社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ＦＩ－ 76　(6)
2004－ＮＬ－163　(6)
2004／9／16

ClassModel を用いた単語分類の拡張及び高速化

岡野原　大輔

東京大学理学部情報科学科
hillbig@is.s.u-tokyo.ac.jp

本稿では、従来の Class Model[11] による単語分類を精度を落とすことなく約 20 倍から 30 倍高速化すると同時に、言語モデルとして最適な分類数を同時決定する手法を提案する。これにより、単語総数が 1 億、を超えるような大規模コーパスを利用した単語分類が数時間で可能となる。Class Model は、学習データの対数尤度の最適化の観点から単語を Class に分類する。[11] では Exchange Algorithm を用いることにより計算量の削減を行っていたが、本稿ではこれに加えサンプリングチェック、TopDown クラスタリングを組み合わせることにより、精度を落とすことなく高速化可能であることを実データを用いた実験と共に示す。また、Class Model は分類数に自由度があるが、これを MDL[16]、AIC[1] を用いた判断を行うことにより、テストデータに対する対数尤度最小化の面から最適な分類数が自動的に求められることを示す。最後に、上記の高速化と最適な分類数決定を組み合わせたアルゴリズムを示し、最適な分類と分類数が同時かつ高速に求められることを示す。

Improved Class Model Construction with Class Number Decision.

Daisuke Okanohara

Department of Information Science, University of Tokyo
hillbig@is.s.u-tokyo.ac.jp

This paper presents a new word-clustering algorithm based on Class Model[11]. It is showed with experimental results that Exchange Algorithm with sampling check and TopDown Clustering is about 20 or 30 times as fast as original one without loss of performance. Class Model cannot decide optimal number of Class, but MDL[16] and AIC[1] criterion can be used for deciding it and shows the effectiveness with experience. Finally a new fast algorithm which decides number of class and clustering at the same time is showed.

1　はじめに

Class Model[11] は、学習用データから得られた情報を基に単語を自動的に分類し、その分類結果を用いた言語モデルである。単語を分類した結果を Class と呼ぶ。Class Model では、それぞれの単語に最適な Class を、学習データにおける Class Model の対数尤度を最大化する Class として決定する。Class Model 自体は言語モデルであるが、その分類結果を他の用途に利用することも可能である。単語を分類し、言語モデルとして利用する利点は、N-gram では十分に学習できない、学習データが少ない場合でも Class Model は推定すべきパラメータ数が少ないため有効な言語モデルとして用いることができる点が挙げられる。また、N-gram との補間に Class Model を用いることで、N-gram のみによる言語モデルと比べ大きな精度向上を行うことができる [11]。他の応用では文書クラスタリングで用いることが挙げられる。文書中の単語出現状況を各文書の feature として用いる場合、単語出現状況をそのまま feature として用いた場合に SVM や NB など計算量が大きい学習機の使用が困難になるのを、単語分類した結果の Class の出現状況を各文書の feature として用いることにより、精度を保ちつつ計算量削減を可能とする [10] ことが挙げられる。また、単語間の関係の情報は Class 間の関係から十分に得られるという考察から Class を用いてデータ圧縮を行う例や、固有情報抽出に用いる例が挙げられる [17]。　[11] では、単語分類に Exchange Algorithm を用いることにより単語分類を単語数を W、単語 bi-gram 数を B、Class 数

を C としたとき、最悪計算量 $O(B+WC^2)$ で行うが単語数や Class 数が非常に大きい場合にはこの計算量でも不十分である。また、言語モデルとして最適な Class 数の決定は人手で決めなければいけない欠点がある。本稿の成果は Exchange Algorithm にサンプリングチェックと TopDown クラスタリングを組み合わせることにより、精度を保ったまま分類速度を約 20 倍から 30 倍高速化し、また最適な Class 数決定が MDL、AIC 原理を用いることにより自動決定可能であること、そして最適な分類数と分類を同時決定するアルゴリズムを示したことである。　以下、単語分類の詳細と Class Model について 2 章で述べる。そしてサンプリングチェックと TopDown クラスタリングを組み合わせることによる高速化を 3 章で述べ、最適な分類数の決定を 4 章で述べる。高速な分類、及び分類数を同時決定するアルゴリズムを 5 章で述べる。6,7 章で BNC World Corpus を用いたそれぞれの提案手法の評価と、考察、そして今後の課題について述べる。

