.48	**6.68**	4.12	0.83	13.66	7.30	**62.36**	34.69
顔	2.86	1.29	2.85	1.31	6.57	**6.72**	3.18	1.38	19.25	4.66	**44.56**	30.38
系	2.53	0.93	2.81	1.47	6.17	**6.44**	2.87	1.00	14.64	4.28	**52.04**	35.46
郵便	2.73	1.03	3.27	1.85	8.13	**8.21**	3.00	1.11	17.87	8.79	**23.90**	22.49
反応	2.80	0.97	2.79	1.47	6.42	**7.08**	3.24	1.10	20.46	7.92	**55.62**	50.07
avg.	2.94	1.13	2.88	1.46	6.67	**7.06**	3.41	1.25	20.51	6.66	**49.12**	34.17

太字はそれぞれの評価関数が最大値を示す特徴ベクトルと w の組み合わせ

トピックベクトル ($w = 10$) の Purity は高い．また表 4.20 によれば 4 単語全てで Inverse Purity が高くなっている．そのほかの対象語では，「ソース」「地方」「コード」においてトピックベクトル ($w = all$) の評価値が隣接ベクトル ($w = 1$) を上回っているが，そのうち「ソース」「コード」については Purity, Inverse Purity ともにトピックベクトルの方が高い．したがって，rel_coh によるクラスタリングの評価値もある程度信頼できる．

表 4.27: rel_intra, rel_coh によるクラスタリングの評価 (連想, 隣接, トピック)

評価関数	rel_intra						rel_coh					
特徴ベクトル	連想		隣接		トピック		連想		隣接		トピック	
w	10	all	1	2	10	all	10	all	1	2	10	all
モデル	6.77	5.00	7.34	7.15	7.71	**8.12**	7.00	5.19	**10.60**	8.63	9.59	9.62
ネタ	5.60	4.82	7.24	7.31	7.57	**7.77**	5.98	5.09	**9.31**	8.39	8.69	8.85
カバー	5.65	4.02	7.87	7.58	7.61	**8.15**	5.87	4.11	**10.27**	8.73	8.95	10.14
ウイルス	5.75	4.65	6.71	6.78	8.64	**8.81**	6.09	4.84	8.64	7.79	**13.46**	12.65
ソース	6.28	5.55	7.26	7.25	8.75	**9.15**	6.64	5.78	10.34	8.35	13.26	**13.85**
肉	6.29	4.46	7.44	7.37	8.34	**8.79**	6.56	4.59	10.53	8.60	**12.19**	12.03
サービス	6.53	4.76	7.36	7.91	7.70	**8.02**	6.82	4.87	**10.09**	9.03	8.96	9.48
地方	6.33	4.75	6.72	7.07	7.54	**8.05**	6.64	4.83	9.08	8.52	9.18	**9.66**
アルバム	5.58	4.65	7.34	7.20	7.92	**8.47**	5.80	4.78	10.48	8.55	**11.77**	10.95
コード	6.06	4.59	7.63	7.64	8.13	**8.38**	6.41	4.80	10.51	8.79	10.79	**11.45**
自分	5.67	4.27	7.74	7.56	7.47	**7.95**	5.96	4.37	**9.60**	9.04	8.72	9.29
場合	6.06	5.04	**8.37**	8.00	7.49	7.96	6.32	5.23	**10.95**	9.59	9.03	9.80
時間	5.71	4.93	7.53	7.65	7.63	**7.66**	5.89	5.07	**9.84**	8.74	9.41	9.13
意味	6.04	4.59	**7.81**	7.62	7.53	7.74	6.48	4.88	**10.11**	9.00	9.68	9.32
電話	6.08	4.53	7.89	7.45	7.66	**8.11**	6.28	4.67	**10.30**	9.11	10.11	10.01
一緒	6.58	5.39	**9.04**	8.50	7.67	8.12	6.96	5.56	**11.49**	10.22	8.80	10.23
目	5.90	4.51	7.76	7.58	7.55	**8.02**	6.18	4.60	**9.54**	9.33	8.56	9.28
以前	6.20	5.18	7.24	7.07	7.38	**7.86**	6.56	5.35	**10.01**	8.81	9.31	9.23
代	5.68	4.79	**8.37**	8.23	7.68	8.02	6.12	4.93	**11.66**	10.29	8.87	9.28
顔	6.35	4.02	7.21	7.40	**7.74**	8.04	6.60	4.10	**10.22**	8.55	9.31	9.59
系	5.85	4.50	**8.58**	7.58	7.56	7.90	6.06	4.61	**12.54**	9.46	8.56	9.21
郵便	6.40	4.38	7.79	7.07	8.77	**8.89**	6.57	4.50	11.36	9.10	**11.76**	11.59
反応	5.73	4.50	7.95	7.65	7.65	**8.05**	6.04	4.66	**10.13**	9.15	9.13	9.76
avg.	6.05	4.69	7.66	7.51	7.81	**8.18**	6.34	4.84	**10.33**	8.95	9.92	10.19

