逆トピックワードを利用した
外れ値文書検出

情報工学科

平成24年2月10日

執筆者：真下飛瑠 (08T4083T)
指導教員：新納浩幸 准教授

目 次

表 目 次

図 目 次

第1章　はじめに

1.1　概要

文書セットに対してクラスタリング等の解析を行う場合、外れ値にあたる文書を予め検出しておくことは重要である。ここではある事柄に関連した文書セットを対象に、外れ値文書の検出を行う。

外れ値文書の検出を行う場合、データマイニング分野の外れ値検出の手法を利用するのが一般的である。また外れ値検出手法は多数存在する [4]。しかし、どの手法にも共通した問題点として、ある同類の外れ値がある程度の個数出現した場合、それらを外れ値として検出できないということがある。通常、外れ値は他のデータからの距離が離れていたり、外れ値近辺の密度が低かったり、外れ値の生成確率が低いなどの性質があり、外れ値検出手法はそれらを手がかりに外れ値を検出する。ところが、ある同類の外れ値がある程度の個数出現した場合、上記のいずれの性質も満たさない状況が生じために、検出に失敗する。

本論文では上記の問題点の解決を主な目的とする。そのためにここでは Bekkerman が提案した関連文書検出の手法を応用する [2]。Bekkerman は対象の文書セットとは別に一般の巨大コーパスを用いて、文書セットの話題と関連度の高いトピックワードを選出し、そのトピックワードを利用して文書セットから関連度の高い文書を順に検出する手法を提案した。ここでは、その手法の一般の巨大コーパスと文書セットを逆に設定することで、一般の文書では頻出するが対象の文書セットでは現れづらい逆トピックワードを選出することで、外れ値文書検出を行う。

実験では、一般の巨大コーパスとして新聞記事３年分を利用する。ここからあるキーワードに関連した文書セットを取り出し、それらに人為的に外れ値文書を混在させ、文書セットからそれら外れ値文書を検出できるかどうかの実験を行った。提案手法の他、外れ値検出手法として LOF [3] と One Class SVM [1] を用い、提案手法の有効性を示す。

1.2 本論文の構成

第 2 章では、Bekkerman が提案した関連文書の検出手法について述べる。まず Max-KL アルゴリズムの概要を説明し、その問題点とそれを応用した One Class Clustering について説明する。

第 3 章では、OCC を元にした、非関連文書を検出するための逆トピックワードについて述べる。

第 4 章では、本論文の実験で結果の比較として用いる、LOF、One Class SVM の概要を述べる。

第 5 章では OCC を利用した非関連文書検出の実験を行い、第 6 章で考察、第 7 章の結論へと進む。

巻末には、本実験で使用したプログラムのソースリストを添付する。

第2章　トピックワードによる関連文書検出

Bekkerman が提案した関連文書検出手法について述べる。Bekkerman はこの手法を One Class Clustering(OCC) と呼んでいるので、ここでも OCC と呼ぶ。OCC は文書セットから話題 (=トピック) を見つけ出し、それに関連した文書を検出する手法である。このような手法はかねてから開発されており、テキスト要約や文書クラスタリング、スパムフィルタリングなどに利用されてきた。

OCC の特徴として、トピックワードがある。トピックワードとは文書のトピックを負う役割を持った単語である。OCC では単語集合の表現下で文書を扱うため、文脈は考慮されない。つまり、どのような単語がどの程度出現したか、がその文書の特徴となるのである。そのような役割が高い、その文書の象徴となるような単語がトピックワードである。出現した単語によって特徴が決まるというのは、文書のまとまりである文書セットについても同様である。

OCC のアルゴリズムはまず文書セットからその文書セットのトピックワードを求め、そのトピックに関連した文書を選び出す、というものである。例えばある文書セットからトピックワードとして「ベイジアン」「分類」「カーネル」などの単語が求まった場合、その文書セットは機械学習に関する文書をトピックとしている可能性が高い。よって関連文書もそれに近いものを選び出すのである。

しかし、文書にはトピックワードとなる単語と、一般的によく使われる、トピックワードではない単語が混在している。そのため、OCC を実現するには以下の問題を解決する必要がある。

1. どのような単語をトピックワードとするか
2. どうやってその単語を選出するか
3. そのトピックワードを使い、どのように関連文書を検出するか

2.1 指標 ρ の導入

問題点 1、2 については、ある単語 w のトピックへの関連度を示す指標として、以下で定義する ρ を導入する。ここで $p(w)$ は対象の文書セット (D) 中での w の出現確率、$q(w)$ は一般の巨大コーパス中での w の出現確率である。

$$\rho(w) = \frac{p(w)}{q(w)}$$

トピックワードを求めるには、単純に単語の出現回数を数えればよいというわけではない。一般的な文章には、数字や「日」「月」などの、日常的によく使われる単語が含まれているからである。このような単語はいくら出現回数が多くても文書の話題そのものとは関連が薄く、トピックワードとはなりづらい。