2　単語分類について

従来、単語分類は人手により品詞分類 (POS: Part of Speech) などが行われてきたが、そうした分類より細かい分類や用途に合わせた分類の要求が高まってきている。また、人手の場合はコストがかかる上、客観的な評価が難しいこと、新規語や専門用語の漏れ、分類の間違いなどの欠点がある。こうした状況をふまえ、コーパスから得られた情報を用いて単語を自動分類する需要が高まっている。
単語分類は、一つの単語が複数の品詞に属する場合のように、一つの単語が複数のクラスに属する場合を考慮し、単語が各クラスに確率的に分類されるソフトクラスタリングと、一つのクラスのみに分類されるハードクラスタリングの二つに大別される。
分類方法の面から単語分類を見ると、全ての単語を違った Class に分類した後、似ている単語動詞をまとめていき、最後に一つの分類にまとめることにより、階層的な分類を行う BottomUp クラスタリングと、その逆で全ての単語を同一のクラスに分類しておき、それらのうち、似ていない単語同士二つ分けていく TopDown クラスタリング、また、最初から分類数を決定しておいてから、単語分類を行う方法がある。本稿で用いる Exchange Algorithm は最後の方法を用いている。　判断基準の面から単語分類を見ると、大きく三種類に分けられる [12]。一つ目は、確率的モデルを用いずにヒューリスティックな判断基準によりクラスタリングを行う手法である [5]。この場合、判断基準が対数尤度など、統計的な基準に拠ってないため言語モデルにそのまま組み入れることが難しい。二つ目は分類するクラスタ数を初めに決めておき、確率的モデルによる対数尤度による判断基準によって分類する手法である [4, 15, 11]。三つ目はクラスタ数も統計的な判断基準から同時に求めるという手法である [14, 2, 12, 8]。これは二つ目の Class 数の決定が行えない方法と比べ、分類能力という観点から優れているといえるが、二つ目と比べ計算量が大きくなってしまうことが欠点として挙げられる。本稿が単語分類の基として利用する Class Model はハードクラスタリングであり、この分類では二番目、つまり対数尤度による判断基準によって分類するが、最適な Class 数は自動決定できない。本稿では、Class Model を計算量の増大を抑えつつ最適な Class 数を自動決定できるよう拡張した。

2.1　Class Model

Class Model[11] は、単語 w_n の属するクラスを c_n とするとき、各単語の生成確率を Class N-gram モデルと、単語の Class 中の Membership 確率の積で与えるモデルであり次のように表される。

$$P(w_n|w_{n-1}) = P(w_n|c_n)P(c_n|c_{n-N+1}^{n-1}) \qquad (1)$$

各単語の最適な Class 決定の評価基準には、このモデルによる学習データの対数尤度を用いる。Class N-gram が Bi-gram であるとき、学習データ w_1^T の対数尤度 $L_{bicm}(\pi)$ は以下のように計算できる。

$$L_{bicm}(\pi) = \Sigma_{c_1,c_2} C(c_1,c_2) log \frac{C(c_1,c_2)}{C(c_1)C(c_2)} + \Sigma_w C(w) log C(w) \qquad (2)$$

このとき、右辺第一項のみがクラスに依存するので、この変化を調べることにより、最適な単語のクラス分類を求めることができる。また、同様に Tri-gram Class Model による、学習データ w_1^T の対数尤度 $L_{tricm}(\pi)$ は以下のように計算できる。

$$L_{tricm}(\pi) = \Sigma_{c_1,c_2,c_3} C(c_1,c_2,c_3) log \frac{C(c_1,c_2,c_3)}{C(c_1,c_2)C(c_3)} + \Sigma_w C(w) log C(w) \qquad (3)$$

実際に全ての単語分類と、その対数尤度を求めるのは現実的な計算量では不可能であるため、準最適な方法を用いる必要がある。Exchange Algorithm[11] は、分類数をあらかじめ決めた後、最初に全ての単語を適当な Class に分類する。次に、それぞれの単語を他の Class に移動した時の対数尤度変化を調べ、増加している場合に移動させる。これを増加しなくなる、もしくは決められた繰り返し回数だけ繰り返し、単語分類を求める。この

