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

執筆者：藤井 丈明

指導教官：新納 浩幸

平成16年3月2日

目 次

図目次

表目次

第1章　序論

1.1　概要

本論文は自然言語処理の研究である語義識別における誤り原因の調査と on the fly の類似語判定法の提案について述べる．[4]

現在国際化社会が進み，インターネットを通して世界中の情報を得ることができる．そのため，機械翻訳における需要は上昇している．しかし日本語文章を解読する際，多義語の語義識別という問題に遭遇する．自然言語処理の研究において，機械による高精度な語義識別は求められ続けてきた．我々人間は，日常において当然のように多義語の語義識別を行ない，その単語の意味を識別しているが，現在用いられている機械学習により得られた学習規則では，人間ほどの高精度な語義識別はできないというのが現状である．

一般に，曖昧性をもつ語義の識別では，人間が識別容易な問題は機械にとっても容易であり，人間が識別困難な問題は機械にとっても困難である．しかし Senseval2 と呼ばれる多義語の曖昧性解消に関するコンテスト形式の国際会議において，日本語の多義語の曖昧性解消のタスクとして設定された辞書タスクのいくつかの単語では，人間が識別容易でも機械にとっては識別困難であったことが報告されている [1]．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．本研究ではこの6単語を用いて実験を行なう．

このような，人間が識別容易でも機械にとっては識別困難な多義語に着目し，その誤り原因を調査することは語義識別の精度向上が期待できる．

1.2　位置付け

ここでは，語義識別における誤り原因の調査を遠方単語の影響と未知語の影響の2つの面から行なった．遠方単語を考慮せずに人間が語義識別を行なった場合

でも，考慮した場合と同様，正解率が機械よりも大きく高い．このことから遠方単語の影響は小さいと考えられる．また，トレーニングデータに未知語が減るように作為的に事例を数個増やした場合，正解率が大きく向上する．このことから，未知語の影響が大きいと考えられる．未知語の影響をなくすために，on the fly の類似語判定法を提案する．そこではコーパスを用いて単語間の類似語を判定する．これを語義識別に応用する．

1.3 本論文の構成

本論文では始めに，多義語の語義識別についての説明を行なう．次に，従来より使用されてきた機械学習における語義識別の手法について述べる．その後に本研究で着目した，語義識別の誤り原因と考えられる遠方単語の影響，未知語の影響についてそれぞれの影響力を調査する．また，未知語の影響力を軽減させるための手法として， on the fly の類似語判定法の提案を行なう．on the fly の類似語判定法を語義識別に応用した実験の解説と結果を示し，その結果に対する考察を行なう．

第2章　多義語の語義識別

多義語の語義識別の例として，ここでは「核」という単語を用いる.「核」には物事の中心(となるもの)，物の中心の部分，細胞核，核兵器，草や木の芽生える種という意味がある．人間は文脈上から，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．これを多義語の語義識別という．今回使用した単語は以下の語義で示されている．[1]

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: 「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 2.4: 「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 2.5: 「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．「一現象」「一力」 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．「立法の一」

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　従来手法

ここでは，従来より使用されてきた語義識別における機械学習の手法について説明を行う．

3.1　Naive Bayes法

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，本研究では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．[1]

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．

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．

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

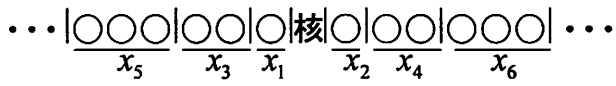
ここで x を構成する各要素は，多義語の前後にある素性を示している．

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．



前後の内容語を素性とする

図 3.1: 素性の説明

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

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．[5]

実装した Naive Bayes 法を用いて本研究で用いた単語の正解率を示す．

表 3.1: Naive Bayes 法を用いた正解率 (%)

単語	正解率
開発	61.0
核	70.0
精神	65.0
乗る	54.0
生まれる	69.0
かかる	54.0

```
if(直前の証拠　== “ の”) return class1;
if(直前の証拠　== “ は”) return class2;
if(直後の証拠　== “ に”) return class1;
...
return class1;
```

図 3.2: 決定リストの表現形式例

単語全体の正解率は，名詞 76.9 %，動詞 78.0 %となった．よって，本研究に用いた単語は全体と比べ正解率が低いということが分かる．

3.2 その他の手法

その他の手法を示す．[6] [7] [8]

3.2.1 決定リスト

決定リストとは従来より語義選択問題に適用されてきたクラス分類手法の一つである．

特定の辞書に基づいて意味タグ（語義）が付与されたコーパスであれば，品詞付けと同様の手法により語義付与が可能かも知れないが，意味タグつきコーパスを作成するコストは非常に大きい．そこで Yarowsky は，語義が付与された少数の初期データ（seed collocation）から，教師なし学習により語義選択のための確率モデルを作製する方法を提案した．その確率モデルが決定リストである．

決定リストは if-then のルールを学習する手法と言える．

決定リストの規則の表現形式は容易であり，直感的で分かりやすい構造をしている．語義の選択に影響を与えるための証拠（*evidense*）がリスト状に並んでおり，その順番は証拠の強さ（予測力）の順番で降順にソートされたものである．証拠としては，様々な単語共起の形（単語・品詞及びその位置関係）が考慮され，予測力はその証拠 $evidense_i$ のもとで，語義 $sense_a$ と語義 $sense_b$ が選ばれる確率との対数尤度比で表される．

$$\log\left(\frac{P(sense_a|evidence_i)}{P(sense_b|evidence_i)}\right)$$

テキスト中の単語の語義は，文脈中に存在する様々な証拠から総合的に判断するのではなく，最も予測力の高い一つの証拠に基づいて決定される．例えば表に英単語"plant"の語義選択に関する決定リストの例を示す．

表 3.2: "plant"における決定リストの例

共起単語（証拠）	予測力	語義
plant growth	10.12	LIVING
car(±k 語以内)	9.68	FACTORY
plant height	9.64	LIVING
union (±k 語以内)	9.61	FACTORY
equipment (±k 語以内)	9.54	FACTORY
assembly plant	9.68	FACTORY
nuclear plant	9.68	FACTORY
...	...	...

