トライ木及びGMMに基づく略語のフルネームのスケーラブルな推測手法

高　明敏† 肖　川†† 石川　佳治†

† 名古屋大学大学院情報学研究科　〒 464–8603 愛知県名古屋市千種区不老町
†† 大阪大学大学院情報科学研究科　〒 565-0871 大阪府吹田市山田丘 1-1
E-mail: †gao@db.is.i.nagoya-u.ac.jp, ††chuanx@ist.osaka-u.ac.jp, †ishikawa@i.nagoya-u.ac.jp

あらまし　ビックデータの時代において，文字列マッチングは自然言語処理やデータクリーニングなど様々な分野で不可欠な一歩となっている．略語は文字列の一種として，長い語の一部を省略して短くした語であり，仕事や日常生活でよく使われている．略語のフルネームの推測とは略語とフルネームの候補となる文字列をマッチングすることによって，テキスト中の略語のフルネームを推測することであり，文書検索やファイル検索などで重要な役割を果たしている．従来の方法では主にルールやテキストパターンを用いて，略語のフルネームを推測するものであったが，個人や団体で定義された略語に対応しにくいという問題があり，大規模な略語のフルネームを推測する時の実行速度も遅い．そこで本研究では略語のフルネームを推測するスケーラブルな手法を提案する．まず POS（part-of-speech）タグを利用してテキスト中の略語を抽出する．そしてトライ木でフルネームの索引を構築し，テキスト中の略語とフルネームの候補となる文字列をマッチングすることによって候補文字列ペアを生成する．また，候補文字列ペアを生成する際に，文字列を高速に検索するため，トライ木での高速な検索手法を提案する．最後に略語とフルネームペアの 12 個の特徴を設定し，その特徴で学習した混合ガウスモデル（GMM）を用いて，生成した各候補文字列ペアをランク付けすることによって，最も適切な Top-k のペアを見つけ出す．

キーワード　略語，テキスト処理，文字列マッチング

1 はじめに

ビックデータの時代を通じて，急速な科学技術の発展に伴い，文字列データが増加している．その中で，略語は文字列の一種として，長い語の一部を省略して短くした語であり，仕事や日常生活でよく使われている．英語では長い語の単語が多く，ある概念を表す長い語を複数回参照する必要がある場合は，略語を使用する方が便利である．しかし，分野によって分かりづらい略語が存在する場合もある．例えば，生物領域で cmv は cytomegalovirus（サイトメガロウイルス）の意味で，生物知識を持っていない人にとっては意味が分かりにくい略語である．その問題を避けるため，略語のフルネームの推測は重要となっている．また，同じエンティティを認識するため，略語や他の表現からエンティティのフルネームを推測するのも重要となっている．略語のフルネームの推測は，さまざまな分野で応用できる．例えば，文書検索ではキーワードの略語で検索する場合，略語とフルネームの両方が検索結果として得られる．長い語のキーワード検索がたやすくなるだけではなく，ユーザーにとっても非常に有益な機能だと考えられる．また，他の応用としては，専門用語検索，検索エンジンやファイル検索などが考えられる．

既存研究では，略語のフルネームの推測は，主にフルネームの各単語の頭文字で構成された略語，すなわち頭字語（Acronym）の意味について推測する問題として定義されている [1–3]．例えば，BI は Business Intelligence の頭字語である．その問題に対しては，頭字語の省略ルールが確定しているため，同一な略語が持つフルネームの意味をテキストの文脈や内容に基づいて判断するのが一般的である．一方で，頭字語ではなく，単一の単語の略語からのフルネームの推測の問題もある．頭字語と違って，単一の単語のフルネームから省略された略語の省略ルールを記述するのは難しい．従来の研究では主にルールやテキストパターンを用いて，略語のフルネームを推測するものであったが [4–7]，個人や団体で定義された略語に対応しにくいという問題がある．そしてもう 1 つの方法，機械学習に基づく方法 [1, 8, 9] では，テンプレートがなくても，学習モデルで略語とそのフルネームの省略ルールを学習できるため，上述の問題点が解決できると考えられる．しかし，大規模な略語のフルネームを推測するときの精度と実行速度を同時に向上させるのが難しい．

本研究では，推測の精度と実行速度を向上させるために，テキスト中の略語のフルネームを推測するスケーラブルな手法を提案する．まず略語を認識するため，POS（part-of-speech）タグ [10] を利用してテキスト中から略語候補の集合を抽出する．そしてトライ木でフルネームの索引を構築し，テキスト中の略語とフルネームの候補となる文字列をマッチングすることによって候補文字列ペアを生成する．候補文字列ペアを高速に生成するためのアルゴリズムを提案する．主な考え方はトライ木の各ノードに対して，そのノードの部分木にあるすべてのアルファベットをノードに格納することである．最後に略語とフルネームのペアの特徴を学習した混合ガウスモデルを用いて，生成した各候補文字列ペアをランク付けすることによって，Top-k

