全部分文字列のクラスタリングとその応用

岡野原 大輔
株式会社 Preferred Infrastructure

hillbig@preferred.jp

1　はじめに

単語列を処理する時に単語情報を直接扱うのではなく，単語をクラスタリングした結果を利用することは多くの自然言語処理タスクで有用であることが知られている[1].

例えば，尤もらしい単語列に対し高い確率を与える言語モデルを構築する場合を考えてみる. 単語列に対するマルコフモデルで言語モデルを構築した場合，単語種類数とその組み合わせ数は非常に多いため, 訓練コーパスだけからパラメータ推定に十分な統計量が得られない．そこで，単語をクラスタリングし，各単語をクラスタリング結果のクラスに置き換え，クラス列に対するマルコフモデルを考えることで，パラメーター数を大きく減らすことができ，確率モデルの平滑化を自然に実現することができる．実際，現時点で最高精度の言語モデルはこのアイディアに基づいている [2].

また，多くの教師付学習では，特徴量として単語や品詞情報を利用するが，これらに加えて単語のクラスタリング結果を特徴量として利用することも考えられる．例えば Koo [4] らは，半教師付学習において，ラベル無しデータを利用して単語のクラスタリングを行い，その結果を係り受け解析タスクにおける特徴量として利用することで，精度向上が達成できることを示している．これは単語や品詞がタスクの特徴量としては必ずしも最適の粒度ではなく，クラスタリング結果を利用することで単語よりは粗いが品詞よりは細かい統計量を得ることができ，タスクの特徴を捉えることができることを示唆している．

しかし，単語をタスクの処理単位とするのは適切ではなく，さらに何を持って単語とするか自明で無い場合が少なくない．例えば，“鈴木” という単語のみでは意味は不明瞭であるが，“鈴木一郎” であれば野球選手であると判別できる．この他にも製品名などの固有表現，レビュー中の感情表現など複数単語で初めて意味を適切に捉えられる場合が多い.

そこで本稿では全ての部分文字列を候補にしてクラスタリングを行うことを考える．本稿では説明のために部分文字列の単位を文字として扱うが，この代わりに，形態素，単語を単位とすれば部分文字列はそれぞれ形態素列，単語列に対応することになる.

全ての部分文字列を対象にクラスタリングを行う場合，まず計算量が問題となる．文書中に出現する部分文字列の数は文書長の二乗に比例し，これをそのままクラスタリングするのは困難である．これを解決するために本稿では，部分文字列を特徴量が同じもの同士をまとめ上げる．さらにクラスタリングの際に乱拓化特異値分解を適用し，候補数，特徴種類数に比例した計算量でクラスタリングを行う.

実験では Wikipedia を利用し，提案手法のスケーラビリティでかつ，高効率であることを示す．例えば1万文書, 1000 万文字からなるコーパス中に出現する全て部分文字列を 1 分未満でクラスタリングを行うことができる．また，実験結果でクラスタリングに利用する特徴情報を変えることにより，どのようなクラスタリング結果が得られるのかを考察する.

2　関連研究

単語をクラスタリングするに当たっては，“同じ文脈を持つ単語は，同じ意味を持つ傾向にある” という Distributional Hypothesis を利用するのが一般的である．文脈情報は疎である場合が多く，文脈から得られた特徴情報を特異値分解を利用して潜在表現に変換し，その結果を利用してクラスタリングする方法が提案されている [9]．同手法は近年 Lamar ら [5] によって，再度実験され，従来の隠れマルコフモデルを利用した手法や，クラス分布の偏りを事前分布として考慮したノンパラメトリックベイズに基づく手法と比べても高い精度を達成することができることを示した．本稿のクラスタリングもこの手法に従う.

[1] 本稿でのクラスタリングは各要素が一つのクラスに割り当てられるハードクラスタリングであるとする.

Lin ら [6] は複数単語列からなる句を対象にクラスタリングを行う方法を提案している．彼らはクエリログから得られた句候補について，文脈を単語ベクトル空間で表現し，それを直接 Kmeans を利用してクラスタリングを行った．彼らの方法では全ての部分文字列ではなく，クエリログから得られた句を対象にクラスタリングを行っており，さらに計算量増加の問題に対して分散計算フレームワークである MapReduce を利用して大量の計算機を利用することで解決している．我々の手法は，全ての部分文字列を対象にクラスタリングを行うことが可能であり（もちろん，その一部分だけを対象にクラスタリングをすることもできる），さらに計算量の問題は計算機を増やすことでは無くアルゴリズムの改善により解決している．