英単語"plant"には「植物 (LIVING)」と「工場 (FACTORY)」という２つの意味がある．この決定リストでは，"growth"や"height"という語が"plant"の直後にあれば「植物」を意味し，"car"や"union"が周辺 (±k 語以内) にあれば「工場」を意味するとしている．

3.2.2 サポートベクトルマシン法

サポートベクトルマシン法は，空間を超平面で分割することにより２つの分類からなるデータを分類する手法である．このとき，２つの分類が正例と負例からなるものとすると，学習データにおける正例と負例の間隔（マージン）が大きいものほどオープンデータで誤った分類をする可能性が低いと考えられ，このマージンを最大にする超平面を求めそれを用いて分類を行なう．通常，学習データにおいてマージンの内部領域に小数の事例が含まれてもよいとする手法の拡張や，超平面の線形の部分を非線形にする拡張（カーネル関数の導入）がなされたものが

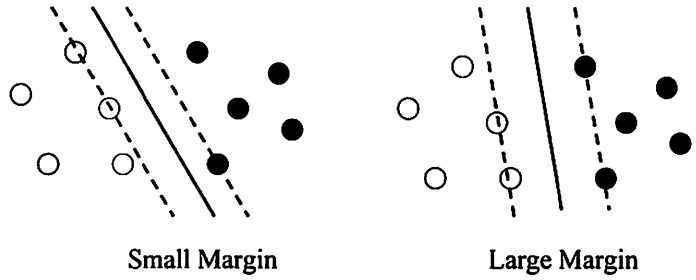


図 3.3: マージン最大化

用いられる．この拡張された方法は，以下の識別関数を用いて分類することと等価であり，その識別関数の出力値が正か負かによって二つの分類をすることができる．

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

$$b = -\frac{\max_{i,y_i=-1} b_i + \min_{i,y_i=-1} b_i}{2}$$

$$b_i = \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i)$$

ただし，$\mathbf{x}$ は識別したい事例の文脈（素性の集合）を，$\mathbf{x}_i$ と $y_i (i = 1, \cdots, l, y_i \in \{1, -1\})$ は学習データの文脈と分類先を意味し，関数 sgn は，

$$sgn(x) = \begin{cases} 1 & (x \geq 0) \\ -1 & (otherwise) \end{cases}$$

であり，また，各 α_i は以下の制約のもと，$L(\alpha)$ を最大にする場合ものである．

$$0 \geq \alpha_i \geq C (i = 1, \cdots, l)$$

$$\sum_{i=1}^{l} \alpha_i y_i$$

$$L(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j})$$

また，関数 K はカーネル関数と呼ばれ，様々なものが用いられるが，論文 [7] では以下の多項式のものが用いられている．

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d$$

C, d は実験的に設定される定数である．ここで，$\alpha_i > 0$ となる $\mathbf{x}_i$ は，サポートベクトルと呼ばれ，通常，$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$ の和をとっている部分はこの事例のみを用いて計算される．つまり，実際の解析には学習データのうちサポートベクトルと呼ばれる事例のみしか用いられない．

サポートベクトルマシン法は分類の数が 2 個のデータを扱うもので，通常これにペアワイズ手法を組み合わせて用いることで，分類の数が 3 個以上のデータを扱うことになる．

ペアワイズ手法とは，N 個の分類を持つデータの場合，異なる二つの分類先のあらゆるペア ($N(N-1)/2$ 個) を作り，各ペアことにどちらがよいかを 2 値分類器で求め，最終的に $N(N-1)/2$ 個の 2 値分類先の多数決により，分類先を求める方法である．

3.2.3 数量化 2 類

数量化 2 類（quantification theory type 2）は，「目的変数である場合」の解析手法の一つで，説明変量（外的基準）を予測する式を求めるものである．数量化 2 類では，説明変数のデータ形態がカテゴリーデータである．すなわち，カテゴリー変量の説明変量をもとに目的変量であるカテゴリーの帰属を予測する，又，カテゴリーにあてはまるかを判別する技法である．

カテゴリースコアの求め方

1) 外的基準と説明変数のクロス集計を行なう．

2) 以下の式を用いて計算を行なう．

$$\left(\frac{q_1(jk)}{n_1} + \frac{q_2(jk)}{n_2}\right) \times \frac{n_1 n_2}{n_1 + n_2}$$

3) 説明変数相互のクロス集計を行なう．

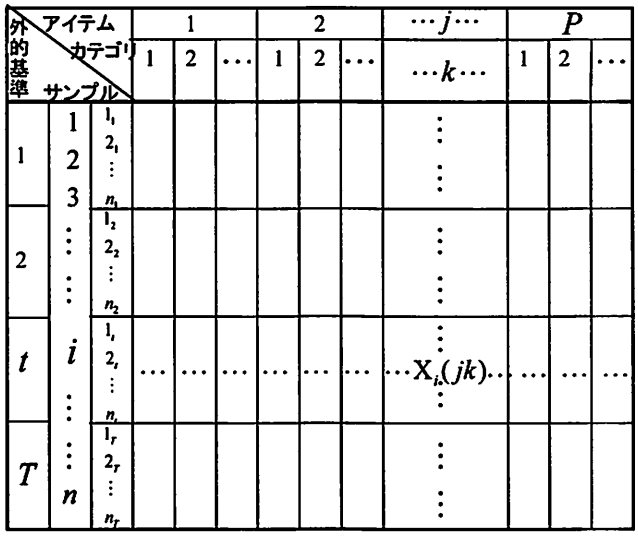


図 3.4: 数量化 2 類に適応するデータ

4) 以下の式を用いて計算を行なう.

$$f(jk, uv) - \frac{1}{n} n_{jk} \cdot n_{uv}$$

5) 求めるカテゴリースコアを次のように定義する.

$$a_{ij}(i = 1, \cdots, \text{アイテム数}, j = 1, \cdots, \text{カテゴリー数})$$

6) 連立方程式 $Xa = b$ を立て解く.