のペアを見つけ出す．

本論文の構成は以下の通りである．まず 2 章では，略語のフルネームの推測の関連研究について述べる．3 章では，本論文の問題定義について述べる．次に 4 章では，略語のフルネームの推測の具体的な提案手法の説明を行う．そして 5 章では，提案手法による性能評価実験について，その内容と結果を示す．最後に 6 章では，本論文のまとめと今後の課題について述べる．

2 関 連 研 究

既存研究で最もよく利用された手法はルールやテキストパターンに基づく方法 [4–7] である．例えば Schwartz ら [4] は，テキスト内の略語が括弧の中にあることと（例えば，“ ...detect cmv (cytomegalovirus)... ”），そのフルネームが必ず同じ文に現れることをルールとして定義した．また，略語のフルネームの候補の中から最も適切なフルネームを見つけ出すため，略語とフルネームの候補となる文字列の両方の右から左まで，略語に一致する最短のフルネームを見つけるアルゴリズムを提案した．Ao ら [6] は，同じくテキスト内の略語が括弧の中にあることと，そのフルネームが必ず括弧の左側の文字列内に現れることをルールとして定義した．そして，手作りのテンプレートを用いて，括弧内の略語探索（Inner Search），括弧左側文字列内のフルネーム抽出（Outer Extraction）と略語とフルネームペアの妥当性の判定（Validity Judgment）この 3 つの調査手法を提案した．そういった方法では，略語の認識は括弧に依存し，略語とフルネームが必ず文の中に出現していなければならないという制約がある上，テンプレート以外の新しい略語に対応しにくいという問題点がある．

ルールやテキストパターンに基づく方法以外，もう 1 つの一般的な方法は機械学習に基づく方法である．Kuo ら [8] は，機械学習に基づいて，略語とフルネームの候補文字列ペアを学習モデルで認識する手法を提案した．また，形態，数値，文脈などの面から豊富な特徴セットを活用し，略語とフルネームの特性を記述した．Yeganova ら [9] は，既存の機械学習アプローチには，手動でラベル付けされたトレーニングデータが必要だと指摘した．すなわち，このようなトレーニングデータは一般に時間と労力がかかるという問題点がある．Yeganova らはそれを解決するために，トレーニングデータ内の正例は単にテキストから抽出され，負例は人工的に生成されるようにした．Li ら [1] は，略語の範囲を頭字語（Acronym）に限定し，企業向けのための頭字語の二種類のフルネーム（企業内で定義されたフルネーム（internal meanings）と公共で定義されたフルネーム（external meaning））を推測できるようにした．また，Distantly Supervised Learning を用いて，手動でラベル付けされたトレーニングデータが不要であり，あらゆる企業に展開できるようにした．機械学習に基づく方法では，学習モデルで略語とそのフルネームの省略ルールを学習できるため，ルールやテキストパターンに基づく方法の問題点が解決できると考えられる．しかし，その学習モデルの効率と効果をもっと向上させるために，まだ研究する価値がある．

他に，略語のフルネームの推測に関する多くの研究はエンティティリンキング問題にも関連する．この問題に対して，主に知識ベース [11] や深層学習 [12, 13] に基づく方法などが提案されている．知識ベースに基づく方法 [11] では，辞書などの知識ベースを利用して，二つのエンティティが同じエンティティかどうかを判断する．しかし，知識ベースに誤ったデータが存在する可能性があり，また，個人や団体によって作成されたエンティティ（本論文では略語）は知識ベース内に存在しないため推測が難しいという問題点がある．深層学習に基づく方法 [12, 13] では，二つのエンティティが同じエンティティかどうかを判断する精度が高い一方で，モデルを学習するために大量な訓練データが必要であり，訓練時間も長い．またその他にも類義語（Synonym）[14, 15] や同じエンティティの変換ルール [16, 17] に関連する研究が多く行っている．

3 問 題 定 義

本章では，略語のフルネームのスケーラブルな推測問題について，いくつかの定義を行う．特に 3.1 節では略語の明確な定義について，3.2 節では推測の定義について，3.3 節では略語のフルネームの推測問題の定式化についてそれぞれ述べる．

3.1 略語の定義

本研究では，略語はある単一の単語で構成されたフルネームの中のいくつかの文字を省略して得られた語であると定義する．例えば，表 1 では cmv は cytomegalovirus を略して得た略語である．

表 1 略 語 の 例

Abbreviation	Fullname
bldg	building
cmv	cytomegalovirus
pb	phenobarbitone

一般に使われる略語としては，複数の単語の頭文字をとる頭字語（Acronym）もあるが，本研究では対象としない．また，本研究では，ユーザー自らが定義した略語も含めて研究対象とする．

3.2 推測の定義

略語とそのフルネームの候補となる文字列をマッチングする際に，複数のフルネーム候補がある略語に対して，候補のフルネームをランク付けし，その中の Top-k 候補文字列を最も適切なフルネームとして出力する．