部分文字列を統計情報が同じもの同士にまとめ上げて効率的に部分文字列を処理する手法は著者によって，文書分類 [8] や文書クラスタリング [7] など他のタスクでも適用できることが示されている．

3 提案手法

提案手法は与えられたコーパス文書に出現する全ての部分文字列を対象にクラスタリングを行う．提案手法の大きな流れは次の通りである．はじめに文書中の全ての部分文字列を列挙する．次に部分文字列それぞれについて，その文脈から特徴ベクトルを計算し，次に特異値分解を適用し特徴ベクトルを潜在表現ベクトルに変換する．最後に潜在表現ベクトルに対して KMeans++を利用してクラスタリングを行う．以降，各ステップについて順に説明をする

3.1 文脈による特徴ベクトル

はじめに部分文字列から，どのように特徴ベクトルを得るかについて述べる．なお，部分文字列の効率的な列挙方法，纏め上げについては 3.5 章で述べる．

今回文脈を大きく分けて adjacent, sentence, doc の三つの方法で特徴情報を抽出した．なお，全ての部分文字列は自然数による添字で対応付けられるとする．

左文脈, 右文脈:　部分文字列の直前の文字列を左文脈，直後の文字列を右文脈と呼び，それらの中で出現した単語から得られる特徴ベクトルをそれぞれ $\boldsymbol{l}, \boldsymbol{r}$ とする．例えば，l_i は，部分文字列の直前に i 番目の単語が出現した回数を表す．多くの言語において，単語の直前と直後の文脈の役割は違うため隣接する文脈を左方向と右方向と区別することでクラスタリングの質をあげることができる [9].

文共起:　対象部分文字列が出現した文番号の集合から得られるベクトル $\boldsymbol{s}$ を特徴ベクトルとする．例えば s_i は文番号 i に部分文字列が出現している時 1、そうでない場合 0 である．

文書共起:　対象部分文字列が出現した文書番号の集合から得られるベクトルを特徴ベクトルとする．

どのような文脈で特徴ベクトルを得たのかは以降のステップでは依存しないので，いずれの場合でも部分文字列 i に対応する特徴ベクトルを $\boldsymbol{x}_i \in R^m$ とし、これらを行にならべた行列を X とする．

3.2 特異値分解による潜在表現への変換

文脈から得られた特徴ベクトルをそのままクラスタリングするのは，データスパースネスの問題からうまくいかないことが知られている [9]．特に今回は出現回数の少ない部分文字列も扱うため，この問題は顕著となる．

これを解決するために，特徴ベクトルを平滑化し、潜在表現ベクトルへ変換する [9, 5]．この潜在表現においては同じような文脈に関する特徴情報が同じ次元に潰されており，先程のデータスパースネスの問題を解消できる．これは潜在意味解析と同じ考え方である．

任意の行列 $X \in R^{n \times m}$ は $X = USV^T$ のように分解できる，但し $U \in R^{n \times n}, V \in R^{n \times m}$ はそれぞれ直交行列であり，S は n 行 m 列の対角行列である．これを X の特異値分解と呼ぶ．

ここで，S の対角成分のうち値が大きい r 個以外を 0 とした行列を S_r とする．そして $X_r = US_rV^T$ を考える．このようにして得られた X_r は階数が r の行列のうち X との二乗距離が最小である行列であることが知られている．

このようにして得られた行列分解から U と S_r を利用し次の行列を考える．

$$H = US_r \tag{1}$$

H の i 行目 $\boldsymbol{h}_i \in R^r$ は元の特徴ベクトル $\boldsymbol{x}_i \in R^m$ の潜在表現である．最後に各 $\boldsymbol{h}_i$ をノルムが 1 となるように正規化する．

3.3 乱拓化特異値分解

従来の特異値分解は計算量が非常に大きい問題があった．特に自然言語処理のように次元数が行，列と

もに大きく，行列が疎である場合，従来の特異値分解手法は効率的に処理することはできなかった．

本稿では [3] で提案された乱拓化に基づく手法を応用して特異値分解を効率的に行う．我々の手法では，行と列方向の両方で乱拓化による行列圧縮を行う点が彼らの手法と異なり，高速化を達成する．

初めに平均 0, 分散 1 のガウシアンからサンプルされた n 行 r 列の行列 $O \in R^{n\times r}$ を用意する．次に $Y = X^T O$ とし，Y の各列を正規直交化した行列を Y' とする．この時 $XY'Y'^T \approx X$ が成り立つ [3][2]．Y', Y'^T はそれぞれ A の圧縮，復元に対応しており，Y を右からかけることで X は圧縮され．さらに圧縮後の行列に対し Y'^T を右から再度掛けることによって元の X に復元していると考えることができる．