7) 連立方程式の解をカテゴリースコアに変換する.

・加重平均を算出する

加重平均はアイテムに対する各カテゴリーの回答者 × それにリンクする連立方程式の解でそのアイテムの加重平均が算出される.

・カテゴリースコアは, 連立方程式の解から加重平均を引くことで算出される

判別式

判別式は以下のように表すことができる.

$y =$ アイテム1のカテゴリースコア $+\cdots+$ アイテムnのカテゴリースコア

このようにして求められた値yをサンプルスコアと呼ぶ. また, サンプルスコアをもとに外的基準のどちらの郡に属するかを判断した結果を推定群という. これに対して事実, つまり同じサンプルを実際に調査して得られたデータを実績群という. 判別式の結果により, 対象がどの群に属するか判別できる.

第4章　誤り原因の特定

4.1　遠方単語

ここでは，文脈上における識別対象の多義語の前後3単語を周辺単語と呼び，周辺単語よりも離れた位置にある単語を遠方単語と呼ぶことにする．人間が正確に多義語の意味を識別できるのは，識別対象の多義語とその周辺単語だけではなく，文章全体を見ているからだとも考えられる．つまり遠方単語をも識別の判断材料としているからだと考えられる．一方，多くの機械学習の手法では，遠方単語を考慮せずに，識別対象の多義語を中心とした周辺の数単語だけを見て語義識別を行なっている．つまり遠方単語が人間と機械の差を生じさせている原因として考えられる．

ここでは人間が機械学習の手法と同様に，識別対象の多義語を中心とした周辺単語だけを見て，つまり遠方単語を考慮しなくても多義語の識別が容易かどうかを調査する．このことにより，遠方単語の影響の有無を確認することができる．

また遠方単語の影響を調査するため，実際に機械学習により得られた規則を用いて，人間と機械の正解率を比較する必要がある．ここでは機械学習の手法として， Naive Bayes 法を用いる．3人の人間が機械学習の手法と同様にして，遠方単語を見ずに語義識別を行なう．3人の正解率の平均を，Naive Bayes 法によって得られた正解率と比較する．結果を表に示す．

人間の方が機械よりも正解率が，ある程度，高くなっている．語義識別において，遠方単語の影響が大きいのであれば人間と機械の正解率はほぼ同程度となるはずである．しかし，人間が遠方単語を判断材料としなかった場合，その正解率にはある程度の差がある．これは人間が遠方単語を見て識別した場合と同様の傾向である．つまり人間が語義識別を行なう際には，遠方単語の影響は小さいと考えられる．

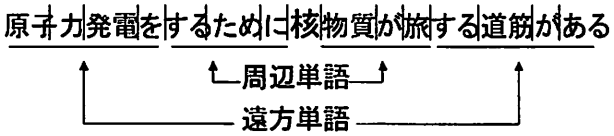


図 4.1: 遠方単語と周辺単語

表 4.1: 人間と機械学習による正解率 (%) の比較

単語	人間	機械学習
開発	76.3	65.0
核	97.3	72.0
精神	84.0	64.0
乗る	82.3	66.0
生まれる	97.3	71.0
かかる	85.0	66.0

4.2 未知語

対象とした6単語の中で人間が正解し，機械が不正解となったテストデータを調査した．その結果，人間が正解する場合でも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．

このことを確認するために，訓練データに作為的に事例を3個増やして学習を行なってみる．その事例とは，先に述べた問題となる未知語を含んだ文である．訓練データ内に事例を増やすことによって，テストデータ内の未知語の数を減少させることができる．具体的には以下の文である．

表 4.2: 追加した事例

単語	事例
かかる	・電話がかかった ・お目にかかった ・チームの命運がかかった重要な試合
開発	・石油開発への貢献が関わってくる ・油田開発への取り組み ・ソフト開発を行なう
精神	・立法の精神を活かす ・日本的精神とその現在 ・精神的，経済的独立支援
のる	・勢いにのる政党 ・脂がのった魚 ・電車にのって出かける
核	・日本の核にいる人物 ・地域を核にした生活 ・マイネルト核から大脳皮質への投射系に障害がある
うまれる	・芸術作品が生まれる時代 ・事態に動きが生まれる ・演奏の時，生まれる音

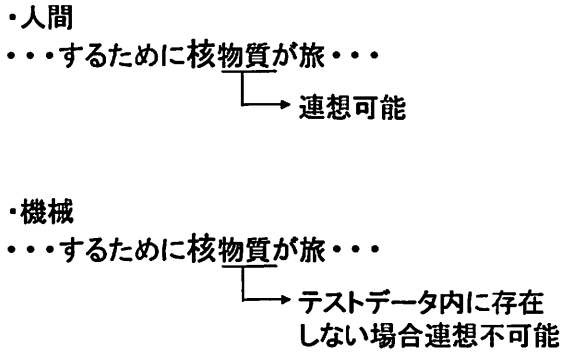


図 4.2: 語義の連想

訓練データ内に事例を増やす前と増やした後のテストデータに対する正解率を比較した．その結果を示す．

表 4.3: データ追加前後の正解率 (%) の比較

単語	追加前 (%)	追加後 (%)
開発	65	70
核	72	75
精神	64	68
乗る	66	71
生まれる	71	75
かかる	66	70

訓練データ内に事例を追加した後は，事例を追加する前よりもどの単語も正解率が3～5％向上した．このことから人間と機械の語義識別の正解率の差は，未知語の影響が大きいと考えられる．