マッチングの際に，以下の二つの制限を置く．

- 略語を推測する際に，略語の 1 つ目の文字が必ずフルネームの頭文字とマッチングする．
- 略語の各文字が順番にフルネームの中に見つかる．

3.3 問題の定式化

本研究の目的は，テキスト中の略語を抽出して，その略語の最も適切なフルネームを推測することである．具体的な問題の

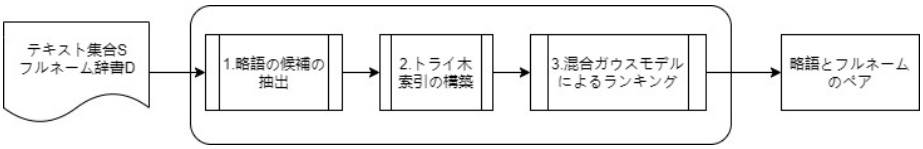


図 1　提案手法のパイプライン

Massive	hepatic	necrosis	from	CMV	was	not	seen	in	any	autopsy	liver	in	this	study
JJ	JJ	NNS	IN	NNP	VBD	RB	VBN	IN	DT	JJ	NN	IN	DT	NN

図 2　POS タグの付与例

定式化は以下となる．

テキスト集合 S とフルネームの辞書 D が与えられたとき，s_i をテキスト集合 S 中の i 番目のテキスト（$s_i \in S$），a を s_i 中の略語，f を辞書 D 中のフルネームとする．以下の条件を満たす (a, f) ペアの集合 P を見つけ出す：

$$P = \{(a, f) \mid a \text{ は } f \text{ の略語，かつ } a \in s_i, f \in D\} \quad (1)$$

この問題を解決するために，本研究ではトライ木と混合ガウスモデルを用いた略語のフルネームのスケーラブルな推測手法を提案する．主なアイデアは検索速度が速いトライ木で候補文字列ペアを生成し，混合ガウスモデルで候補文字列ペアが正しいかどうかを判断することである．具体的な提案手法は 4 章で紹介する．

4　提 案 手 法

本章では，3 章で述べた問題に対して提案したアプローチについて具体的に説明する．特に，4.1 節では提案手法の概要について，4.2 節では略語候補の抽出について，4.3 節ではトライ木の構築について，4.4 節では混合ガウスモデルによるランキングについてそれぞれ述べる．

4.1　提案手法の概要

提案手法のパイプラインの概要を図 1 に示す．まずは与えられたテキストに対して，略語の候補の抽出を行う．次に略語の候補とフルネームで構築されたトライ木索引でマッチングを行い，候補文字列ペアを生成する．最後にマッチングした候補文字列ペアを学習済みの混合ガウスモデル（GMM）でランク付けし，Top-k のペアを見つけ出す．

4.2　略語の候補の抽出

テキスト全体に対して，まずは明らかに略語ではない単語（例えば，動詞や形容詞など）をフィルタリングし，略語となりうる候補を抽出する必要がある．ここでは，POS（part-of-speech）タグを利用することによって，テキスト中の各トークンに品詞のラベルを付与する．例として，図 2 ではある例文の各トークンにラベル付けした例を示す．図 2 により，略語には固有名詞（NNP）のラベルが付けられるので，固有名詞集合のみを出力すれば，略語は固有名詞集合に含まれるため処理範囲が縮まる．しかし，略語ではない一般の名詞も固有名詞集合内に含まれる可能性があるので，ここで抽出した固有名詞集合はあくまでも略語の候補の集合である．固有名詞集合内の略語ではない名詞を取り除く方法は次の 4.3 節で説明する．

4.3　トライ木の構築

固有名詞集合内の略語の候補とフルネームの候補となる文字列をマッチングするため，フルネーム辞書をもとにトライ木の索引を構築する．トライ木では共通接頭辞の検索が可能となり，検索速度が速いというメリットがある．トライ木の構築ではアルファベットの大文字と小文字を区別せず，全部小文字で表示する．図 3 に baidu，busniess，building，chinese 四つのフルネームのトライ木を示す．