文書セット D の全単語からなる集合を G、そのトピックワードによる集合を R としたとき、$|G| \gg |R| \gg 0$ ならば、トピックワードはそうでないものよりも頻繁に出現する。ρ の値は一般には使われにくく、しかし対象の文書セット中では頻出する単語において高くなる。そのような単語は、文書のトピックに関連する可能性が高い。よって、$\rho(w)$ の値の高い単語 w をトピックワードとして選ぶことにする。

これを導入することで、次節の Max-KL アルゴリズムの問題点を解決することができるようになる。

2.2 Max-KL アルゴリズム

問題 3 については基本的に、Max-KL アルゴリズムを用いる。Max-KL アルゴリズムは関連文書を見つけるためのアルゴリズムであり、「文書の長さが同じ」という条件下では、関連文書を見つけるのに最適と言われている。

文書セット D の“ 話題性 ”は次式を計算することによって表される。

$$\begin{aligned} KL(p \parallel q) &= \sum_{w \in G} p(w) log \frac{p(w)}{q(w)} \\ &= \sum_{d \in D w \in G} p(d, w) log \frac{p(w)}{q(w)} \end{aligned}$$

名前からも分かるとおり、KL ダイバージェンスを用いている。KL ダイバージェンスはカルバック・ライブラー情報量とも言われるもので、2 つの確率分布

の差異を計る尺度である。この場合、$p(w)$ を基準に、$q(w)$ がどの程度違っているかを表している。これにより、文書間の類似度を求めるのである。

文書 d のトピックへの関連度は以下の式を計算することで計る。

$$KL_d(p \parallel q) = \sum_{w \in G} p(d,w) log \frac{p(w)}{q(w)}$$

求めている関連文書はこの値が高いものと思われるので、単純に

1. 文書を KL ダイバージェンスの大きさに従い降順に並べる
2. 上位 k 個を選ぶ

ということをすればよい。

先に述べたとおり、Max-KL アルゴリズムは同じ長さを持った文書に対しては最適であるが、通常の文書は長さが異なるものが大半である。そこで長さの異なる文書にも適用できるように考慮されたのが OCC である。

2.3 One Class Clustering

OCC では、トピックワードではない単語は無視して計算している。つまり、文書に非関連な話題が混じっていようと、トピックに関連する部分が含まれているならば関連文書として扱っている。最初にトピックワードによる集合 R を求め、次に文書から集合 R に使われていない単語を削除する。そして、関連文書を求めるのである。

まとめると、OCC は次のようなアルゴリズムとなっている。

1. 対象文書セットの全単語に対して $\rho(w)$ を計算する
2. $\rho(w)$ の値が高い単語上位 m 個をトピックワードとする
3. 各文書をトピックワードからなる単語集合として表現しなおす
4. 各文書の KL ダイバージェンスを計算する
5. KL ダイバージェンスの高いものを関連文書とする

ある文書 d のトピックワードを元にしたKLダイバージェンスは、次式から求まる。

$$\begin{aligned} KL_d^r(p \parallel q) &= \sum_{w \in R} p'(d,w) log \frac{p(w)}{q(w)} \\ p'(d,w) &= \frac{p(d,w)}{\sum_{w \in R} p(d,w)} \end{aligned}$$

$p'(d,w)$ は文書とトピックワードの結合分布である。

第3章　逆トピックワードの設定

OCC では関連文書の検出に重きを置いていたが、これを利用した外れ値文書の検出を考える。ある話題に沿って集められた文書セットがあり、その文書セットに外れ値となる文書が紛れ込んでいた場合、その集合の単語にはトピックワード、一般的な単語、その文書セット内に限り使われづらい単語の三種類がある考えられる。この文書セット内で使われづらい単語を用いれば、そこから非関連文書を検出できるのではないかと考えた。このような単語をここでは逆トピックワードと呼ぶことにする。

逆トピックワードを求めるため、ρ を計算する際の「一般の巨大コーパス」と「与えられた文書セット」を逆にすることを考える。つまり、以下の値を考える。

$$\rho(w) = \frac{q(w)}{p(w)}$$

$q(w)$ が分子にきているので一般の文書で使われやすい単語ほど ρ 値は高くなる。また $p(w)$ が分母となっているので、その文書セットで使われにくい単語ほど ρ 値は高くなる。つまり、この値が高い単語は、一般的な文書には頻出するが、文書セット中では使われづらい単語となる。以前の指標 ρ は単語のトピックへの関連度を表すが、この場合は逆となり、トピックへの非関連度を表すようになるのである。

本論文では OCC で使われた ρ を上記の ρ で置き換える、つまりトピックワードを逆トピックワードに替えて OCC を実行するとによる、文書セットのトピックとは関連度の低い文書の検出を提案する。

トピックワードと逆トピックワードの具体的な例として、今回の実験で用いた「災害・死者」文書セットにおけるトピックワード、逆トピックワード各上位 10 個を表 3.1 に示す。これは外れ値文書を混ぜる前のものである。