第5章　on the fly の類似語判定

調査結果から，未知語の影響が大きいと考えられる．未知語は登録語数を増やすことで回避できそうだが，Zipf の法則によれば未知語は必ず出現するために，静的に単語を準備しておくアプローチには限界がある．

ここでは未知語の問題を解決するために，on the fly の類似語判定法を提案する．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．ここでは 新聞記事と Web をそれぞれコーパスに用いる．

まず基底となる単語を 100 単語選出した $(v_1, v_2, \cdots, v_{100})$．これは論文 [3] の結果を考慮してアドホックに選んだものである．基底となる単語を以下に示す．

未知語 w の特徴ベクトルを得るのに，“w v_i” をクエリにして コーパス を元に検索を行ない共起頻度 h_i を求める．そして未知語 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．

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

表 5.1: 基底の単語

単語				
決定	伊勢	信用組合	目標	確信
問題	促進	政界	故郷	一致
開催	一部	後期	回帰線	野茂
不審	年間	負け	幸い	文藝春秋
以上	平安	二郎	対策	状態
招致	女性	兵庫	航空	協同
連勝	学生	同日	浮島	住宅
主婦	捜査	私立	意識	脚光
画面	行政	影響	義援金	公示
新党	病死	課税	国有	大江
景気	本格	厚生省	実業	移植
運転	救急	分析	差別	資金
福利	人事	現金	章一郎	現場
緊急	現地	警察	敏夫	大会
事実	機関	井戸	個人	取材
東京	融資	朝鮮民主主義人民共和国	高温	日曜
特別	最高	準備	世界	葬儀
解散	安打	戦後	装置	実習
入院	搭載	私有	手作り	青葉
毎日新聞	入り	洗濯	決議	日本電信電話

5.1 新聞記事をコーパスに用いた場合

ここでは，未知語 w を特徴ベクトルとして表すための共起頻度 h_i を新聞記事から得る方法を試みる．用いたデータは95年度版毎日新聞1年間分である．ここでは共起の定義として，同一文章中に対象単語と基底の単語が出現していれば共起しているとする．共起頻度を得る手法として Access，C言語を用いる．

Access とは，マイクロソフト社が開発した「リレーショナルデータベース管理システム」という類のソフトウェアであり，テーブル，クエリ，フォーム，レポート，マクロの五つのオブジェクトを組み合わせ，一つのデータベースを作りあげる．

表 5.2: Access における各オブジェクトの説明

オブジェクト名	説明
テーブル	元となるデータ
クエリ	データの検索，抽出
フォーム	視覚的に理解しやすく，工夫を凝らした操作体系が可能
レポート	報告書としての印刷物作成
マウス	キーボードやマウスの一連の作業の自動化

まず，Access を用いてコーパスをデータベース化し，共起頻度を得られるフォームを作成した．これは，抽出条件を [$forms$]![フォーム名]![テキストボックス名] とすることにより作成できる．また，C言語を用いてプログラムを作成した．しかし，どちらの手法を用いても適当な共起頻度は得られなかった．その理由として，コーパスと基底の単語との解析結果が異なるということが考えられる．コーパスは形態素解析ソフト「茶筅」を用いて形態素解析を行なっているのに対し，基底の単語選出は，形態素解析ソフト「JUMAN」を用いて形態素解析を行ない、人間によって誤りを修正されたものを用いている．そのため適当な共起頻度を得ることはできなかった．また，Access を用いた場合，データ容量が1.7GB と大きすぎたため処理時間が非常にかかるという問題も発生した．

最適な共起頻度を得る方法として，新聞記事をコーパスにするのは適切ではないと考える．未知語を特徴ベクトルとして表すために，適当な共起頻度を得る必要がある．しかし，新聞記事をコーパスに用いた場合は全ての単語に対応できな

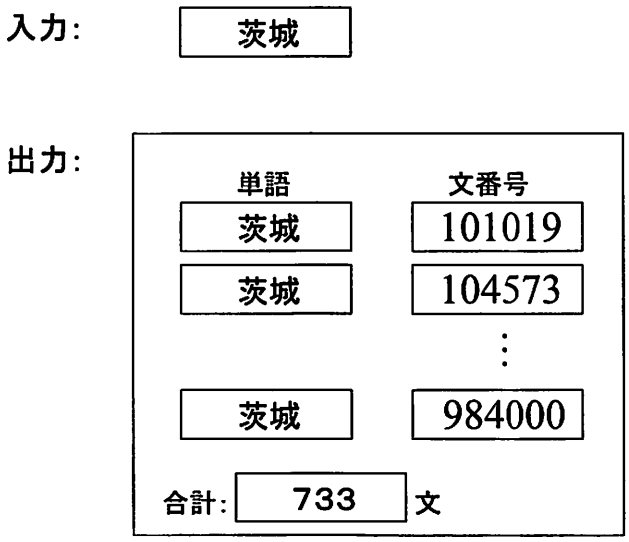


図 5.1: 作成したフォームによる入出力の例

いという問題がある.

5.2 Webをコーパスに用いた場合

ここでは，未知語 w を特徴ベクトルとして表すための共起頻度 h_i を Web から得る方法を試みる．検索エンジンである google を用いて，共起頻度を求める．ここで共起頻度は検索ヒット数として用いる．Web は情報量が膨大であるため，全ての単語に対しての適当な共起頻度を得ることができる．特徴ベクトル作成の際には，適当な共起頻度を得ることが重要であるため，本研究では，Web をコーパスに利用し共起頻度を得ることにする.