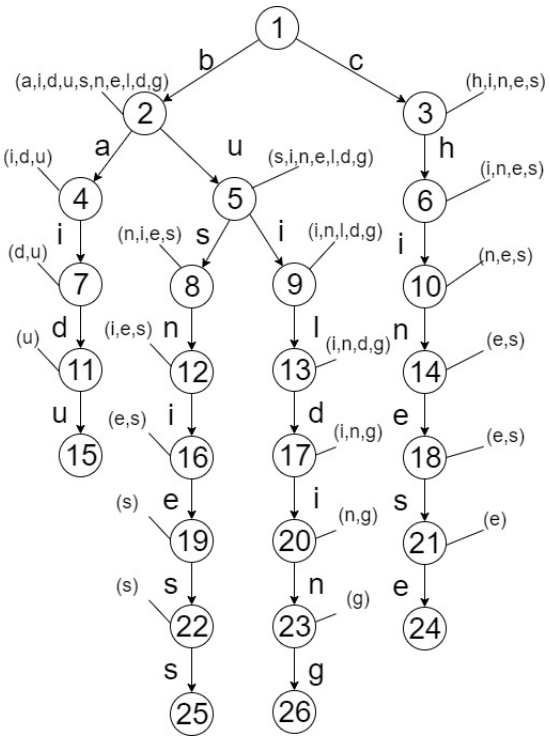


図 3　トライ木

そして 4.2 節で得られた固有名詞集合に対して，フルネームで構築したトライ木を用いてマッチングを行う．検索の際には，検索文字列，すなわち略語の候補中のすべての文字がトライ木中の 1 つの経路で見つかった場合，検索文字列と検索経路の全文字を 1 つの候補文字列ペアとして生成する．例えば，検索文字列が bldg の場合，図 3 のトライ木では bldg が building の検索経路で見つかるため，(bldg, building）を候補文字列ペアとし

て出力する．

このような処理では，略語の候補とフルネームの候補がマッチングするかを葉ノードまで探索することが頻繁に生じるため，コストが大きい．そこで本研究では高速な検索手法を提案する．主なアイデアは，各ノードに対してそのノードの部分木にあるすべてのアルファベットをそのノードに格納することである．例えば，図 3 のノード 3 には（h,i,n,e,s）が格納されている．この情報を用いれば，検索時にこの部分木以下を探索すべきかどうかを早期に判断できる．具体的な検索アルゴリズムを Algorithm 1 に示す．2 行目では，フルネームの候補集合を空集合に初期化する．3 行目では，検索文字列に対してまずはトライ木の根ノードから検索する．4 から 6 行目では，略語の最初の文字をトライ木でマッチングする．7 行目の *queue* は，検索の際に至るノードを格納するデータ構造であり，初期状態として略語の最初文字を検索できたノードを格納する．9 行目の *SearchQueue* 関数では *queue* を利用して，再帰的にフルネームの候補集合を見つけ出す．18 から 21 行目では，14 行目の *queue* からポップしたノードに対して，もし検索する文字がその子ノードで検索できる場合は，新しい *queue*1 を利用し，*SearchQueue* 関数で次の略語中の文字を検索する．検索できない場合はそのノードのすべての子ノードを *queue* に格納する．15 から 17 行目では，葉ノードまで検索できた場合，その経路での全文字をフルネームの候補集合に出力する．以上の処理を各 *queue* が空集合になるまで繰り返し，最終的にフルネームの候補集合を出力する．

また，検索文字列が検索経路の全文字とマッチングする場合は，その検索文字列が一般の名詞（すなわち一般のフルネーム）として扱い，略語の候補集合から削除する．そうすれば，略語の候補集合内の略語ではない名詞を取り除くことができる．

4.4 混合ガウスモデルによるランキング

4.3 節で得られた候補文字列ペア集合中の略語とフルネームのマッチングの度合いを評価するために，ここでは混合ガウスモデル（GMM）を利用して候補文字列ペアの略語とフルネームの省略パータンを学習する．学習する前に，まずは省略パータン p の特徴をベクトル化する必要がある．本研究では頭字語を考慮しないため，略語のフルネームは単一の単語であると考えている．また，略語の長さは 5 以下と想定する．実際の略語を観察することにより，一般に用いられている省略ルールは母音数子音数，略語の各文字の位置情報やフルネーム中の形態素に関係があるといえる．そのため，本研究では省略パターンに 12 個の特徴を付与する．表 2 に省略パータンの特徴ベクトルの内容を示す．形態素に関する特徴を求める際には，Morfessor [18] を使用する．

表 2 の特徴により，GMM のトレーニングを行う．GMM の密度関数関数の式は以下となる．

$$P(p) = \sum_{k=1}^{n} w_k N(p \mid \mu_k, \Sigma_k) \qquad (2)$$

n は混合ガウス分布の数，w_k は k 番目のガウス分布の重み，

Algorithm 1 Fast search algorithm

Input: Abbreviation: a, T : a trie tree which build on the fullname set
Output: Candidate fullname set: $F = \{f_1, f_2, ..., f_m\}$

```
1: function FastSearch(a, T)
2:     F = ∅
3:     A ← root of the T
4:     if a[0] can be found in A's children then
5:         A ← A's children
6:     end if
7:     queue.push(A)
8:     i = 1
9:     F = SearchQueue(queue, T, a, i)
10: return F
11: end function
12: function SearchQueue(queue, T, a, i)
13:     while queue ≠ 0 do
14:         c = queue.pop()
15:         if c reach the leaves node then
16:             F ← fullname of the path where c is located
17:         end if
18:         if a[i] ≦ len(a) and a[i] can be found in c's children then
19:             queue1.push(c)
20:             i += 1
21:             SearchQueue(queue1, T, a, i)
22:         else if a[i] can be found in c's children subtree then
23:             queue.push(c's children)
24:         end if
25:     end while
26: return F
27: end function
```