表 3.1: 「災害・死者」文書セットの (逆) トピックワード

トピックワード	逆トピックワード
倒	表示
検視	戦
流言	改革
ＦＥＭＡ	大会
神戸海洋気象台	派
烈震	監督
高潮	連続
ノースリッジ	女子
圧死	系
がけ崩れ	捜査

第4章　外れ値文書検出

提案手法の有効性を示すために、一般の外れ値検出手法として LOF と One Class SVM を試し、本手法と比較する。本章ではこれら手法について概説する。

4.1　LOF

外れ値検出の手法は距離ベースの手法、密度ベースの手法、クラスタリングベースの手法等に分類できるが、LOF は密度ベースの代表的な手法である。データの近傍の密度を利用することで、そのデータの外れ値の度合いを測り、その値によって外れ値を検出する。図 4.1 がその簡単な考え方を図示したものである。円は各データから 3 番目に近いデータへの距離を半径としている。点 A の円は他のデータと円と比べその半径がはるかに大きい。そのため、外れ値を判断されるのである。

LOF におけるデータ $x \in D$ における外れ値の度合いを $LOF(x)$ と表記する。ここで D はデータ全体の集合である。$LOF(x)$ を定義するために、いくつかの式を定義しておく。まず $kdist(x)$ は x に対する k 距離 と呼ばれる値で、以下の条件を満たすデータ $o \in D$ との距離 $d(x, o)$ として定義される。

1. 少なくとも k 個のデータ $o' \in D \setminus \{x\}$ に対して$d(x, o') \leq d(x, o)$が成立する。
2. 高々$k-1$ 個のデータ $o' \in D \setminus \{x\}$ に対して$d(x, o') < d(x, o)$が成立する。

直感的には、上記のデータ o はデータ x からの k 番目に近いデータとなる。データ x から同じ距離を持つデータが複数存在する場合を考慮して、上記のようなテクニカルな定義になっている。

次に $kdist(x)$ を利用して、$N_k(x)$、$rd_k(x, y)$ 及び $lrd_k(x)$ を以下のように定義する。

$$N_k(x) = \{y \in D \setminus \{x\} | d(x, y) \leq kdist(x)\}$$

$$rd_k(x, y) = \max\{d(x, y), kdist(y)\}$$

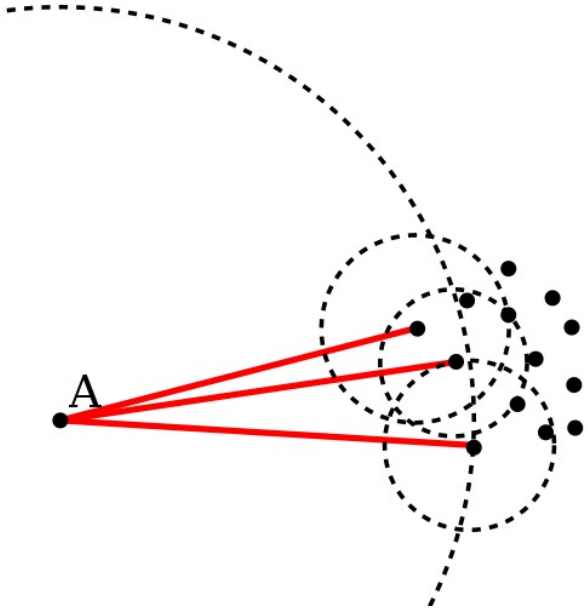

図 4.1: LOF の考え方

$$lrd_k(x) = \frac{|N_k(x)|}{\sum_{y \in N_k(x)} rd_k(x, y)}.$$

これらの式を用いて、$LOF(x)$ は以下で定義される。

$$LOF(x) = \frac{1}{|N_k(x)|} \sum_{y \in N_k(x)} \frac{lrd_k(y)}{lrd_k(x)}$$

4.2 One Class SVM

One Class SVM(Support Vector Machine) はν−SVM を利用した教師なしの外れ値検出手法である。すべてのデータは +1 のクラスに属し、原点のみが −1 のクラスに属するとして、ν−SVM を使って 2 つのクラスを分離する超平面を求める。Gaussian カーネルを用いると入力空間で他のデータから孤立しているデータは原点付近に写像される。その性質を利用し、−1 のクラス側に属するデータを外れ値とするのである。図 4.2 にその様子を示す。

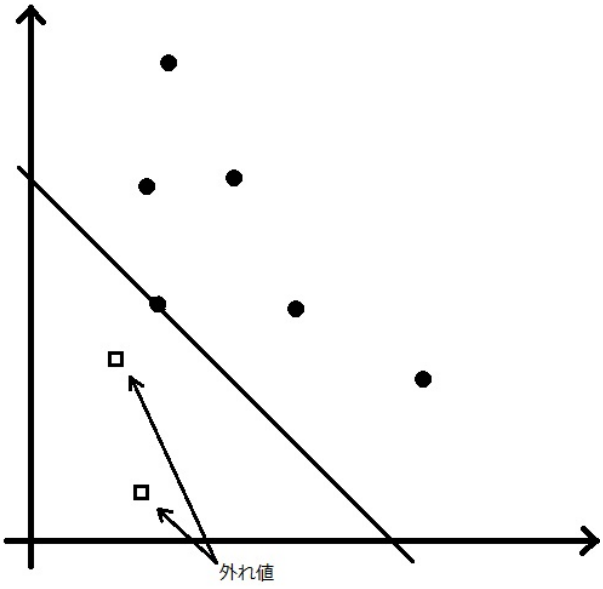

図 4.2: One Class SVM

目的関数の式は以下である。

$$\min_{w,b,\xi,\rho} \frac{1}{2} w^T w - \rho + \frac{1}{\nu N} \sum_{i=1}^{N} \xi_i$$