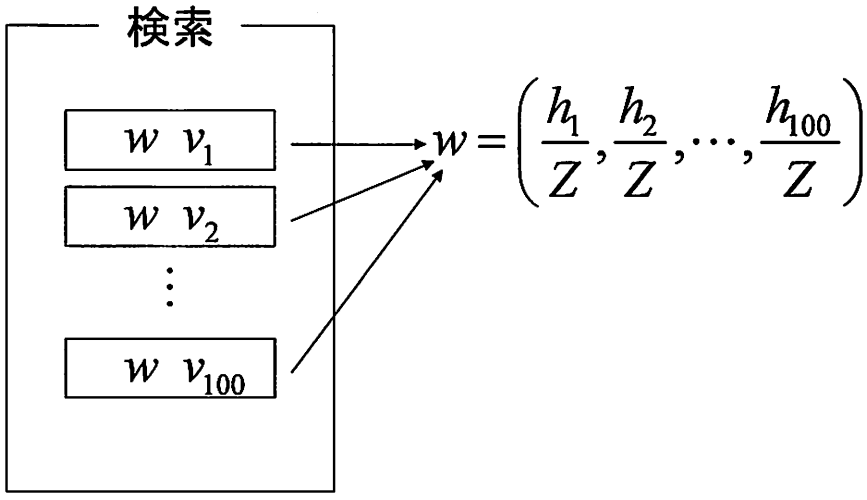


図 5.2: on the fly による特徴ベクトルの作成

第6章　実験

ここでは on the fly の類似語判定を語義識別に応用した実験を行なう．

6.1　実験方法

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 [2] 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．

ここでは「開発」に対して実験を行なった．

6.2　複合語を利用した識別

「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．前後を調べる場合や，重複を含むものもあるので，調査すべき未知語は 92 単語であった．調査した未知語を示す．

表 6.1: 調査した未知語

単語			
エイズワクチン	エンジン	コードネーム	サイド
システム	センター	ソフト	ソフトウエア
ニュータウン	プログラム	プロジェクト	ベイエリア
ミサイル	リゾート	レジャー	ワクチン
案件	営業	援助	会議
会社	核	環境	企画
危険	機	機構	技術
疑惑	共同	協会	協議
協力	計画	権	研究
原爆	公社	公団	口
抗がん剤	構想	今後	再
市	市街地	事業	時
時点	次第	自社	自主
主体	場	職能	新薬
人口	陣	推進	政策
政府	製品	石油	全般
総合	態勢	宅地	担当
段階	地域	中	停止
電源	途上	途上国	都市
土地	独自	熱	農村
農地	部門	別荘	本部
問題	薬品	油田	輸入
優先	利用	力	話

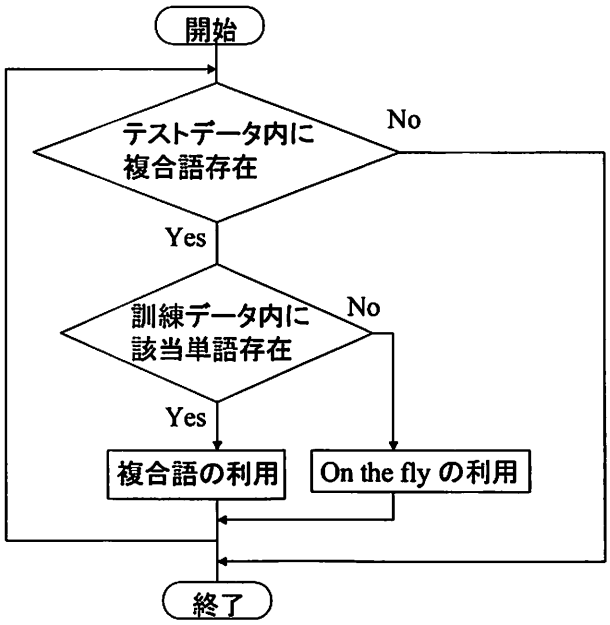


図 6.1: 実験の流れ

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．正解数を以下に示す．参考のためにこの 21 問に対する Naive Bayes での正解数も示す．

表 6.2: 複合語を利用した識別

手法	正解数	不正解数
Naive Bayes	16	5
複合語の利用	17	4

6.3 on the fly の類似語判定法を利用した識別

「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．正解数を以下に示す．参考のためにこの 50 問に対する Naive Bayes での正解数も示す．

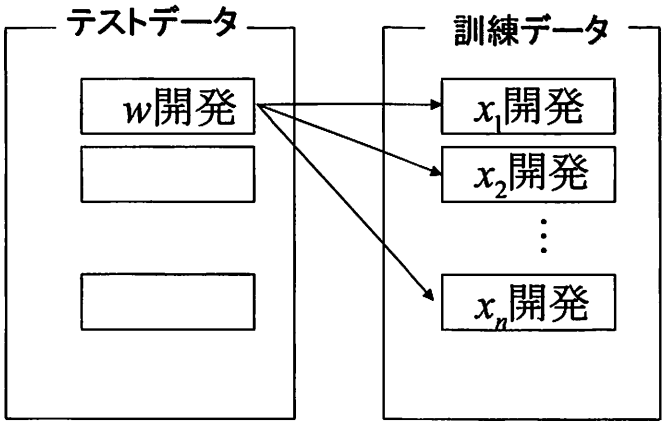


図 6.2: on the fly を用いた実験

表 6.3: on the fly の類似語判定を利用した識別

手法	正解数	不正解数
Naive Bayes	19	31
on the fly	25	25

第7章　考察

7.1　誤り原因

今回の実験で，遠方単語の影響が小さいことがわかった．これは機械学習の手法に遠方単語の素性を取り入れなくても，十分な語義識別が可能であることを示している．

また，未知語の影響が大きいこともわかった．語義識別の精度向上には未知語の解決が重要だと考えられる．通常はシソーラスの作成により未知語の問題の解決が図られるが，未知語は必ず出現するために，on the fly による未知語の解決が望ましい．