方法は局所最適解は保障し、大域最適解を保障しない。具体的には、次のようなアルゴリズムを用いて単語分類を行う。

Exchange Algorithm

1. 全ての単語を適当に各 Class に割り当てる
2. 全ての単語 w について以下の操作を対数尤度の変化が収束するまで、または前もって決められた回数だけ繰り返す
   (a) 単語 w を現在のクラスから他のクラス c に移動したときの対数尤度の変化 $\Delta(w, c)$ を調べる。(trymove)
   (b) 最も対数尤度が増大したクラスに単語 w を移動する。(maxmove)

Exchange Algorithm は一般的な分類手法の一つである K-means 法と結びつけて考えることもできる。各クラスの、周辺の単語出現状況は、クラスに属している単語集合それぞれの周辺の単語出現状況の重み付き平均であり、trymove の部分で単語がどの Class に属するか（近いか）をチェックして、maxmove の部分で Class の中心を再計算していることに相当する。[11] によると、Exchange Algorithm により分類された場合の Class Model の精度は BottomUp クラスタリングを用いて分類した場合の精度と同程度であると結論づけている。

Exchange Algorithm の計算量を解析する [11]。今後の議論をわかりやすくするため、2(a) の操作を trymove、2(b) の操作を maxmove と呼ぶことにする。単語数を W、クラス数を C、trymove、maxmove のそれぞれ一回の操作の計算量を T,M としたとき、trymove は全ての単語に対し、全てのクラスについて行うので全体で WC 回行われ、maxmove は全ての単語について高々 1 回行うので全体で高々 W 回行われる。よって全体の計算量は $O(WCT + WM)$ である。T、M の計算量について詳細に調べる。

trymove は、単語からクラス、クラスから単語への bi-gram を持っておくことにより、$O(C)$ で行える [11]。さらに、この bi-gram が 0 でないもののみ再計算を行えばよいので、単語からクラスへの bi-gram の個数を wc_{bi} としたとき、$T = O(\frac{wc_{bi}}{W})$ に抑えることができる。

maxmove は、対数尤度の再計算を O(C)、trymove に用いる補助情報の再計算を O(W) で行うことができる。また、単語、クラスの bi-gram の再計算の際、単語から単語への bi-gram が 0 でないもののみを用いればよいので、単語から単語への bi-gram の個数を ww_{bi} としたとき、$M = O(C + \frac{ww_{bi}}{W})$ であり、一般に $C << W$ なので、$M = O(\frac{ww_{bi}}{W})$ である。

よって、全体の計算量は $O(WCT+WM)$ の T,M に上記の計算量を代入することにより $O(WC(\frac{wc_{bi}}{W}) + W\frac{ww_{bi}}{W}))$ と得られ、最悪計算量（bi-gram が全て 0 でない場合）は $O(WC^2 + W^2)$ である。

Class Model は単独でも言語モデルとして用いることができるが、N-gram の補間として用いることで言語モデルの精度向上をはかることができる。例えば、Tri-gram(P_{tri}) と Bi-gram Class Model(P_{bcm}) を補間した場、次の式となる。

$$P(w_n|w_{n-N+1}^{n-1}) = \lambda P_{tri}(w_n|w_{n-1}^{n-2}) + (1-\lambda)P_{bcm}(w_n|w_{n_1})$$

この λ は Class Model 学習用データとは別のデータを用いて EM アルゴリズムにより推定される。

2.2 Class Model のスムージング

Class Model は、N-gram のスムージングを自然と行っているので、N-gram より頑健であるといえるが、それでも学習データ中に出現回数が少ない場合があるので Smoothing を行う必要がある。今回用いる Class Model は未知語 (以下 $w_{unknown}$) を全て同一の単語として扱うため、未知語（正確には未知語の集合からなる）は複数のクラスに属すると考えられる。そのため未知語をいずれかのクラスに分類するのは不適切であるので、新しいクラス ($c_{unknown}$) を一つ加え、未知語はそのクラスに属すると考える。スムージングは Membership 確率 $P(w_n|c_n)$ と Class bi-gram 確率 $P(c_n|c_{n-1})$ それぞれに対し独立に行うが、Membership 確率は、未知語以外の単語は全て 0 ではなく、また未知語の場合は、それ自身のみからなる Class に属しているので、その Membership 確率は、$P(w_{unknown}|c_{unknown}) = 1$ であり、Membership に対しては Smoothing を行わない。Class N-gram に対するスムージングは次の式で表される Absolute Discounting Smoothing[13] を用いた。

$$P_{abs}(w_n|w_{n-N+1}^{n-1}) \tag{4}$$

$$= \frac{C(w_{n-N+1}^{n}) - D}{C(w_{n-N+1}^{n-1})}(C(w_{n-N+1}^{n}) > 0) \tag{5}$$

$$= \frac{N_{1+}(w_{n-N+1}^{n})}{C(w_{n-N+1}^{n-1})}(C(w_{n-N+1}^{n}) = 0) \tag{6}$$