subject to

$$w^T \phi(x_i) \geq \rho - \xi_i$$

$$\xi_i \geq 0 \quad (i = 1, 2, \cdots, N).$$

上記を以下の双対問題に変換して超平面を求める。

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha$$

subject to

$$0 \leq \alpha_i \leq \frac{1}{\nu N}$$

$$\sum_{i=1}^{N} \alpha_i = 1.$$

第5章　実験

5.1　実験手順

一般の巨大コーパスとして新聞記事3年分 (毎日新聞'93 '95) を利用する。ここからあるキーワードが含まれる記事を取り出し、そこに人為的に外れ値文書を混ぜる。そこから外れ値文書を検出できるかどうかの実験を行った。同様の実験をLOF、One Class SVMでも行い、結果を比較する。

文書セットは3つ用意した。「災害・死者」が含まれる記事、「女優・結婚」が含まれる記事、「会社・倒産」が含まれる記事である。それぞれ文書セット1、2、3とする。外れ値文書として2種を用意した。先の3つの文書セットとは関係のなさそうな、ランダムな内容のもの5つ、同類の外れ値文書としての、「野球」に関する内容のもの5つである。それぞれ外れ値文書1、2とする。文書セット1、2、3の記事数はそれぞれ334,168,607個となっている。記事の総数は304204個、全単語数は207622個となった。

5.1.1　記事の抽出と形態素解析

まず元となった新聞記事から記事の本文を抜き出し、通し番号をつけて形態素解析を掛けた。さらに「災害・死者」の単語が含まれる記事を探し、文書セット１を作成した (図5.1)。同様にして、文書セット２、３も作成した。単語には名詞、形容詞などがあるが、この手法では名詞のみを用いる。形態素解析後の記事は図5.2のようになっている。ここの2段目が「名詞」となっている語句を利用する。

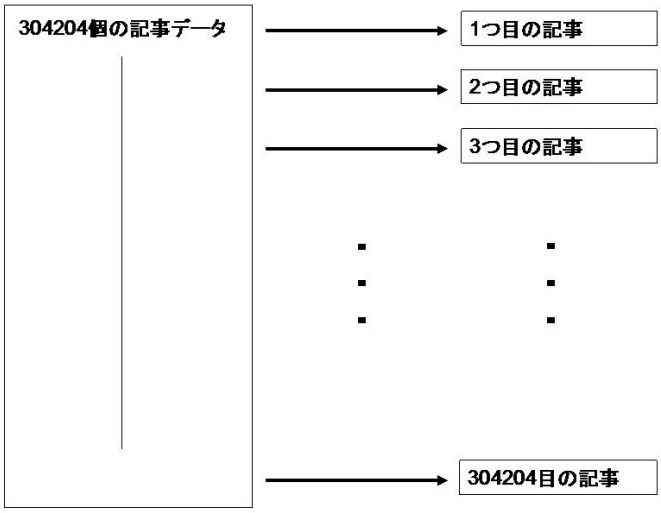

図 5.1: 記事ごとに分割

```
　記号, 空白,*,*,*,*,　,　,
一月 名詞, 副詞可能,*,*,*,*, 一月, イチガツ, イチガツ
の 助詞, 連体化,*,*,*,*, の, ノ, ノ
阪神 名詞, 固有名詞, 組織,*,*,*, 阪神, ハンシン, ハンシン
大震災 名詞, 一般,*,*,*,*, 大震災, ダイシンサイ, ダイシンサイ
による 助詞, 格助詞, 連語,*,*,*, による, ニヨル, ニヨル
死者 名詞, 一般,*,*,*,*, 死者, シシャ, シシャ
は 助詞, 係助詞,*,*,*,*, は, ハ, ワ
六 名詞, 数,*,*,*,*, 六, ロク, ロク
千 名詞, 数,*,*,*,*, 千, セン, セン
三 名詞, 数,*,*,*,*, 三, サン, サン
百 名詞, 数,*,*,*,*, 百, ヒャク, ヒャク
八 名詞, 数,*,*,*,*, 八, ハチ, ハチ
人 名詞, 接尾, 助数詞,*,*,*, 人, ニン, ニン
に 助詞, 格助詞, 一般,*,*,*, に, ニ, ニ
.....
```