7.2　on the fly の類似語判定法

本方式を用いたことで「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．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．

on the fly の類似語判定法を用いた手法は Web を利用しているために未知語の解決が図れる可能性はあるが，まだいくつかの課題がある．

第1に，基底の単語の設定が挙げられる．今回は基底の単語として，アドホックに選定した 100 単語を用いたが，この 100 単語が基底として適しているかどうかは更に実験を重ねて確認する必要がある．

第2に，処理時間の問題が挙げられる．ここでは google を用いて検索を行なっているが，1つの未知語に対して 100 回の検索を行なっている．このために処理時間がかかっている．次元数を増やすと更に処理時間がかかる．今後，処理時間を減らす方法も考える必要がある．

第3に，応用性の問題が挙げられる．今回は「開発」という一単語の名詞のみで実験を行なったが，識別対象の多義語が動詞であった場合は，今回のように複合語となる単語から語義識別を行なう方法を用いることはできない．そのため，複合語の代わりに識別対象の多義語の周辺単語を調べ，その共起頻度を得て on the fly の類似語判定を行なえば良いと考えている．これを確認し，他の品詞にも on the fly の類似語判定が有用であるかを確かめる必要がある．

第8章　おわりに

本論文では語義識別の誤り原因の調査と on the fly の類似語判定の提案を行なった．語義識別の誤り原因として，遠方単語の影響および未知語の影響の2つの面から調査を行なった．結果，遠方単語の影響は小さく，未知語の影響は大きいことが確認できた．

またここでは未知語の問題を解決するために on the fly の類似語判定法を提案した．多義語の「開発」を用いた実験ではその効果を確認できた．

今後の課題としては，基底の単語の選定や処理時間の軽減の問題がある．また動詞への拡張や大規模な実験なども必要である．

第9章　謝辞

本研究の遂行及び論文の作成に多大な御助言及び指導を賜わった 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 紺野 憲一 氏（茨城大学大学院修士課程），大城 亜里沙 氏（茨城大学工学部システム工学科４回生），結城 隆 氏（茨城大学工学部システム工学科４回生），脇坂 恭志郎 氏（茨城大学工学部システム工学科４回生）及び，岩崎研究室の 時田 陽一 氏（茨城大学工学部システム工学科４回生）にも深く感謝します．

付 録A　プログラムソースリスト

```
/*****************************/
クラス数とその総頻度数
/*****************************/


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

#define LINESIZE  256

#define TBSIZE    150
#define NAMESIZE  50
#define WORDSIZE  100
void quit(char *);

int main(int argc, char *argv[])
{
  FILE *f1;
  char buf[LINESIZE], nametable[TBSIZE][LINESIZE],
word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE],
str[TBSIZE][LINESIZE],*c,*p;
//nametable[][] は1次元配列の集合
 int  r,i,j,k,l,flag;
```

```
  int  counttb[TBSIZE], lastid = 0;
int counter=0;


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

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

     i=j=k=flag=0;
     strcpy(word1,"");
     strcpy(word2,"");
     strcpy(word3,"");

      while(buf[j]!='\0')
       {
       word2[i]=buf[j];
       if(word2[i]==' ')
          {
          word2[i+1]='\0'; //文字列の最後に\0 を追加
          i=-1;
          flag=1;

         strcpy(word1,word2);
          }
        if(word2[i]=='\n' && flag==1)
         {
        flag=0;
        counter++;

        strcpy(word3,word1);
```

```
          i=-1;
           }
         i++;
         j++;

       }
  c=word3;

 for(k=0;k<lastid;k++)
        {
         if(c==NULL) break;
         if (strcmp(str[k],word3) == 0)
            {
           counttb[k]++;
           break;
             }
         }

   if(k==lastid)
         {
          strcpy(str[k],word3);

          counttb[k]=1;
           lastid++;
         }


   }
for(l=0;l<lastid;l++)
 printf("%s %d個\n%d番目\n",str[l],counttb[l],l+1);
printf("\n全部で%d個ある. \n",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);
}
```

```
/*****************************/
素性とクラスの組み合わせに分割
/*****************************/

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

#define LINESIZE  256

#define TBSIZE    150
#define NAMESIZE  50
#define WORDSIZE  200
void quit(char *);

int main(int argc, char *argv[])
{
  FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],
word2[WORDSIZE],word3[WORDSIZE],word4[WORDSIZE],
str[TBSIZE][LINESIZE],*p;
//nametable[][] は1次元配列の集合
 int  r,i,j,k,l,flag;
  int  counttb[TBSIZE], lastid = 0;

  if(argc != 2) quit(" 引数の数が違う prog datafile ");
  if((f1 = fopen(argv[1],"r")) == NULL) quit("ファイルが開けない");
  while(fgets(buf,LINESIZE,f1) != NULL) {

   i=j=flag=k=0;
   strcpy(word1,"");
   strcpy(word2,"");
```

```
strcpy(word3,"");

   while(buf[j]!='\0')
    {
     word1[k]=buf[j];
     if(word1[k]==' ')
      {
       word1[k+1]='\0';
       strcpy(str[i],word1);
       i++;
        strcpy(word3,word1);
       flag=1;
       k=-1;
       }
      if(word1[k]=='\n'&& flag==1)
        {
         flag=0;

         strcpy(word2,word3);
         for(l=0;l<i;l++)
           {
           if(strcmp(word2,str[l])==0) continue;
           printf("%s %s\n",str[l],word2);
            }
         k=-1;
         }
       j++;
       k++;

       }
```

```
 }

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

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