この操作を同じように列方向に対しても適用する．$B = XY'$ を求め，次に先ほどと同様に平均 0, 分散 1 のガウシアンからサンプルされた行列 $P \in R^{r\times r}$ を用意し，$Z = BP$ を求める．そして，Z の各列を正規直交化した行列を Z' とし，$C = Z'^T B$ を求める．最後に C に対する特異値分解を求める．C は r 行 r 列の行列であり，元の行列と比べると非常に小さいので，この特異値分解は高速に行うことができる．

さて，$X = XY'Y'^T = BY'^T = Z'Z'^T BY'^T = Z'CY'^T = Z'USV^T Y'^T$ である．$Z'U, Y'V$ は直交行列であることから，特異値分解の一意性より，$X = (Z'U)S(Y'V)^T$ と特異値分解された．

全体の計算時間は，X の中で 0 ではない成分の数を s とした時，$\mathrm{O}(sr + r^3)$ である．

3.4 クラスタリング

最後に，得られた潜在表現 $\boldsymbol{h}_i$ に対してクラスタリングを行う．この部分については [5] と同様に K-means を利用した．初期値の設定には Kmeans++ [1] を利用した．

3.5 極大部分文字列の纏め上げ

全ての部分文字列を対象に上記のステップを行うには，部分文字列の種類数が多く計算量が大きすぎる問題がある．もし，特徴情報が位置情報にのみ依存しているのであれば，出現位置が同じ部分文字列に対する文脈からの特徴ベクトルは同一である．今回利用する特徴情報は全て文脈から求められ，これは位置情報にのみ依存している．よって，全ての部分文字列のクラスタリングを求める代わりに，纏め上げたグループ毎のクラスタリングを行うことで部分文字列のクラスタリングを情報を落とさず効率的に行うことができる．

ここで利用する部分文字列の列挙方法は基本的には [8, 7] で提案されている方法と同じであり，概要のみ説明する．

テキスト $T[1,n]$ 中の部分文字列 $Q[1,m]$ の左端の出現位置の集合を $L(T,Q)$，右端の出現位置の集合を $R(T,Q)$ とする．例えば $T =$ “$abracadabra$” の時，$L(T,a) = R(T,a) = \{1,4,6,8,11\}$，$L(T,abra) = \{1,8\}$，$R(T,abra) = \{4,11\}$ である．次に，二つの部分文字列 Q, U について，Q が U の接頭辞または U が Q の接頭辞であり，かつ $P(T,Q) = P(T,U)$ である時，$Q =_L U$ と書くことにする．二項関係 $=_L$ は，反射律，対象律，推移律を満たすため，$=_L$ は部分文字列の同値関係である．$=_L$ は左端の出現位置が同じ部分文字列を集めている．このことより，$Q =_L U$ であれば，Q と U に関する左文脈特徴ベクトルは同じである．また，同様に文共起，文書共起ベクトルも同じである．また，$Q =_R U$ であれば Q と U に関する右文脈特徴ベクトルは同じである．

T 中の同値類を $\mathrm{O}(n)$ 時間で全て列挙するアルゴリズムはいくつか知られている．本稿では [8, 7] と同様に拡張接尾辞配列を利用して $\mathrm{O}(n)$ 時間で列挙した．

4 実験

提案手法の有効性を示すために実験を行った[3]．実験データは日本語 Wikipedia を利用した．以降の実験では全てクラスタリング対象の部分文字列は出現回数が 10 回以上，長さが 3 以上のものを対象とする．

はじめに提案手法の性能を確かめるためデータサイズを変えて計算時間を確かめた．

表 1 に部分文字列クラスタリングを計算する各ステップで要した時間を示す．この実験結果からは，どのステップもデータ量に対しほぼ線形に計算量が増えていることが分かる．大部分の計算量を占める後半のステップに関しては並列化により高速化が可能であり，今後の課題である．なお，文書数 50000 の時の最大メモリ使用量は 5.0GB であった．

また、クラスタ数を変えた場合、理論上は計算量はクラスタ数に比例するはずだが，クラスタ数が 100 の場合に比べ，クラスタ数 1000 の時は時間は約 2 倍程度であった．これは KMeans++を利用したことによ

[2] さらに X の階数が r の時は $XY'Y^T = X$ が成り立つ．

[3] 乱拓化特異値分解の実装については次の場所で公開されている (http://code.google.com/p/redsvd/)