表 2　省略パータン p の特徴

ID	Description
p_1	略語中の母音数
p_2	略語中の子音数
p_3	フルネーム中の母音数
p_4	フルネーム中の子音数
p_5	略語中の 1 つ目の文字の位置
p_6	略語中の 2 つ目の文字の位置
p_7	略語中の 3 つ目の文字の位置
p_8	略語中の 4 つ目の文字の位置
p_9	略語中の 5 つ目の文字の位置
p_{10}	略語中の最後の文字の位置/フルネームの長さ
p_{11}	フルネーム中の形態素数
p_{12}	略語の長さ/形態素数

$N(p \mid \mu_k, \Sigma_k)$ は平均 μ_k および共分散行列 Σ_k を有するガウス分布による p の確率密度関数である．n の決め方は 5 章の 5.2 節で説明し，他のパラメータは EM アルゴリズム [19] を利用して計算できる．この後，収集した略語とそのフルネームの訓練データセットを混合ガウスモデルに入力し，学習することによって，混合ガウスモデルの n と他のパラメータの値が得られる．

学習の際には，正例と負例の両方をモデルに入れる．正例のデータは実世界から得られ，負例のは負例の生成アルゴリズム

によって作成する．ここでは，負例のフルネームの作成が困難であるため，代わりに負例略語を作り，正しいフルネームと組み合わせて，1 つの負例文字列ペアとして定義する．具体的なアルゴリズムを Algorithm 2 に示す．1 行目では，負例の略語の長さを 0，負例の略語とフルネームペアの集合を空集合に初期化する．2 行目はフルネームセット Y 内の各フルネーム y_i に対して，それぞれの負例略語を作る．具体的な作り方としては，まずは 3 行目で略語の長さ l を 2 から 5 までランダムに選択する．そして 5 から 8 行目では，フルネーム中の各文字に対して略語の長さ分に分割し，それぞれ 1 つの文字を取り，負例略語を作成する．最後に負例の略語とフルネームペアの集合を出力する．

Algorithm 2 Negative sample generation algorithm

Input: Fullname set : $Y = \{y_1, y_2, ..., y_n\}$, length of abbreviation : l
Output: (negative abbreviation,fullname) : $N = \{(x_1, y_1), ..., (x_n, y_n)\}$

1: $l = 0, N = \phi$
2: **for** y_i in Y **do**
3: $l = \text{random}(2, 5)$
4: $i = 1$
5: **while** $i\frac{len(y_i)}{l}$ <= $len(y_i)$ **do**
6: $x_i[i-1] = y_i[i\frac{len(y_i)}{l}]$
7: i += 1
8: **end while**
9: $N \leftarrow (x_i, y_i)$
10: **end for**
11: **return** N

トレーニングが終了した後，学習済みの混合ガウスモデルを用いて，トライ木で検索して得られた候補文字列ペアセットに対して，各候補文字列ペアの $P(a_i, f_i)$ 値が得られる．それによって各候補文字列ペアをランク付けし，Top-k のペアを見つける．例えば，k を 2 に設定したときに，GMM によって評価された bldg の各候補文字列ペアの $P(a_i, f_i)$ は以下の表 3 の通りだとする．このとき，確率（Prob.）の大きさにより，Ranking 列に示すようにランクを付ける．k は 2 なので，Top-k の結果は p^1 と p^2 である．

表 3　ランキングの例

ID	(a_i, f_i)	Prob.	Ranking
p^1	(bldg,building)	0.7	1
p^2	(bldg,balding)	0.6	2
p^3	(bldg,breathholding)	0.5	3
p^4	(bldg,ballistocardiogram)	0.4	4

5　実　　験

本章では，4 章で導入した提案手法を実験により評価する．実験準備として，5.1 節で実験で使用するデータセットについて，5.2 節で GMM の訓練について説明する．また，5.3 節で GMM によるランキングの効果について，5.4 節でトライ木による検索時間について，それぞれの実験内容と結果を示す．最後に，5.5 節で考察を述べる．

5.1　データセット

実験では，ALLIE [20] という実世界のデータセットを用いた．ALLIE は MEDLINE から抽出された略語とその長い語（フルネーム）のデータセットである．元のデータセット中には誤りのデータや重複したデータもあるので，利用する前に前処理をする必要がある．また，数字が含んでいる略語やフルネームは考慮しないため，前処理で削除する．処理後のデータセットには複数の単語で構成したフルネームと単一の単語で構成したフルネームの二種類があるため，データセットから単一の単語で構成したフルネームとその略語のデータだけを残す．今回使用する処理後のデータセットを表 4 に示す．その中，20000 のデータを GMM のトレーニングの正例に使い，10000 のデータを GMM のトレーニングの負例に使用する．

表 4　データセット

State	Abbreviation and Fullname Pairs' Size
処理後	63658
トレーニングの正例	20000
トレーニングの負例	10000