図 5.2: 形態素解析後の記事本文

5.1.2 単語のカウント

ρ 値を計算するには、各単語がどの記事に出現するのか把握する必要がある。そのために、まず各文書セットごとに出現する単語をそれぞれまとめた。また各記事ごとに使用された単語の種類と、単語数を数えたファイルを作成した。図5.3はその例である。記事番号：単語数：出現した単語の順になっている。同じ単語が複数個連続しているのはその記事内での単語の出現率を求めるためである。

```
930116019:440:3 3 3 3 4 4 9 9 14 14 16 16 19 22 22 22 25 25 25 ...
930121165:527:3 3 6 7 7 9 9 9 14 14 14 16 16 16 17 19 19 22 22 ...
930122210:790:3 3 7 7 7 7 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 14 ...
930203204:61:16 628 704 808 1034 1197 1363 1363 1503 1507 1651 ...
930226217:139:19 27 40 64 66 111 133 317 317 317 317 323 323 ...
930226238:1033:1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
930304045:176:3 3 7 7 9 9 14 14 16 19 19 22 27 29 32 51 87 91 ...
930428288:210:9 14 16 184 274 323 323 323 604 604 798 1013 1797 ...
.....
.....
```

図 5.3: 各記事の出現単語を求める

5.1.3 逆トピックワードの選出

次に各単語ごとに、文書セット内でその単語が出現した記事数をカウントする。その上位10単語を表5.1に示す。横の数値はその出現記事数である。そこから ρ 値の計算を行った。本論文では、その値が高い単語上位9割を逆トピックワードとした。計算の結果、実験では文書セット1、2、3の逆トピックワード上位10個は表5.2のようになった。

表 5.1: 各文書セットでの上位単語 (記事数)

文書セット 1	文書セット 2	文書セット 3
災害:329	女優:163	倒産:602
死者:329	結婚:163	会社:602
人:315	人:129	十:529
日:303	年:124	二:506
十:282	二:118	日:501
二:270	十:118	一:486
一:253	一:116	円:462
三:242	の:115	三:443
五:241	日:113	年:426
百:234	こと:108	九:410

表 5.2: それぞれの逆トピックワード

文書セット 1	文書セット 2	文書セット 3
交渉	交渉	勝
表示	長	主催
改革	政府	葬儀
派	法	選手
戦	政治	メートル
大会	結果	女子
選手	側	発売
女子	区	戦
系	会議	大会
捜査	首相	党

5.1.4 各文書の KL ダイバージェンスを求める

最後に各記事の逆トピックワードそれぞれの出現率とその合計を求めた。それを元に KL ダイバージェンスの計算を行った。そのさい作成したファイルの例を表 5.4、表 5.5 に示す。

930116019%%%人:0.01590909 恐れ:0.0022727 八:0.013636363 四:0.00909090909 ...
930121165%%%足:0.00379506 人:0.018975332 八:0.003795066 予想:0.001897533 ...
930122210%%%足:0.00253164 人:0.025316455 恐れ:0.0012658 八:0.00379746835 ...
930203204%%%人:0.06557377 八:0.016393442 ら:0.016393442 発見:0.016393442 ...
930226217%%%何:0.00719424 人:0.007194244 指摘:0.0071942 主張:0.007194244 ...
930226238%%%一家:0.000968 何:0.001936108 人:0.028073572 心:0.00096805421 ...
930304045%%%何:0.00568181 人:0.028409090 八:0.011363636 開始:0.005681818 ...
930428288%%%人:0.01904761 八:0.009523809 四:0.009523809 年齢:0.004761904 ...
930519027%%%何:0.00857142 人:0.008571428 ダム:0.0085714 末:0.00571428571 ...
.....
.....

図 5.4: 逆トピックワードの記事ごとの出現率

222222222:-0.457979925629485
111111111:-0.461794055899092
333333333:-0.497341354344644
930720048:-0.525869013677103
444444444:-0.571326699758757
950221047:-0.623712942669633
555555555:-0.635901156891584
.....

図 5.5: 求まった KL ダイバージェンス

5.2 結果

実験結果を表 5.3 から 5.6 に示す。表で示した数値は 5 つの外れ値文書の外れ値の順位を昇順に並べたものである。括弧の数値はその平均を表す。

表 5.3: OCC での実験結果

文書セット	外れ値文書セット 1	外れ値文書セット 2
災害・死者	2,6,8,16,48 (16)	1,2,3,5,7 (4)
女優・結婚	2,3,4,13,22 (9)	1,4,5,7,44 (12)
会社・倒産	3,17,18,65,122 (45)	2,10,12,28,43 (19)

表 5.4: LOF での実験結果

文書セット	外れ値文書セット 1	外れ値文書セット 2
災害・死者	62,124,179,303,331 (200)	181,226,244,256,264 (234)
女優・結婚	27,28,35,135,161 (77)	5,25,30,49,60 (34)
会社・倒産	142,208,215,492,509 (313)	250,282,346,509,517 (381)

OCC と LOF では結果に外れ値の度合いを出力するので、外れ値文書の平均順位を比べることでそのパフォーマンスを比較できる。その結果、明らかに本手法の方が優れていることが確認できた。