ただし、$N_{1+}(w_{n-N+1}^{n}) = |w_i : C(w_{i-N+1}^{i-1}w_i) > 0|$
今回は $D = 0.5$ を用いた。

2.3 Class Model 高速化の関連研究

[7] は、単語 N-gram において履歴が全て Class であり、単語を直接予測するモデル (One-Sided Model) となっている。この場合の計算量は単語数と Class 数に比例する程度で抑えられ、精度も

Class Model と同程度だと結論付けられている。本稿はオリジナルの Class Model をベースにしたが、One-Sided Model でも本稿で用いた高速化手法は同様に用いることができると考えられる他、MDL 原理、AIC 原理の適用も可能だと考えられる。

3 Class Model の高速化

Class Model による単語分類に対しサンプリングチェックを用いる、TopDown クラスタリングを組み合わせる、の二つの方法で高速化を図る。サンプリングチェックは trymove に対する高速化であり、TopDown クラスタリングは収束を速める高速化である。それぞれについて順に説明する。

3.1 サンプリングを用いた高速化

Class Model は trymove 操作により、単語 w を現在分類さている Class c_{pre} から、他の Class c_{new} へ移動させた時の全体の対数尤度変化 $\Delta(w, c_{new})$ を求めている。この時の対数尤度変化 $\Delta(w, c_{new})$ は、全て再計算を行い求めるのではなく、実際に影響を受けた分のみ再計算を行う。単語移動の影響を受け、再計算が必要な Class BigramT は $T = \{(c1, c2)|(c1 = c_{new}\ or\ cpre, c_2) \cup (c_2 = c_{new}\ or\ c_{pre})\}$ と表される。$|T| = 4(C-1)$ である。 この時、全ての T に属する要素の再計算を行なって $\Delta(w, c_{new})$ を求めずに、T から一部分のみを取り出し、その部分の変化を調べることで、対数尤度変化の近似値 $\Delta_{app}(w, c_{new})$ を求めることを考える。trymove の前に $\Delta_{app}(w, c_{new})$ を調べることで、明らかに対数尤度の向上に貢献しなさそうなものに対しては trymove 操作を行わないことにより、trymove を実際に行う個数を減らすことが期待できる。これがサンプリングを用いた高速化手法の概要である。具体的な方法について以下に述べる。 Class 数が C の時、n($n < C$) 個ランダムに Class$A = \{a_1, a_2, a_3, .., a_n\}$ を選び、A が含まれる場合のみ Class-Bigram の再計算を行い $\Delta_{app}(w, c_{new})$ を求める。この A によって決定される、再計算を行う Class Bigram T′ は $T' = \{(c1, c2)|(c1 = c_{new}\ or\ cpre, c_2 \in A) \cup (c_1 \in A, c_2 = c_{new}\ or\ c_{pre}, c_2)\}$ と表される。$|T'| = 4n$ である。 さらに A を求める際、全ての要素からランダムにクラスを選ぶのではなく、Class bigram の再計算に用いる単語と Class の bi-gramC(w,c)、C(c,w) が 0 でない Class c の中からのみ、ランダムに選ぶことにより、Δ により近い正確な推定が行える。このときランダムに選ばれたクラス集合 $B = \{b_1, b_2, b_3, ..., b_n\}$、$B' = \{b'_1, b'_2, b'_3, ..., b'_n\}$ によって、再計算される Class Bigram T″ は $T'' = \{(c1, c2)|(c1 = c_{new}\ or\ cpre, c_2 \in B) \cup (c_1 \in B', c_2 = c_{new}\ or\ c_{pre}, c_2)\}$ と表される。$|T''| = 4n$ である。T′、T″ を用いて対数尤度変化を調べる操作をそれぞれ、trymoveS、trymoveSC とする。

ここで注意しなければならないのが Exchange Algorithm の trymove をそのまま trymoveS(C) に置き換えると、極大値にさえ収束しなくなってしまうことである。そのため、trymoveS(C) は trymove を実際に行うかどうかの必要条件として用いる。trymoveS(C) を導入した Exchange Algorithm は次の通りである。

Exchange Algorithm with trymoveS(C)