太字はそれぞれの評価関数が最大値を示す素性と w の組み合わせ

4.4.6 まとめ

本節では Yahoo! 知恵袋コーパスから抽出した 23 単語を対象として, 以下の手法を評価した.

1. 特徴ベクトルを単独で使用する手法
2. 類似度を組み合わせる手法
3. 特徴ベクトルの選択によるクラスタリング手法

各手法によるクラスタリングの結果を Purity の降順で並べた表を表 4.28 に示す.

まず, Purity の値により手法を比較する. 特徴ベクトルをただ一つ利用してクラスタリングした場合は, 連想ベクトル, 隣接ベクトル, トピックベクトル, Schütze's Context Vector, LDA 拡張文脈ベクトルの順に Purity が高かった. 隣接ベクトル, トピックベクトル, LDA 拡張文脈ベクトルの類似度を組み合わせる手法 (3.3.1 項) によりクラスタリングを行ったところ, 単独で最も良かった連想ベクトルと比較して Purity が向上した. 特徴ベクトルを選択する手法 (3.3.2 項) により, 類似度を組み合わせる手法よりさらに Purity が向上した.

次に, Inverse Purity の値により手法を比較する. 特徴ベクトルをただ一つ利用してクラスタリングした場合は, Schütze's Context Vector, 連想ベクトル, トピックベクトル, 隣接ベクトル, LDA 拡張文脈ベクトルの順に Inverse Purity が高かった. 隣接ベクトル, トピックベクトル, LDA 拡張文脈ベクトルの類似度を組み合わせる手法によるクラスタリング結果の Inverse Purity は隣接ベクトルの次に大きい値となり, 単独で最も高かった Schütze's Context Vector には及ばなかった. しかし, 特徴ベクトルを選択する手法によるクラスタリング結果の Inverse Purity は Schütze's Context Vector を上回った. 全体として, 特徴ベクトルを単語毎に選択する手法が, Purity, Inverse Purity 共に最大となることがわかった.

特徴ベクトルの選択に利用するクラスタリングの評価関数についてまとめる. intra, coh で特徴ベクトルを選択する手法によりクラスタリングを行ったところ, 特徴ベクトルを単独で用いてクラスタリングする手法と比較して Purity や Inverse Purity の向上はみられなかった. また, intra や coh はクラスタリングの結果に関係なくトピックベクトルの評価値が高く, 特徴ベクトルの選択には適切ではないことがわかった. 一方, rel_intra や rel_coh で特徴ベクトルを選択する手法によりクラスタリングを行ったところ, 特徴ベクトルを単独で用いた場合と比較して Purity や Inverse Purity が向上した. rel_intra を用いた場合は隣接ベクトル, LDA 拡張文脈ベクトル, トピックベクトルが選択されていた. その中でも

LDA 拡張文脈ベクトル以外は Purity や Inverse Purity が高くなるような特徴ベクトルが対象語毎に選択された. rel_intra を用いた場合は, 隣接ベクトル, トピックベクトルのうち Purity や Inverse Purity の高くなる方が対象語毎に選択される傾向が見られた. 4 つの評価関数の中では, rel_coh による特徴ベクトルの選択が最も良いクラスタリング結果となることがわかった. ただし, 単独でクラスタリング結果のよかった連想ベクトルはこれらの評価関数では選択されなかった.