One Class SVM は直接外れ値文書を出力するため、単純には本手法と比較できない。そのため、本手法の平均順位を抽出数、その平均順位より高かった数を正解数と仮定して F 値による比較を行った。F 値は次式によって計算した。p は抽出した文書のうち、正解した文書の割合 (=精度)、r は外れ値文書のうち、抽出された文書の割合 (=再現率) である。

$$F = \frac{2 * p * r}{p + r}$$

表 5.5: One Class SVM での実験結果

文書セット	外れ値文書セット 1		外れ値文書セット 2	
	抽出数	正解数	抽出数	正解数
災害・死者	25	1	23	0
女優・結婚	46	1	50	1
会社・倒産	31	0	16	0

表 5.6: OCC と One Class SVM の F 値比較

	ランダムな外れ値		同類の外れ値	
文書セット	OCC	OCSVM	OCC	OCSVM
災害・死者	0.381	0.067	0.667	—
女優・結婚	0.429	0.039	0.471	0.036
会社・倒産	0.120	—	0.25	—

比較の結果、本手法の方が優れていることが確認できた。また外れ値文書セット 2 に対しては LOF、One Class SVM ともに検出が困難になっているが、本手法では逆にパフォーマンスが良くなっていることが分かった。

第6章　考察

実験により、OCC の方が良い結果となることが分かった。特に LOF や One Class SVM では検出率が悪くなると言われていた、同類の外れ値文書に対しても良い結果となった。同類の外れ値文書に対してのパフォーマンスは、ランダムに選んだ外れ値分書のものより良い結果となっている。これは外れ値分書の種類が固まることにより野球関連の単語がまとまり、逆トピックワードとして上位に上がり易くなったためだと思われる。用意した外れ値文書よりも順位の高かった文書を見ると、一週間の出来事の簡単なまとめなど、他の話題が多く混じっている記事であった。図 6.1 にその例を示す。元から関連度の低い記事が混じっていたため、用意した非関連文書より高い順位になったと考えられる。より正確な結果を求めるにはそのような記事を取り除いておく必要がある。

改良すべき点は多くある。実験では一般の巨大コーパスとして新聞記事 3 年分を使用したが、この量が十分といえるかは検証する必要がある。また、この手法は逆トピックワードとして設定する単語数によって、結果が大きく変わる。設定する単語数があまりに少ないと、逆トピックワードが 1 つも含まれない文書が出現し、外れ値文書を取りのがす可能性が高くなる。今回は、元の手法がトピックワードを 9 割近く採用していたのでそれに習い 9 割とした。元の単語数からどの程度の割合とすべきか、あるいは別の点から採用数を判断するのか、適切な単語数を決める方法が必要である。実験では一般の巨大コーパスからみても出現数があまりにも少ない単語はあらかじめ削除して行ったが、この少ないと判断する出現数にも適切な決定法が必要となる。

【１３日】　※神戸市立中学校７３校の授業再開完了。（以下、※は阪神大震災関連）　★経営破たんの東京協和、安全の両信組が臨時総代会を開き解散。両信組が山口敏夫元労相の関連企業４社に２０億円を超える融資をしていたことが判明。１４日、山口氏が新進党幹事長代理を辞任。１５日、東京協和が田中角栄元首相の秘書だった佐藤昭子さん（６６）が代表取締役を務める企業に３０億円を融資していたことが判明。大蔵省と東京都が大口の預金者リストを匿名で衆院予算委に提示。１６日、日本長期信用銀行の東京協和への資金協力がピーク時に３９３億円に上ったことが判明。１７日、安全信組が大口預金を集めるため、一部に半年５％の裏金利を払っていたことが判明。１８日、両信金が、法律で禁じられている導入預金を行っていた疑いが強まる。東京共同銀行が日銀資金の貸し出し対象となることが判明。　★中川和雄大阪府知事後援会のヤミ献金事件で、政治資金規正法違反で起訴された事務局長（７０）に、大阪地裁が有罪判決。　★警視庁が写真家、加納典明容疑者（５２）ら４人をわいせつ図画販売容疑で逮捕。　★アフガニスタンで新軍事組織「タリバン」が７州制圧。首都カブールに迫る。　★９４年の国内パソコン販売台数、前年比３割強の急成長で、３３０万台超。　★米ロサンゼルス・ドジャースが前近鉄の野茂英雄投手（２６）の入団を発表。　★ペルーのフジモリ大統領がエクアドルへの攻撃を中止すると発表。　★米プリンストン大が、Ａ・ワイルズ同大教授によるフェルマーの最終定理の証明について、「誤りはない」とお墨付き。　【１４日】　※政府が一般呼称を「阪神・淡路大震災」と決定。　★東京都知事選に意欲を示す島根県出雲市の岩國哲人市長が辞表を提出。１５日、石原信雄・官房副長官が下旬に辞任の意向。　★「マルコポーロ」廃刊の文芸春秋で、田中健五社長が辞任、代表取締役会長に。後任に安藤満専務。「ジャーナリズムの責任を全うするには人心一新が不可欠」とのコメントを発表。