1. 全ての単語を適当に各 Class に割り当てる
2. 全ての単語 w について以下の操作を対数尤度の変化が収束するまで、または前もって決められた回数だけ繰り返す
   (a) 単語 w を現在のクラスから他のクラス c に移動したときの対数尤度の変化近似値 $\Delta_{app}(w, c)$ を trymoveS(C) を用いて調べる。
   (b) もし、$\Delta_{app}(w, c) > D$ ならば trymove を用いて $\Delta(w, c)$ を調べる。
   (c) 最も対数尤度が増大したクラスに単語 w を移動する。(maxmove)

3.2 TopDown クラスタリングを組み合わせた高速化

Exchange Algorithm は、初期収束でほとんどの時間を消費する。そこで初期分類状態を工夫することで最も時間がかかる初期収束を速めることが可能である。[11] では各単語をその POS に従って初期分類させておくことにより収束を速めることが可能であることが報告されている。本稿では、より少ない Class 数での分類結果を、初期分類に用いることで初期収束を速める。この手法は TopDown クラスタリングが、全ての単語を同一の Class に分類した後に、再帰的に各 Class を分割していき、各 Class への分類を求める方法と似ている。違う点は、それぞれの分類数において、Exchange Algorithm が適用されるので、全ての単語は、他の全ての Class へ移動する可能性がある点である。少ない Class 数での分類結果を初期分類に用いる時には分類結果をランダムに分割し、Class 数を増やし、求める Class 数に合わせる。Class 数 n の時の TopDown クラスタリングを組み合わせた Exchange Algorithm は次の通りである。

Exchange Algorithm with TopDown

1. $k := 2$
2. 全ての単語を Class、c_1, c_2 に適当に割り当てる。

3. Exchange Algorithm を Class 数が k として行う
4. もし $k = n$ なら終了。
5. $2k <= n$　ならば各 Class を二つに分割し、k := k * 2 とする。
   $2k > n$　ならば各 Class を一つ、もしくは二つに分割し、分割後の Class 数を n とし、k := n とする。
6. 3 へ戻る

ここで、Exchange Algorithm の代わりに Exchange Algorithm with trymoveS(C) を用いることも可能である。trymoveS(C) は収束後半では、ほとんどの trymove 候補を棄却するので収束をさらに加速することが可能である。

4　最適な分類数の決定

Class Model にとって最適な分類数とは、Class Model に用いた学習用データとは違うテストデータの対数尤度を最大化するような分類数といえる。Class Model は分類数を上げれば上げるほど学習用データの対数尤度は増加するため、対数尤度をそのまま最適な分類数決定の指標として用いることはできない。クラス数はモデル化に用いるモデルの複雑さとして考えることもでき、モデルが複雑になりすぎて過学習になることを抑制しつつ、データを表すのに最も最適な複雑度のモデルを選択する問題として考えることもできる。本稿では MDL 原理、AIC 原理を用いて、最適なモデルの複雑度の決定、つまり分類数の決定を行う。

4.1　MDL 原理

MDL(Minimum Descripsion Length) 原理 [16] はモデルを M、データを D としたとき、モデルによって符号化されたデータの符号長:$L(D|M)$ と、そのモデル自身を表現するのに必要な符号長:$L(M)$ の総和:$L(D|M) + L(M)$ が最小になるときが最適なモデルだと定義される。$L(M)$ がモデルの複雑度によるペナルティだと考えることができる。データ長 N が十分大きければ、モデルに用いたパラメーター数を d とした時、$L(M)$ は $\frac{d}{2}logN$ と近似できる [16]。よって次の値が最小のものを最適なモデルとして定義する。

$$-(\text{データの対数尤度}) + \frac{d}{2}logN \qquad (7)$$

4.2　AIC 原理

AIC(Akaike's Information Criteria) 原理 [1] は、統計的な判断基準を元に次の値が最小のモデルを最適なモデルとする。ただし、d はモデルに用いたパラメーター数である。

$$-2(\text{データの対数尤度}) + 2d \qquad (8)$$

4.3　Class Model への MDL、AIC 原理の導入

単語種類数が W、Class 数が C の時、Class Model のパラメーター数 d は、次のように求められる [8]。Bi-gram Class Model では、Class 間 Bi-gram 確率で $C(C-1)$ 個、Membership 確率で各 Class c_i に所属する単語数を n_i としたとき、$\Sigma_i(n_i - 1) = W - C$ 個であるので、$d = C(C-1) + W - C = C(C-2) + W$ である。同様に Tri-gram Class Model では、Class 間 Tri-gram で $C^2(C-1)$ 個、Membership 確率で $W - C$ 個、より全体で $d = C(C^2 - C - 1) + W$ として求められる。