処理後のデータセット内の略語長さの分布は図 4 に示す．図

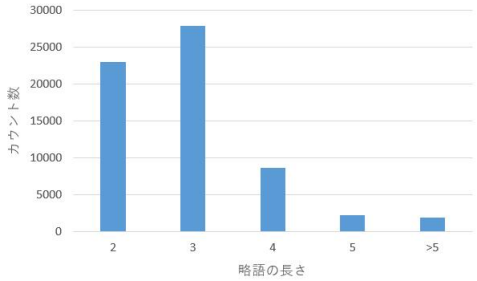


図 4　略語の長さの分布

4 に示したように，略語の長さは主に 2 から 5 までに分布しているので，本研究では長さ 2 から 5 までの略語を研究対象として扱う．

5.2　GMM のトレーニング

GMM の訓練にガウス分布の数 n 以外のパラメータは全部 EM アルゴリズムで得られるため，ここではベイズ情報量規準を利用して n を決める．ベイズ情報量規準の一般的な形式は次の通りである．

$$BIC = -2\ln(L) + k\ln(n) \tag{3}$$

L は尤度関数，n は標本の大きさあるいは観測の数，k は母数あるいは独立変数の数である．

クラスタ数を [1,30] の範囲に設定し，共分散のタイプを“ spherical ”,“ tied ”,“ diag ”,“ full ”の 4 つに設定する．それぞれの意味は以下のとおりである．

- “ spherical ”: 各構成要素には独自の分散がある．
- “ tied ”: すべての構成要素が同じ一般的な共分散行列を共有する．
- “ diag ”: 各構成要素には独自の対角共分散行列がある．

• “ full ”: 各構成要素には独自の一般共分散行列がある．

そして各クラスタ数のモデルの BIC スコアを図 5 に示す．

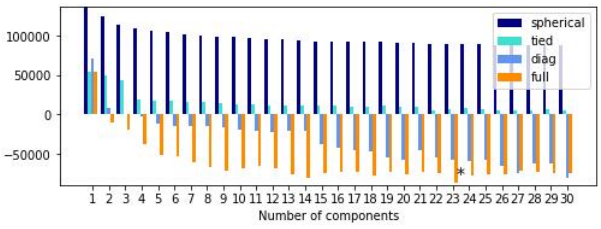


図 5　BIC score per model

BIC スコアが低いモデルが優れているので，一番いいモデルはクラスタ数が 23，共分散のタイプが“ full ”のモデルであることがわかる．

5.3　GMM によるランキングの効果

本節では混合ガウスモデルの効果を評価するために実験を行う．

5.3.1　実験内容と評価指標

ここでは，PubMed からテストテキスト集合を収集し，クラスタ数を 23 として学習した混合ガウスモデルを利用し，各候補文字列ペアにランク付けを行い，その結果を評価する．実験の評価指標は以下のとおりである．

- Mean Reciprocal Rank（MRR）
- Success Rate

MRR は上位 k 件のランキングにおける正解の文字列のランクの逆数の平均として定義される（ランキング結果に含まれない場合は 0 とする）．一般的な形式は次の通りである．

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{4}$$

$rank_i$ は i 番目のクエリの最初の正解した文字列のランク位置を示す．

Success Rate は正解文字列がランキング内に存在する確率として定義される（存在する場合は 1，しない場合は 0 としてカウントされる）．

また，混合ガウスモデルの効果と比較するため，以下の素朴な手法（Naive Method）を定義する．

Naive Method

Naive Method では，Morfessor [18] を用いて候補文字列ペアが正しいかどうかを判断する．例えば，候補文字列ペア（bldg, breathholding）を判断する場合，breathholding は breath と holding に分割するので，breathholding の略語は bh であるはず．そのため，候補文字列ペア（bldg, breathholding）は誤りの文字列ペアであると判断する．

5.3.2　実 験 結 果

混合ガウスモデルにより，代表的な略語のフルネーム候補の Top-5 の結果を表 5 に示す．“ * ”が付いているフルネームの候補はその略語の正しいフルネームであることを示す．

結果としては，1 行目と 3 行目の例では略語の正しいフルネームは Top-1 に存在し，そして 2 行目と 4 行目の例では略語の正しいフルネームは Top-2 に存在する．また，2 行目の例では略語の正しいフルネームが二つ存在し，どれが一番適切なフルネームを判断したい場合はテキストの内容を考慮する必要があり，本研究の枠組みでは対応できない．

表 5　略語のフルネーム候補の Top-5 ランキング結果

Abbreviation	Cadidate Fullname
bp	benzopyrene*, bromperidol, bitemporal, breakpoint, bromopride
cpz	centpropazine*, chlorpromazine*, cyclopenthiazide, cephazolin, chimpanzee
pfbe	perfluorobutylene*, plasmafibronectin, perfluorobutane, pentafluorobenzyl, perfluoroisobutylene
dheas	demethylsterigmatocystin, dehydroepiandrosterone*, dimethylacetals, desmethylazelastine, dihydrobenzothiazines

Top-5 の実験結果を MRR と Success Rate で評価した結果を図 6 に示す．また，GMM と Naive Method の Success Rate を比較し，結果は図 7 に示す．