表 1: 日本語 Wikipedia のクラスタリングにかかった時間．グループ数はクラスタリング対象となった部分文字列のグループ数．文書数、総文字数、グループ数以外の単位は全て秒である．

文書数	総文字数	グループ数	データ読み込み	索引構築	特徴抽出	特異値分解	KMEANS	合計時間
1000	929701	69098	0.10	0.29	0.23	1.45	0.8	2.9
5000	4587082	126338	0.50	2.28	1.77	8.58	5.23	18.2
10000	8954434	252882	4.04	4.95	4.4	17.7	12.7	43.8
50000	41486596	1355902	22.3	25.8	27.1	134.5	74.7	285.3

り，初期値の段階でデータがクラスター毎にきれいに分類されており，収束が速かったためではないかと推察される．

今回行った結果のクラスタリングの精度を定量的に測ることは難しい．そのため以降では文書数 10000, クラスター数が 1000 の時のクラスタリング結果の定性的な考察を行う．各クラスタの大きさのばらつきは，最小が 3，最大が 4149，その他の殆んどは平均である 300 付近の値であった．

まず，特徴量として左文脈と右文脈を利用した場合，前後の単語を元にクラスタリングしているため文法的な役割が同じものがまとまりやすかった．結果例は次の通り[4]．年号 [1982 年,1983 年,...]，地名 [大分, 青森, 和歌山,...]，書名 [月刊アフタヌーン, 月刊コロコロコミック, 月刊少年キャプテン,...], 動詞 [をまとめた, を加えた, をかけた, をあわせた,...]，名詞 [の名前に, の表記に, の実現に], 名前 [アンドリュー, アンドレアス, アンドレア, アンドレイ,...]

それに対し，文共起，文書共起の場合は共起している単語が同じものが集められるため，意味が似たものがまとまりやすかった．また，名詞のみならず，その分野に特徴的な表現方法がクラスタリングされる傾向があった．結果例は次の通り．近代ヨーロッパ [ナポレオン, プロイセン軍, ヴェルサイユ条約, 百年戦争], 僧侶関係 [日蓮宗, 浄土宗, 浄土真宗, 臨済宗...], 学問系の動詞 [を研究する分野, の一分野である, 自然科学の...], 江戸時代 [徳川家康,3 代将軍, 徳川将軍家, 桶狭間, 豊臣秀吉,...]

5 まとめ

本稿では，単語ではなく全ての部分文字列を対象にクラスタリングすることを提案し，それに対する効率的なアルゴリズムを提案した．提案アルゴリズムでは，部分文字列を出現位置が同じもの同士でまとめあげ，さらに乱拓化特異値分解を適用することで，効率的なクラスタリングを実現した．

また部分文字列を特徴付ける文脈の特徴情報として直前/直後の単語，文共起，文書共起の三種類の方法を利用した．直前，直後の文脈の場合は文法的な特徴量が抽出されるために，文法的に同じ役割である部分文字列がまとめられるのに対し，文共起、文書共起の場合は，意味が同じ部分文字列がまとめられる傾向がみられた．

今後，提案アルゴリズムを言語モデルや，教師付学習の特徴量として適用する他に，単語区切りとクラスタリング，教師無形態素解析などに応用していくことを考えている．

[4]各例の前のラベル名は結果に基づいて著者が付けた．

参考文献

[1] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *Proc. of SODA*, 2007.

[2] S. F. Chen, L. Mangu, B. Ramabhadran, R. Sarikaya, and A. Sethy. Scaleing shinkage-based language models. Technical report, IBM Research Division Thomas J. Watson Research Center, 2010.

[3] N. Halko, P. G. Martinsson, and J. Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. *arXiv*, 2009.

[4] T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proc. of ACL*, 2008.

[5] M. lamar, Y. Maron, M. Johnson, and E. Bienenstock. SVD and clustering for unsupervised POS tagging. In *Proc. of ACL*, pp. 215–219, 2010.

[6] D. Lin and X. Wu. Phrase clustering for discriminative learning. In *Proc. of ACL and IJCNLP*, pp. 1030–1038, 2009.

[7] D. Okanohara. *Large Scale Machine Learning for Practical Natural Language Procesing.* PhD thesis, University of Tokyo, 2010.

[8] D. Okanohara and J. Tsujii. Text categorization with all substring features. In *Proc. of SDM*, 2009.

[9] Hinrich Schütze. Distributional part-of-speech taggin. In *Proc. of EACL*, pp. 141–148, 1995.