図 6.1: 外れ値文書よりも順位の高かった文書

第7章　おわりに

本論文では、ある事柄に関連した文書セットを対象とした、外れ値文書の検出手法を提案した。外れ値文書の検出を行う場合、一般的にはデータマイニング分野の外れ値検出手法を用いる。しかし、この手法はある同類の外れ値文書がある程度の数出現した場合、検出に失敗しやすくなるという問題があった。そこで Bekkerman が提案した関連文書検出の手法を応用し、外れ値文書の検出を試みた。

実験では一般の巨大コーパスとして新聞記事 3 年分を利用した。そこからあるキーワードに関連した文書セットを作成し、そこから人為的に混ぜた外れ値文書の検出を試みた。結果は LOF、 One Class SVM での実験結果と比較し、有効性を示した。

謝辞

本研究の遂行および論文の作成に伴い、多大なご助言およびご指導を賜った新納浩幸 准教授に深い感謝の意を表します。

また、本研究に助言や協力を頂きました、同研究室の林華氏、江口晃氏、西野太樹氏、全太俊氏にも深く感謝します。

参考文献

[1] B. Schölkopf and J. C. Platt and J. Shawe-Taylor and A. J. Smola and R. C. Williamson. Estimationg the support of a high-dimensional distribution. *Neural Computation*, Vol. 13, No. 7, pp. 1443-1471, 2001.

[2] Ron Bekkerman and Koby Crammer. One-class clustering in the text domain. In *Proceedings of the Conference on Empirical Methods In Natural Language Prosessing (EMNLP '08)*, pp. 41-50, 2008.

[3] Markus M. Breuning and Hans-Peter Kriegel and Raymond T. Ng and Jörg Sander. LOF: Identifying Density-Based Local Outliers. In *ACM SIGMOD 2000*, pp. 93-104, 2000

[4] 山西健司. データマイニングによる異常検知. 共立出版, 2009.