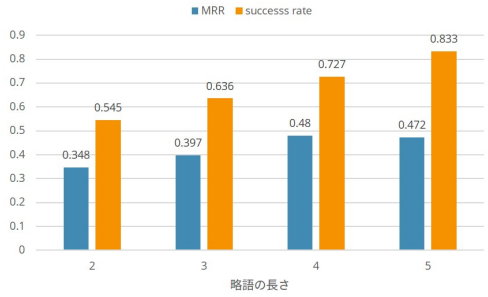


図 6　Mean Reciprocal Rank と Success Rate

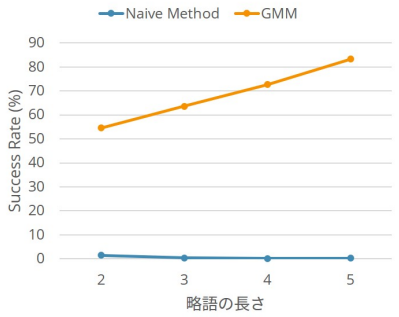
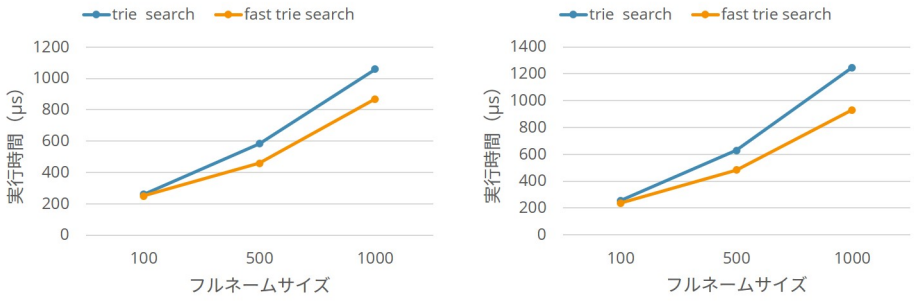


図 7　Success Rate in GMM and Naive Method

結果として，略語の長さが増加するほど，MRR と Success Rate が高くなることが分かった．そして，Naive Method と比べて GMM の Success Rate が明らかに高いということも分かった．

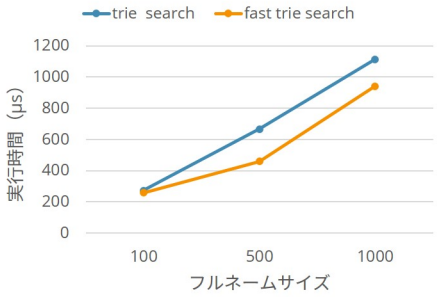
5.4　トライ木による検索時間

本節では高速な検索手法の効率を評価するために実験を行う．実験内容としては，フルネームのトライ木を構築し，通常のトライ木検索と提案手法による検索を行い，それぞれの検索時間を測る．略語の長さをそれぞれ 2, 3, 4, 5 に設定し，フルネームのデータサイズは 100 から 1000 の間で変化させる．100 から

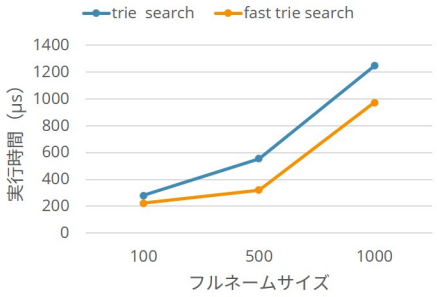


(a) 長さは 2 の略語の検索時間

(b) 長さは 3 の略語の検索時間



(c) 長さは 4 の略語の検索時間



(d) 長さは 5 の略語の検索時間

図 8　略語の検索時間

100 までのサイズのフルネームで構築した各トライ木のノード数を表 6 に示す．そして各長さに対する略語の検索時間の計測結果を図 8 に示す．

表 6　トライ木のノード数

Fullname Size	Node Size
100	1250
500	4663
1000	8420

結果として，通常のトライ木検索より提案手法の方が実行時間が少なくなった．また，フルネームのサイズの増加に伴い，両方の実行時間が線形的に増加することがわかった．

5.5　実験のまとめ

混合ガウスモデルについて

本実験により，提案手法で利用した混合ガウスモデルは定義した Naive Method により効率的に候補文字列ペアを判断できることがわかった．実験で使用した Allie は生物領域での略語とフルネームを収集したデータセットなので，生物領域の省略パータンを学習できる．同じように，他の領域，個人や団体で定義した略語とフルネームのデータがあれば，特定領域中の省略パータンを学習し，その領域内の略語のフルネームも推測できると考えられる．

高速な検索について

本研究で提案した高速な検索により，検索の際にかかる時間を短縮可能である．すなわち，検索対象となった略語の候補の集合内の各略語候補に対して，その候補文字列ペアを見つけ出す時間が短くなる．しかし，あらかじめにノードの部分木にあるすべてのアルファベットをノードに格納するのは，速度を向上した代わりにメモリに負担がかかる恐れがある．利用するトライ木の規模によっても，高速な検索を利用する重要性が変わると考えられる．