表 4.28: Yahoo! 知恵袋を対象とした実験のまとめ

手法	Purity	Inverse Purity
特徴ベクトル選択: rel_coh	0.787	0.306
類似度組み合わせ: 0.5, 0.4, 0.1	0.781	0.248
特徴ベクトル選択: rel_intra	0.779	0.297
類似度組み合わせ: 0.6, 0.3, 0.1	0.778	0.244
連想 (w=all)	0.777	0.293
隣接 (w=1)	0.775	0.271
特徴ベクトル選択: intra	0.766	0.282
特徴ベクトル選択: coh	0.766	0.279
トピック (w=all)	0.764	0.282
Schütze's Context Vector(w=all)	0.763	0.292
Schütze's Context Vector(w=10)	0.756	0.302
LDA 拡張文脈 (w=all)	0.737	0.220
文脈 (w=10)	0.706	0.171
ランダム	0.706	0.158

表中において各手法は Purity の降順で並んでいる. また, 「類似度組み合わせ」は, 手法の右に記載されている数値を左から順に隣接ベクトル (w=1), トピックベクトル (w=all), LDA 拡張文脈ベクトル (w=all) の類似度に対する重みとした場合の結果を表す. 同様に, 「特徴ベクトル選択」は, 手法の右に記載されている評価関数の値の高い特徴ベクトルを, 隣接ベクトル (w=1), 連想ベクトル (w=all), LDA 拡張文脈ベクトル (w=all) の中から選択した場合の結果を表す.

第5章　おわりに

WSD の手法は辞書によってあらかじめ語義を定義し, 辞書の定義文やシソーラスを基に語義の判別規則を教師あり学習するアプローチが主流である. しかしながら, 辞書は語義の移り変わりにより再編しなければならない. 本研究では, 語義を自動的に発見する手法を提案した. 提案手法中で, Word Sense Discrimination や語義別用例分類といわれる教師なし学習の手法を応用した. しかしながら, 従来の手法が対象語毎に適切な素性が異なることを考慮していなかったのに対して, 本研究では特徴ベクトルを対象語毎に選択したり, 異なる特徴ベクトルにおける類似度を組み合わせることで語義判別の正解率を向上させることを試みた.

評価実験の結果, 毎日新聞のテストデータに対しては類似度の組み合わせが有効であることを示した. また Yahoo! 知恵袋のテストデータに対しては特徴ベクトルを選択する手法が有効であることを示した.

最後に本研究に関する課題を述べる.

- より多くの対象語についての実験
  本研究では新聞記事より 10 単語, Yahoo!知恵袋より 23 単語を選び実験した. しかし, 辞書作成の半自動化を想定して, より多くの単語について実験を行わなければならない.

- ウインドウ幅の自動判別
  知恵袋を対象とした実験ではいくつかの特徴ベクトルでウインドウ幅を広げることでクラスタの質が向上した. 特徴ベクトルの選択だけでなく, 最適なウインドウ幅の選択についても何らかの手法を考案しなければならない.

- 隣接, トピックベクトル以外の選択方法
  本研究では相対的な類似度によりクラスタ内類似度やクラスタ凝集度を算出したが, 単語によってはクラスタリング結果が最も良い連想ベクトルについては評価値が低く, 選択できなかった. したがって, クラスタリングの評価関数を再考する必要がある.

- 語義数の自動発見

  本研究ではクラスタ数を 10 と固定して考えたが, 本来は語義の数だけクラスタが出力されることが望ましい. このようなクラスタ数の自動判定に関してはいくつか先行研究が存在し, 本研究に応用することを考えている. 例えば, Tibshirani らは Gap statistic と呼ばれる手法を提案している [11]. これはクラスタ数 k を変化させながら

  (a) ランダムに生成したデータのクラスタ

  (b) 実データのクラスタ

  を生成し, それぞれの評価関数を計算し, そして (a) と (b) の評価関数の差が最大となる k を選択するというものである. これは直感的には最も意味のありそうなクラスタが生成される k を選択している.