5　最適な分類数、分類を同時決定するアルゴリズム

最適なクラス数の決定に用いる評価基準を前章で述べたが、これを基にどのようにしてクラスタリングを行うかについてこの章で述べる。各 Class 数において Exchange Algorithm を適用して、それぞれでの MDL、AIC を求め、最小の MDL、AIC を与えるクラス数を求める方法は計算量的に問題がある。ここで、MDL や AIC を用いた評価値は Class 数を軸にとった場合、その評価値は下に凸であり極小値は一つであることを用いる。これは、Class 数が増加するに従って対数尤度は単調減少し収束するのに対し、パラメータ数 d は Class bi-gram が $O(C^2)$、Class tri-gram が $O(C^3)$ で増加することから示せる。これを用いて、最小の MDL、AIC を与える Class 数は次の Doubling Binary Search を用いて決定することができる。Doubling Binary Search は MDL（AIC) が、減少している間に探索範囲を 2 倍にしていき、増加したら探索範囲を半分にしていく。Doubling Binary Search のアルゴリズムは次の通りである。P_i は Class 数が i の時の MDL（AIC) の値である。

Doubling Binary Search

1. P_2 を求める。$i := 2$
2. P_i の結果を用いてクラス数 2i の分類の初期化を行い、P_{2i} を求める。
3. もし、$P_i < P_{2i}$ ならば 5 へ進む
4. そうでないなら、$i := i * 2$、2 へ戻る。
5. k := $\frac{i}{2}$ もし $k < 1$ ならば 8 へ進む。
6. $P_{i-k}P_{i+k}$　を求め、P_{i-k}、P_i、P_{i+k} のうち最小のものを選び、そのクラス数を j としたとき、$i := j$ とおく。
7. $k :=$ $\frac{i}{2}$、5 へ戻る。

8. このときの i が最適な Class 数であり、分類も与えられている。

最適な Class 数を C_{opt} とする時、$O(log(C_{opt}))$ 個の P_i を求めることにより、最適な Class 数を求めることができる。また P_i を求める全ての部分で TopDown Exchange Clustering を組み合わせることができるので、高速に分類を求めることが可能である。

6 実験と考察

本章では、上記で述べた手法についての実験結果を述べる。実験は British National Corpus(BNC) を学習用データ、テストデータとして用いて、各種高速化による、CPU 時間とそれのモデル精度への影響を PP として報告する。また、Tri-gram の補間に Class Model を用いた場合の PP についても述べる。

6.1 実験設定

実験では BNC を N-gram 構築用、Class Model 構築用、N-gram Class Model の係数用、及びテストデータに分けた。Class Model 構築用のデータについて BNC の詳細については 1 にまとめる。N-gram 構築用、N-gram Class Model の係数用、テストデータについては、ほぼ A と同様の内容である。BNC は、評価に必要の無い品詞などのタグ情報を全て除いた。また、各文の先頭では履歴に学習データ中には現れない特殊単語があるとした。出現回数が 4 回以下の Unigrams、Bigrams、Trigrams については全て取り除いた。

実験環境は OS は Turbolinux、CPU は Opteron 2GHz x 2、Memory は 12GB である。

6.2 Class Model の高速化

サンプリングを用いた高速化についての実験結果を示す。最初に trymove の結果である Δ を、trymoveS、trymoveSC の結果である Δ_{app} が、trymove を行うための必要条件としての指標として有効化どうかを確かめた。$\Delta(w,c) >= 0$ であるような事例を Δ_+、$\Delta_{app}(w,c) >= 0$ であるような事例を Δ_{app+} とおく。必要条件として有効かどうかを示す示す Recall と、高速化に寄与する候補集合がどれだけ少なくなったかを示す Refinement を次の通り定める。

$$Recall := \frac{|\Delta_+ \cap \Delta_{app+}|}{|\Delta_+|} \quad Refinement := \frac{|\Delta_{app+}|}{|\Delta_+|} \quad (9)$$