6　おわりに

本論文では，テキスト中の略語（単一のフルネームから略した略語）のフルネームの推測問題に対して，トライ木及び GMM に基づくアプローチを提案した．まず POS（part-of-speech）タグを利用してテキスト中から略語候補の集合を抽出し，フルネームで構築したトライ木でマッチングを行い，候補文字列ペアを生成した．そして候補文字列ペアを高速に生成するための高速な検索手法を提案した．また，12 個の省略パータンの特徴を挙げて，正例と負例を含んでいるトレーニングデータを利用し，GMM で略語とフルネームのペアの特徴を学習した．最後に学習済みの GMM を用いて，生成した各候補文字列ペアをランク付けすることによって，Top-k のペアを見つけることができた．実験では，提案した高速な検索手法は一般のトライ木検索より速いということと GMM が効率的であることを示した．

今後の課題としては，単一のフルネームから略した略語以外に頭字語（Acronym）も研究対象として考慮したいと考えている．その場合は，候補文字列ペアを判断する際にテキストの構文や内容などの影響要素も考慮しなければいけないと考えられる．

謝　　辞

本研究の一部は科研費 (16H01722，19K11979) による．

文　献

[1] Y. Li, B. Zhao, A. Fuxman, and F. Tao, "Guess me if you can: acronym disambiguation for enterprises," in *Proceedings of ACL*, pp. 1308–1317, 2018.

[2] N. Okazaki and S. Ananiadou, "A term recognition approach to acronym recognition.," in *Proceedings of COLING/ACL*, 2006.

[3] W. Zhang, Y. Sim, S. Jian, and C. L. Tan, "Entity linking with effective acronym expansion, instance selection and topic modeling," in *Proceedings of IJCAI*, pp. 1909–1914, 2011.

[4] A. S. Schwartz and M. A. Hearst, "A simple algorithm for identifying abbreviation definitions in biomedical text," in *Proceedings of Pac. Symp. Biocomput.*, pp. 451–462, 2003.

[5] J. Chang, H. Schütze, and R. Altman, "Creating an online dictionary of abbreviations from medline," *Journal of the American Medical Informatics Association : JAMIA*, vol. 9, pp. 612–620, 2002.

[6] H. Ao and T. Takagi, "ALICE: an algorithm to extract abbreviations from MEDLINE," in *Proceedings of J. Am. Med. Inform. Assoc.*, vol. 12, pp. 576–586, 2005.

[7] Y. Park and R. J. Byrd, "Hybrid text mining for finding abbreviations and their definitions," in *Proceedings of EMNLP*, 2001.

[8] C.-J. Kuo, M. H. Ling, K.-T. Lin, and C.-N. Hsu, "BIOADI: A machine learning approach to identifying abbreviations and definitions in biological literature," in *Proceedings of BMC Bioinformatics*, vol. 10, 2009.

[9] L. Yeganova, D. C. Comeau, and W. J. Wilbur, "Machine learning with naturally labeled data for identifying abbreviation definitions," in *Proceedings of BMC Bioinformatics*, vol. 12, 2011.

[10] S. Petrov, D. Das, and R. McDonald, "A universal part-of-speech tagset," in *Proceedings of LREC*, pp. 2089–2096, 2012.

[11] W. Shen, J. Wang, P. Luo, and M. Wang, "Linden: linking named entities with knowledge base via semantic knowledge," *Proceedings of WWW*, 2012.

[12] Y. Li, C. Wang, F. Han, J. Han, D. Roth, and X. Yan, "Mining evidences for named entity disambiguation," in *Proceedings of SIGKDD*, pp. 1070–1078, 2013.

[13] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: a design space exploration," in *Proceedings of SIGMOD*, pp. 19–34, 2018.

[14] J. Lu, C. Lin, W. Wang, C. Li, and H. Wang, "String similarity measures and joins with synonyms," in *Proceedings of SIGMOD*, pp. 373–384, 2013.

[15] W. Tao, D. Deng, and M. Stonebraker, "Approximate string joins with abbreviations," in *Proceedings of PVLDB*, vol. 11, pp. 53–65, 2017.

[16] A. Arasu, S. Chaudhuri, and R. Kaushik, "Transformation-based framework for record matching," in *Proceedings of ICDE*, pp. 40–49, 2008.

[17] A. Arasu, S. Chaudhuri, and R. Kaushik, "Learning string transformations from examples," in *Proceedings of PVLDB*, vol. 2, pp. 514–525, 2009.

[18] S. Virpioja, P. Smit, and S.-A. Grönroos, "Morfessor." Website, 2018. `http://morfessor.readthedocs.io/`.

[19] 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.

[20] Y. Yamamoto, A. Yamaguchi, H. Bono, and T. Takagi, "Allie: A database and a search service of abbreviations and long forms," *Database : the journal of biological databases and curation*, vol. 2011, p. bar013, 2011.