謝辞

本研究を進めるにあたり, 白井清昭准教授, 島津明教授, 東条敏教授, 中村誠助教には, 数多くのご教示をいただきました. また, 白井研究室・島津研究室の皆様方には, 本研究に関する貴重なご支援をいただきました. この場を借りて感謝申し上げます.

参考文献

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, Vol. 3, pp. 993–1022, January 2003.

[2] Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 318–329, 1992.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 39, No. 1, pp. 1–38, 1977.

[4] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, Vol. V42, No. 1, pp. 143–175, January 2001.

[5] Brian S. Everitt. *Cluster Analysis*. A Hodder Arnold Publication, 3rd edition, March 1993.

[6] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57. ACM Press, 1999.

[7] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics*, Vol. 24, No. 1, pp. 2–40, March 1998.

[8] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 613–619, New York, NY, USA, 2002. ACM Press.

[9] Amruta Purandare and Ted Pedersen. Word sense discrimination by clustering contexts in vector and similarity spaces, 2004.

[10] Hinrich Schutze. Automatic word sense discrimination. *Computational Linguistics*, Vol. 24, No. 1, pp. 97–123, 1998.

[11] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic, 2000.

[12] David Yarowsky. One sense per collocation. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pp. 266–271, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

[13] Ying Zhao and George Karypis. Comparison of agglomerative and partitional document clustering algorithms. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, 2002.

[14] 菊田篤史, 白井清昭. 未定義語の判別を含む語義曖昧性解消. Master's thesis, 北陸先端科学技術大学院大学　情報科学研究科, 2006.

[15] 玉垣隆幸, 白井清昭. 読解支援システムのための語義曖昧性解消に関する研究. Master's thesis, 北陸先端科学技術大学院大学　情報科学研究科, March 2003.

付 録A　毎日新聞における対象語の語義

毎日新聞における対象語の語義を, 本研究で語義の基準とした岩波国語辞典より抜粋する. 左側の数値はその語義で使われていたインスタンスの数を表す.

頭
35 動物の、脳や目・口・耳・鼻などがある部分。
22 髪や頭の働き。
12 物の上部。てっぺん。上に立つ人。うわまえ。最初。
 1 あたまかず。人数。「一人頭千円」

場所
57 ところ。場（ア)。「一をふさぐ」
13 相撲を興行する所。その一定の期間。

ケース
61 場合。事例。
 9 箱。入れ物。

核
58 物の中心の部分。
12 物事の中心。かなめ。

記録
52 競技などの成績・結果。特にその最高のもの。「世界一」「一破りの暑さ」
18 後後まで伝える必要のある事柄を書きしるすこと。その書きしるしたもの。「一に残す」

目

40 生物の、物を見る働きをする器官。また、その様子・働き。
26 順序を表す時に添える語。「三番―の問題」「二つ―の角を曲がる」「十軒―」
2 ある物事に出会うこと。経験。体験。また、局面。「ひどい―を見る」「落第の―」
1 形が(1)に似ているもの。「台風の―」「碁盤の―」「さいころの―」
1 (読み誤り)原文は「が進んでいる。各界各層のリーダーと目された人たちの責任感、倫理」

ポイント
30 「パーセント　ポイント」の略。二つの百分率の値を比べた時の差。「内閣支持率が前回調査より三―下がった」▽ｐｏｉｎｔ（＝点）
25 要点。「―を押さえる」
11 点数。得点。「―をかせぐ」
4 地点。位置。「チェック―」「釣りの―」

線
28 面の上に描かれるすじ。「点―」「水平―」
22 まっすぐに進み、または一つらなりになっていると、考えたもの。道すじ。「紫外―」「視―」「新幹―」
　決められた筋道。方針。「この線で行こう」「国策の線に沿う」。水準。「いい線に達した
20 糸のように細長く作られたもの。「線を引く」「線香・線路・鉄線・電線・ピアノ線・導火線・断線・脱線・混線」

自然
42 この世のあらゆる物の総称。
28 人手を加えない、物のありのままの状態・成行き。

運動
54 目的達成のために、いろいろな方面に働きかけて努力すること。
13 健康や楽しみのために、体を動かすわざ、特にスポーツ。
3 物体（質点）が時の経過と共に位置を移す物理現象。

付 録B　Yahoo! 知恵袋における対象語の語義

Yahoo! 知恵袋における対象語の語義を, 本研究で語義の基準とした岩波国語辞典より抜粋する. 左側の数値はその語義で使われていたインスタンスの数を表す. 新語義については「(新語義)」というマークをつけて説明する.