```
/*****************************/
素性のクラス毎の頻度及び総頻度
/*****************************/

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

#define LINESIZE  2000

#define TBSIZE    1000
#define NAMESIZE  50
#define WORDSIZE  200
void quit(char *);

int main(int argc, char *argv[])
{
  FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],
word2[WORDSIZE],word3[WORDSIZE],word4[WORDSIZE],
str[TBSIZE][LINESIZE],str2[TBSIZE][LINESIZE],*p;
//nametable[][] は1次元配列の集合
 int  r,i,j,k,l,flag,a,b,c,count[TBSIZE];
  int  counttb[TBSIZE], lastid = 0;

  if(argc != 2) quit(" 引数の数が違う prog datafile ");
  if((f1 = fopen(argv[1],"r")) == NULL) quit("ファイルが開けない");
   strcpy(word1,"");
   strcpy(word2,"");
   strcpy(word3,"");
   strcpy(word4,"");
   a=i=j=l=flag=b=c=0;
```

```
while(fgets(buf,LINESIZE,f1) != NULL) {
sscanf(buf,"%s %s %s",word1,word2,word3);



 for(i=0;i<lastid;i++)
 {
  if(strcmp(word2,str[i])==0)
    {
    count[i]=count[i]+atoi(word1);
     a=0;
     break;
     }
   }

    if(i==lastid)
   {
   strcpy(str[i],word2);

   lastid++;
   count[i]=atoi(word1);
   }

  strcat(str2[i],word3);
strcat(str2[i]," ");
strcat(str2[i],word1);
strcat(str2[i]," ");
//  printf("%s %d番目\n",str2[i],i);
```

```
 }
for(b=0;b<lastid;b++)
printf(" %s 総頻度%d %s \n",str[b],count[b],str2[b]);
  if((r = fclose(f1)) == -1) quit("ファイルが閉じれない");
}

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

```
/*****************************/
それぞれの素性毎のクラスと計算結果算出
/*****************************/

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

void quit(char *);
int skip_space(char [],int);
int get_string(char [],char [],int);
int get_integer(char [],int *,int);
double keisan(int x, int y);
int main(int argc, char *argv[])
{
   FILE *f1;
   char buf[2000],word[100],word1[100],word2[100];
   int  r,n,sum,h,pos;
   double kai;
   kai=0;
  if(argc != 2) quit("引数の数が違う");
  if((f1 = fopen(argv[1],"r")) == NULL) quit("ファイルが開けない");

  while(fgets(buf,2000,f1) != NULL) {
     pos = 0;
     pos = skip_space(buf,pos);
     pos = get_string(buf,word,pos);
     strcpy(word1,word);
     pos = skip_space(buf,pos);
     pos = get_integer(buf,&sum,pos);
//printf("第1列  %s\n",word);   printf("その数  %d\n",sum);
     while(pos != -1) {
        pos = skip_space(buf,pos);
```

```
        pos = get_string(buf,word,pos);
        pos = skip_space(buf,pos);
        pos = get_integer(buf,&h,pos);
       if(pos==-1) break;
//printf(" 次の列 %s\n",word);   printf("その数  %d\n",h);

kai=keisan(sum,h);

printf("%s %s %f\n",word1,word,kai);
     }
sum=h=0;
strcpy(word,"");
strcpy(word1,"");
  }
  if((r = fclose(f1)) == -1) quit("ファイルが閉じれない");
}

int skip_space(char buf[],int pos)
{
   while(buf[pos] == ' ') pos++;
   return pos;
}


int get_string(char buf[],char word[],int pos)
{
   int i = 0;
   char c;

   if (pos == -1) return -1;
   if (buf[pos] == '\n') return -1;
   for(     ;((c = buf[pos]) != ' ') && (c != '\n');pos++) {
```

```
      word[i] = c; i++;
   }
   word[i] = '\0';
   return pos;
}


int get_integer(char buf[],int *h,int pos)
{
   int i = 0;
   char c,num[10];

   if (pos == -1) return -1;
   if (buf[pos] == '\n') return -1;
   for(      ;((c = buf[pos]) != ' ') && (c != '\n');pos++)
  {
      num[i] = c; i++;
   }
   num[i] = '\0'; *h = atoi(num);
   return pos;
}

double keisan(int x, int y)
{
double a;
a=(y+0.1)/(x+100);
return a;
}

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

```
/*****************************/
データベース作成
/*****************************/

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

#define LINESIZE 1000
void quit (char *);

int main(int argc, char *argv[])
{
  FILE *f1,*fopen();
  int r,i,j,k,flag1,flag2;
  char line[LINESIZE],clm1[LINESIZE],
clm2[LINESIZE],clm3[LINESIZE],str[LINESIZE],*p;
  DBM *mydb;
  datum key,content;



  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１が開けない");
  mydb = dbm_open(argv[2],(O_RDWR|O_CREAT), 0640);
  while(fgets(line,LINESIZE,f1) != NULL){

  i=j=k=flag1=flag2=0;
  strcpy(clm1,"");                    //初期化
  strcpy(clm2,"");
```

```
  strcpy(clm3,"");
  strcpy(str,"");

      while(line[j]!='\0')
    {
      str[i]=line[j];
      if(line[j]==' ') flag1++;      //flag1== の数
      if(flag1==2)
        {
         str[i]='\0';
         strcpy(clm1,str);
         i=-1;                          //最後で i++されるので i=-1
         flag2=1;
         flag1=0;
         }
   if(flag2==1 && str[i]=='\n')
         {
         str[i]='\0';
         strcpy(clm3,str);
          }

      i++;
      j++;
    }


      key.dptr = clm1;
      key.dsize = strlen(clm1);
      content.dptr = clm3;
      content.dsize = strlen(clm3);
      if(dbm_store(mydb,key,content,DBM_INSERT) < 0)
quit("登録できない");
```