Corpus A において、trymoveS(C) のサンプリング点数を 1,2,4,8,16 とし、それぞれの Class 数が 16,32,64,128,256 の時の Recall と Refinement を求めた。サンプリング点数の増加と共に Recall が大きく、Refinement が小さくなっていることが分かる。また、trymove S に比べ trymove SC が有効に働いていることが分かる。

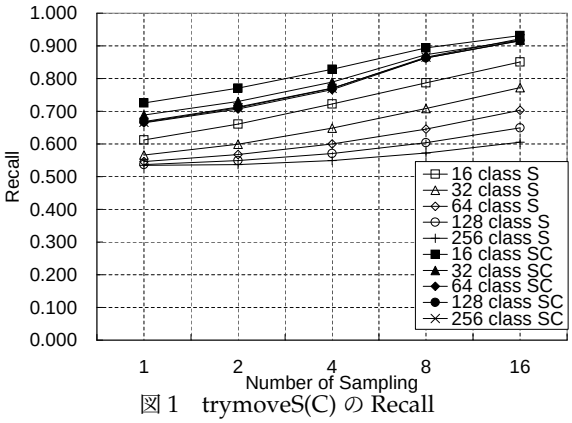

図 1　trymoveS(C) の Recall

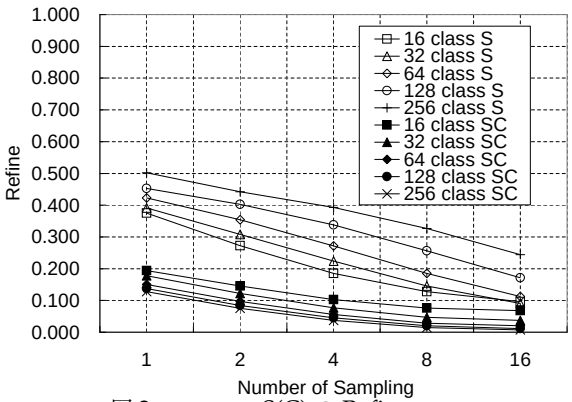

図 2　trymoveS(C) の Refinement

次に、通常の Exchange Algorithm(Baseline)、trymoveS、trymoveSC を用いた Exchange Algorithm S(+S)、Exchange Algorithm SC(+SC) と、trymoveSC と TopDown クラスタリングを組み合わせた TopDown Exchange Algorithm SC(+TD) を用いて単語を分類したときに収束に要した時間と、その分類を用いた Class Model によるテストデータの PP、N-gram の補間に用いた場合のテストデータの PP を求めた。

表 1　N-gram、Class Model 学習用データに用いた BNC Corpus の情報

コーパス	A	B	C
単語総数	1049079	8145528	100150946
Unigrams	11551	39210	142502
Bigrams	265907	1472601	8495670
Trigrams	297899	2528580	23996138

表 2　Corpus A（上段)、B（下段）での単語分類の所要時間 (s)

Class 数	Baseline	+S	+SC	+TD
32	42.8	21.2	14.1	11.2
64	174.1	68.6	36.8	22.3
128	658.0	255.2	98.1	54.3
256	2288.0	877.9	253.8	118.8
512	6304.0	2746.9	636.2	300.3
1024	20540.0	9197.3	1723	826.0
32	164.8	90.4	51.4	42.4
64	663.6	254.5	131.4	92.4
128	2675.9	535.2	357.5	202.4
256	10146.8	4051.4	1069.9	456.5
512	35793.4	17429.3	3379.0	1204.0
1024	115482.8	57741.4	10035.4	3562.9

表 3　Corpus A（上段)、B（下段）を学習データに用いた Class Model の PP

Class 数	Baseline	+S	+SC	+TD
32	385.6	382.1	383.2	383.9
64	312.9	317.4	331.5	329.6
128	300.1	302.0	301.7	309.6
256	289.2	288.5	290.4	293.4
512	286.9	288.9	287.8	289.1
1024	298.9	301.5	300.5	299.5
32	483.5	482.1	483.1	493.6
64	413.4	413.7	411.1	413.1
128	350.8	352.4	356.9	355.8
256	322.2	321.5	323.1	326.0
512	300.8	303.4	301.4	304.8
1024	287.6	287.9	288.4	289.0