モデル
45（新語義）機械・自動車などの型式。
44 ファッションモデルの略。
 7 美術製作の対象となるもの・人。文学作品の素材となる人。
 2（新語義）事象を定式化したもの。
 2 手本。模範。

ネタ
58 新聞記事などの材料。
21（新語義）うそ。
14（新語義）物語などの核心。
 6 （料理の材料としての）食物。「すしの―」
 1 手品の仕掛け。

カバー
44 他のものを覆うのにつかうもの。
22（新語義）カバーバージョン。楽曲を別の人が演奏・または歌ったりすること。
22 足りないところを補うこと。
12（新語義）覆う行為。

ウイルス

85 電子計算機で、ソフトウェアにひそかに仕掛けて、正常な働きをさせなくしたり格納データを消し去ったりする、悪性のソフトウェア。
15 最近より小さく、光学顕微鏡では見えない、一群の微生物。

ソース
74 西洋料理に調味料として使う汁。種類が多い。
18 (新語義) ソースコード。
 8 出どころ。源泉。
 1 不明。原文は「*ソース・プルゥング・ロッとタオ・ヂアオの２つ」

肉
96 動物のかだらの、皮膚におおわれ、骨格を包む、やわらかな物質。食用にする鳥獣のにく。
 2 ふとりかげん。あつみ。ふとさ。
 1 「印肉」の略。
 1 精神に対する、人間の物質的要素。からだ。

サービス
67 商売で、値引きしたり客の便宜を図ったりすること。「百円―しておきます」「アフター―」
18 客に対するもてなし。接待。優遇。「―のよい旅館」
 9 (新語義) (コンピュータ、情報) サーバが提供するもの。
 6 奉仕。「家庭―」
 1 不明。原文は「スタッフ*サービスで稼動中の方、教えてください。」

地方
63 首府以外の地域。⇔中央。「―機関」
37 (国内の) ある一定の地域。

アルバム
91 (特定のテーマにより) いくつかの曲を収めたＬＰレコードやＣＤなど。▽ａｌｂｕｍ

9 写真帳。記念帳。「卒業―」

コード
46 符号。「―ブック」
　こういうことをしないようにと定めた準則。倫理規定。「プレス―」▽ｃｏｄｅ（＝法典。おきて）
34 ゴムなどで絶縁した電線。主に室内用。▽ｃｏｒｄ
19 楽器の弦。
　和音。▽ｃｈｏｒｄ
1 不明。原文は「はありませんが、まだまだ未使用の*コードはたくさんあるので、再利用は当分」

自分
64 その人自身。「―のことは―でせよ」「―で言ったくせに」「太郎は弟に―の（＝太郎の）荷物を持たせた」
35 《代名詞的に》わたくし。「―は二等兵であります」＜関連＞己・みずから・私・我・余・自家・自我・自己・自身・当人・本人
1 (新語義) 自分の体

場合
55 とき。おり。「雨が降った―（には）中止する」▽概して、時（4）と同様に使う。法令文で、「場合」と「とき」とを重ねて用いるときは、前
提とする条件の大きい方に
41 物事の、その時に応じて分けて考えられる状態・事情。「―によっては」「万一の―」「問題を―分けして扱う」
3 (新語義) 事例。
1 (新語義) 可能性。

時間
58 ある時刻と他の時刻との間（の長さ）。時のへだたりの量。「この仕事は―がかかる」「―が切れる」「―の問題」（その物事の結着まで長くはかかるまいという情況にまで立ち至ったこと）▽多くは一日より短い場合について使う。