```
  }

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

void quit(char *s)

{
 printf(s); putchar('\n');
 exit(1);
}
```

```
/*****************************/
データベース検索
/*****************************/


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

#define LINESIZE 1000
void quit (char *);

int main(int argc, char *argv[])
{
  FILE *f1,*fopen();
  int r,size,i,j,k,flag;
  char line[LINESIZE],clm1[LINESIZE];
  DBM *mydb;
  datum key,val;


  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);
  while(fgets(line,LINESIZE,f1) != NULL){
      strcpy(clm1,"");                    //初期化
       j=k=flag=0;

     while(line[j]!='\n')
     {
```

```
        clm1[k]=line[j];
         k++;
         j++;
         }
         clm1[k]='\0';
         key.dptr = clm1;
         key.dsize = strlen(clm1);
         printf("%s ",clm1);
         val = dbm_fetch(mydb,key);
         size = val.dsize;
         for(i=0; i<size; i++)putchar((val.dptr)[i]);
         putchar('\n');
    }

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

void quit(char *s)

{
 printf(s); putchar('\n');
 exit(1);
}
```

```
/*****************************/
素性のクラス毎に分ける
/*****************************/


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

#define LINESIZE  256

#define TBSIZE    150
#define NAMESIZE  50
#define WORDSIZE  200
void quit(char *);

int main(int argc, char *argv[])
{
  FILE *f1;
  FILE *f2;

  char buf1[LINESIZE],buf2[LINESIZE],
word1[WORDSIZE],word2[WORDSIZE],word3[WORDSIZE],
word4[WORDSIZE],word5[WORDSIZE],str[TBSIZE][LINESIZE],*p;
//nametable[][] は1次元配列の集合
 int  r,i,j,k,l,flag,counter;
  int  counttb[TBSIZE], lastid = 0;

  if(argc != 3) quit(" 引数の数が違う prog datafile ");
  if((f1 = fopen(argv[1],"r")) == NULL)
quit("ファイルが開けない");
  if((f2 = fopen(argv[2],"r")) == NULL)
```

```
quit("ファイルが開けない");

i=j=k=l=flag=counter=0;
     strcpy(word1,"");
     strcpy(word2,"");
     strcpy(word3,"");

//class のファイル処理
  while(fgets(buf2,LINESIZE,f2) != NULL) {
 sscanf(buf2,"%s %s",word1,word2);
 strcpy(str[i],word1);

  //printf("%s\n",str[i]);
  i++;
 }

//TEST のファイル処理
 while(fgets(buf1,LINESIZE,f1) != NULL) {
  j=k=0;
     strcpy(word4,"");
     strcpy(word3,"");

     while(buf1[j]!='\0')
       {
        word3[k]=buf1[j];
         if(word3[k]==' ' && flag==0)
         {
          strcpy(word3,"");
          flag=1;
          k=-1;
         }
         if(word3[k]==' ' && flag==1)
```

```
            {
             word3[k]='\0';
             strcpy(word4,word3);
             for(l=0;l<i;l++)
             printf("%s %s\n",word4,str[l]);
             strcpy(word3,"");
             k=-1;
             }
            if(word3[k]=='\n' && flag==1)
            {
          flag=0;
         k=-1;
          strcpy(word3,"");
             }
           j++;
           k++;
             }


 }


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

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

```
/*****************************/
各クラスの計算結果
/*****************************/

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

#define LINESIZE  256

#define TBSIZE    150
#define NAMESIZE  50
#define WORDSIZE  200
void quit(char *);

int main(int argc, char *argv[])
{
  FILE *f1;
  char buf[LINESIZE],word1[WORDSIZE],
word2[WORDSIZE],word3[WORDSIZE],
word4[WORDSIZE],str[TBSIZE][LINESIZE],*p;
//nametable[][] は1次元配列の集合
 int  r,i,j,k,l,flag;
  double counttb[TBSIZE];
  int lastid = 0;

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

  while(fgets(buf,LINESIZE,f1) != NULL) {
  i=j=flag=k=0;
   strcpy(word1,"");
```

```
   strcpy(word2,"");
   strcpy(word3,"");

   sscanf(buf,"%s %s %s",word1,word2,word3);

   if(word3[0]=='\0') strcpy(word3,"0");
  //  printf("%s %s %s\n",word1,word2,word3);


 for(k=0;k<lastid;k++)
        {
         if (strcmp(str[k],word2) == 0)
            {
           counttb[k]=counttb[k]+atof(word3);
           break;
             }
         }

   if(k==lastid)
         {
          strcpy(str[k],word2);

          counttb[k]=atof(word3);
           lastid++;
          }
 }
for(l=0;l<lastid;l++) printf("%s %lf\n",str[l],counttb[l]);
  if((r = fclose(f1)) == -1) quit("ファイルが閉じれない");
}

void quit(char *s)
{
```

```
  puts(s); exit(1);
}
```

関連図書

[1] 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).

[2] 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).

[3] 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, to appear, (2004).

[4] 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, to appear, (2004).

[5] 新納浩幸：“EM アルゴリズムを用いた教師なし学習の日本語翻訳タスクへの適用”，自然言語処理, Vol.10, No.3, pp.61-73 (2003).

[6] 野澤洋一，：“決定木による同音異義語の誤り検出とその修正” (2003).

[7] 村田真樹，内山将夫，内元清貴，馬青，井佐原均：“SENSEVAL2J 辞書タスクでの CRL の取り組み -日本語単語の多義性解消における種々の機械学習手法と素性の比較-”，自然言語処理, Vol.10, No.3, pp.115-132 (2003).

[8] 菅民郎：“多変量解析の実践”，現代数学社 (1993).