付 録A　プログラムリスト

#特定のキーワードを含む記事を取り出すプログラム

```
use File::Copy;
foreach $file(@ARGV){
    open(IN, $file);    #$file開く
    while (<IN>){    #1行読み込み
        @list= split(/[\t,]/,$_);
        $s.=" $list[0]" if($list[1] eq "名詞");
    }
    close(IN);
    $match[0]="災害"; $match[1]="死者";
    if( (($s =~ /\s$match[0]/)||($s =~ /^$match[0]/))
                                        && ($s =~ /$match[1]/) ){
        print "$fileが該当\n";
        @cpyname= split(/\./,$file);
        $cpyname[0].=".cpy";
        copy($file,$cpyname[0]);
        move($cpyname[0],"to");
        #print "$s\n";
    }
    $i++; $s=""; @list=();
    if(($i%10000)==0){ print "$iまで終わり\n";}
}
```

```
#記事内の単語を調べるプログラム

open(IN,"Anouncpy+outs.txt");
    for($i= 1;<IN>; $i++){
        chomp; $noun{$_}= $i;
    }
close(IN);
print "読み終わり\n";

#$n= 1;
open(FH,">mtrixcpy2.txt");
foreach $file(@ARGV){
    @list=();$nnumber=0;    #初期化  list に読み込み
    open(IN,$file);
    @fn= split('\.' ,$file);
    while(<IN>){
        @line=();        #初期化
        $num=""; #  〃
        @line= split(/[,\t]/);
        if($line[1] eq "名詞"){
            unless($noun{$line[0]}){print "$line[0] at $fn[0]\n";}
            $num= $noun{$line[0]};
            #print "$num\n";
            $list[$num]++; $nnumber++; #記事内の単語数も数える
        }
    }
    close(IN);
    #mtrix に書き込み
    print(FH "$fn[0]:$nnumber:");
    $p=9638;        #文書セット数
    #$p=207622;

    for($i=1; $i<=$p; $i++){
        if(defined $list[$i]){
            while($list[$i]>0){
```

```
                print(FH "$i "); $list[$i]--;
            }
        }
    }
    print(FH "\n");

    $j++;
    if( ($j%100) ==0){
        print "$j まで完了\n";
        #close(FH); $n++;
        #open(FH,">mtrix$n.txt");
    }
}
close(FH);
```

```
#各単語の出現した記事数を求めるプログラム

@count=(); #出現数をカウント 型番→カウント数

open(IN,"Anouncpy+outs.txt");
    for($i= 1;<IN>; $i++){
        chomp; $noun[$i]= $_;
    }
close(IN);

# $a[$i] １文書の出現単語リスト　内容は単語番号
# $noun[$a[$j]] 単語
# $count{$noun[$a[$j]]}++

print("start\n");
#w が出現した記事数を数える
#for($u=1;$u<=7; $u++){
    open(IN,"mtrixcpy2.txt") or die("error[$!]");
    while(<IN>){
        split(/:/);
        @a= split(/\s/,$_[2]);
        $j=$#a;
        for($i=0; $i<=$j; $i++){
            if($flg{$noun[$a[$i]]} != 1){
                $count{ $noun[$a[$i]] }++;
                $flg{$noun[$a[$i]]}=1;
            }
        }
        @a=();%flg=();
    }
    close(IN);
#print "$u end\n";
#}

open(OUT,">kanren.txt");
```

```
foreach $key (sort {$count{$b} <=> $count{$a} } (keys(%count)) ){
    print OUT "$key:$count{$key}\n";
}
close(OUT);
```

```
#ρ値を計算するプログラム

$kijisu=329+5;
open(IN2,"zentai.txt");
while(<IN2>){
    chomp; split(/:/);
    $all{ $_[0] }=$_[1]/304205 if($_[1]>10);
}
close(IN2);

open(IN1,"kanren.txt");
while(<IN1>){
    chomp; split(/:/);
    $cpy{ $_[0] }= $_[1]/$kijisu if( ($_[1]>4)&&( exists($all{$_[0]}) ));
}
close(IN1);

@key= keys(%cpy);

foreach(@key){
    $rou{$_}= $all{$_}/$cpy{$_};
}

open(OUT,">rrou.txt");
foreach $key (sort {$rou{$b} <=> $rou{$a} } (keys(%rou)) ){
    print OUT "$key:$rou{$key}\n";
}
close(OUT);

open(OUT,">ppp.txt");
foreach $key (sort {$cpy{$b} <=> $cpy{$a} } (keys(%cpy)) ){
    print OUT "$key:$cpy{$key}\n";
}
close(OUT);
```

```
open(OUT,">qqq.txt");
foreach $key (sort {$all{$b} <=> $all{$a} } (keys(%all)) ){
    print OUT "$key:$all{$key}\n";
}
close(OUT);
```

```
#トピックワードの各記事での出現率を調べるプログラム

open(IN1,"RR.txt");    #トピックワード
while(<IN1>){
    chomp; split(/:/);
    $R{$_[0]}= $_[1]; #トピックワード→ロー
}
close(IN1);
open(IN2,"Anouncpy+outs.txt");
    for($i= 1;<IN2>; $i++){
        chomp; $noun{$_}= $i;    #単語→行番号
    }
close(IN2);

@topicalwords=keys(%R);         #トピックワード
foreach(@topicalwords){
    $topicalnumber{$_}=$noun{$_};    #トピックワード→行番号
}

print "始め\n";
open(OUT,">rpdw.txt");
open(OUT2,">rApdw.txt");
open(IN3,"mtrixcpy2.txt");    #各記事について各単語の出現率を調べる
while(<IN3>){             #1 記事読み込み
    chomp; split(/:/);
    $k_num=$_[0]; $k_n_num=$_[1];
    @list=split(/\s/,$_[2]);
    print(OUT "$k_num%%%");
#print "$k_num\n";
    foreach $word(@topicalwords){
        $count=0;
        foreach(@list){
            $count++ if( $_ == $topicalnumber{$word});
        }
```

```
        $p=$count/$k_n_num; $f+= $p;
        print(OUT "$word:$p ") if($count != 0);
    }
    print(OUT "\n");
    print(OUT2 "$k_num:$f\n"); $f=0;
}
close(IN2);
close(OUT2);
close(OUT);
```

```
#各文書ごとのKLダイバージェンスを求めるプログラム

open(IN1,"RR.txt");         #トピックワード
while(<IN1>){
    chomp; split(/:/);
    $ro{$_[0]}= $_[1];    #トピックワード→ロー
}
close(IN1);
open(IN2,"ppp.txt");         #p(w)
while(<IN2>){
    chomp; split(/:/);
    $ppp{$_[0]}= $_[1]; #単語→p
}
close(IN2);
open(IN3,"qqq.txt");         #q(w)
while(<IN3>){
    chomp; split(/:/);
    $qqq{$_[0]}= $_[1]; #単語→q
}
close(IN3);
open(IN2,"rApdw.txt");
while(<IN2>){
    chomp; split(/:/);
    $Apdw{$_[0]}= $_[1];
}
close(IN2);

@R= keys(%ro);

open(IN,"rpdw.txt");
while(<IN>){             #1つの文書について
    #前準備
    chomp; split(/%%%/);
    $k_name=$_[0];
```

```
    @list= split(/\s/,$_[1]);
    foreach(@list){            #1つの文書についてトピックワードの出現
率
        split(/:/);
        $pdw{$_[0]}=$_[1];
    }
    #式
    foreach $w(@R){            #各トピックワード
        $ssss+=log($ro{$w})*($pdw{$w}/$Apdw{$k_name});
    }
    $kekka{$k_name}=$ssss; $ssss=""; $k_name=""; @list=(); %pdw=();
}
close(IN);

open(OUT,">rkld.txt");
foreach $key(sort{ $kekka{$b} <=> $kekka{$a} } keys(%kekka) ){
    print(OUT "$key:$kekka{$key}\n");
}
close(OUT);
```