30 時間（1）の長さの単位。一時間は六十分。一日は二十四時間。
 9 日常語で、時刻。「帰りの一がおそい」「お一（＝刻限）です」「日本では昼ですが現地一では午後十時です」＜関連＞時・時刻・タイム・年月・歳月・月日・日月・時日・光陰・星霜・春秋・多年・積年・短時日・片時・半時・一時・一刻・一瞬・瞬間・瞬時・刹那・寸陰・寸時・つかの間・たまゆら
 3 空間と共に、物体界を成り立たせる基礎形式と考えるもの。普通、過去から未来に絶えず移り流れると考えている。▽↓とき（時）（1）。

意味
69 その言葉の表す内容。意義。「辞書を引けば一がわかる」
20 表現や行為のもつ価値。意義。「そんな事をしても一がない」
11 表現や行為の意図・動機。「どういう一でそんなことをしたのか」

電話
68 電話機による通話。「一をかける」「一を切る」
30 「電話機」の略。「一ボックス」
 2 （新語義）電話番号。電話回線。

一緒
82 ほかのもの・ことと併せて一つにする、または一つになるさま。「これも一に処理する」「君と一に行こう」「一になる」（一つに合わさる。特
に、夫婦になる）
18 同じであること。「趣味が一だ」▽正しくは「一所」だといわれる。

目
45 眼球・視神経から成る器官。
44 順序を表す時に添える語。「三番一の問題」「二つ一の角を曲がる」「十軒一」
 5 目（1）（ア）に見える姿・様子。「一ばかりよくて実質は悪い」
 2 不明。原文は「胸のない女性と３０代の太*目の巨乳の女性と恋人にするなら」
「い明度が低い緑系。あとは濃い*目のグレーもカッコいいと思いますよ。」
 2 ある物事に出会うこと。経験。体験。また、局面。「ひどい一を

2 《量・程度に関係する形容詞の語幹などに付いて》どちらかと言えば、その性質を持つ意を表す。「話を一に　打ち切る」「大き一な方を選ぶ」「細一の糸」「幾分早一に出掛ける」「控え一な人」

以前
87 ずっと前（の時）。「彼は一横浜に住んでいた」
13 それから前。⇔以後。

代
47 年齢の範囲を示す語。「台」に同じ。「十代の少年」
47 用をするためのもの。「舌代・糊代」
造・名かわりになるもの。商品や労力などを得たかわりに与えるもの。「お代を払う」「代価・代金・代償・地代・間代・足代・身代金」
3 造・名家督または統治権を相続して当主である間。「代がかわる」「先祖代代・譜代・歴代・末代・初代・聖代・累代・世代」
1 (切り間違い)
1 歴史上の長い時間の区分。「時代・上代・古代・近代・現代・当代」
地質時代の最も大きい区分。「古生代・新生代」
1 かわりになる。他にかわって仕事をする。「代理・代用・代作・代官・代診・代役・代弁・代数・代議士・代言人・名代・城代・所司代」

顔
60 頭部の前面。目や鼻や口がある所。「一を洗う」「一から火が出る」
38 顔つき。表情。「変な一をする」
容貌。「きれいな一」
《接尾語的に》…の様子。「得意一」「一」「わけ知り一」▽人の態度に言う。
2 人によく知られていること。「一が広い」
面目。体面。「一が立つ」「一をつぶす」
『一がさす』人に見られては具合の悪い状態になる。

系
87 系統だった分類。またその部門。「文学系・結晶系・六方晶系」

7 いとすじ。すじ。ひも。つながり。一つづきの関係をなすもの。「系図・系統・系譜・系列・世系・大系・体系・家系・直系・母系・傍系・太陽系」
6 体系（としてとらえられるもの）。システム。「公理系・測定系・力学系・神経系・人間機械系」

郵便
90 書状・はがき・小包などを集配・送達する通信制度。また、それによって送られる書状・はがき等。「―が届かない」
7 郵便局で取り扱うものに冠する語。「―貯金」「―年金」
3（新語義）郵便局

反応
78 生体が、刺激を受けた結果として起こす変化・活動など。「薬物―」「―がない」（手ごたえがない意にも）「冷たい―」
12（新語義）非生物による応答。
10 二種以上の物質の間に起こる化学的変化。「可逆―」