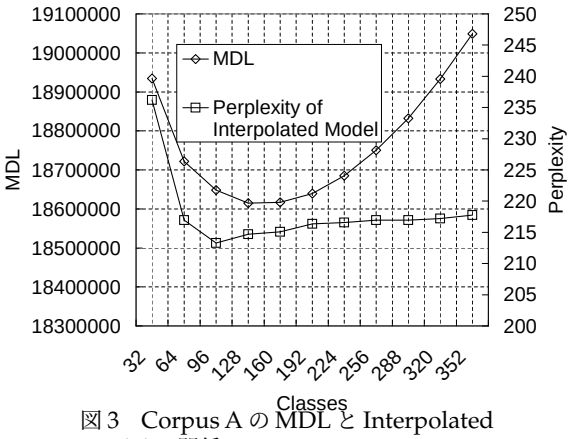

図 3　Corpus A の MDL と Interpolated Model の関係

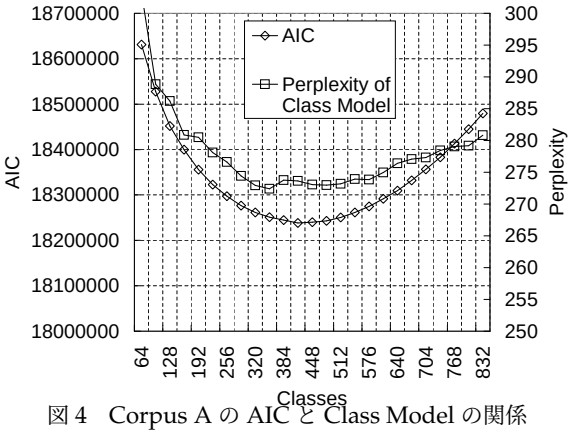

図 4　Corpus A の AIC と Class Model の関係

実験結果からは、trymoveS を用いることにより約 2 倍の高速化、trymoveSC を用いることにより約 10 倍の高速化、trymoveSC に TopDown クラスタリングを組み合わせることにより約 20 倍から 30 倍の高速化を達成できることがわかる。また、この時の分類精度は Exchange Algorithm を用いた場合と比べて、1% 以内の誤差範囲内であり、精度を保ったまま高速化を達成できていることがわかる。

6.3　最適な分類数の決定

MDL 原理、AIC 原理を用いて最適な Class 数が求められているかについての実験結果を示す。各 Class 数における分類後の MDL、AIC を求め、またその分類結果を用いた Class Model によりテストデータの PP、Class Model を N-gram の補間として用いた場合の PP を求めた。

図 3 では MDL が Interpolated Model における最適な Class 数の指針、図 4 では AIC が Class Model 単体における最適な Class 数の指針として有効であることを示している。MDL は AIC より小さめのモデルを選択するが、Interpolated Model は N-gram から情報を得られるのでより簡単なモデル、Class Model 単体ではより複雑なモデルが最適となることと、AIC、MDL のモデルの選好性が一致したものと考えられる。

6.4　大規模データコーパスに対する適用結果

最後にこれらの最適化を用いることにより、大規模データコーパスを用いて単語分類が行えることを示す。用いたのは Corpus C である。TopDown と trymove SC の高速化を用いたクラスタリングにより、Class 数が 1024 のものを約 6 時間（21773s）で行えた。この時同時に Class 数が 1,2,4,..,512 の分類結果も同時に得ている。この分類結果を用いた Class Model によりテストデータの PP は 284.6、Class Model と Trigram の補間を用いることによる PP は 179.4 であった。

7　おわりに

本稿では Class Model による単語分類の様々な高速化手法を紹介し、それらにより従来の Class Model による単語分類と比べ約 20 倍から 30 倍の

高速化を精度を保ったまま可能であることを示した。また、MDL 原理、AIC 原理を Class 数決定の指針に用いることが可能であることを示した。最後に上記の高速化手法を Class 数決定アルゴリズムに組み込むことが可能であり、Class 分類と Class 数決定を同時に行うアルゴリズムを紹介した。

改良すべき点は、TopDown クラスタリングはそれぞれの初期収束で、まだ時間を必要とするが、maxmove のように 1 回に一つの単語を移動させるのではなく、複数の単語を移動させることが考えられる他、TopDown において単純にそれぞれの Class をランダム以外の方法で分割する方法である。また用いるモデル自体を [7] のように変えることも考えられる。本稿で述べた高速化、最適な Class 数決定は多くの場合、他の手法でもそのまま用いることができる。今後はこうした高速化の他、学習データが疎になる、より複雑なモデル [6] において Class を用いることも考えていきたい。

8 謝辞

BNC の提供を東京大学辻井研より受けた。また辻井研の方々、特に鶴岡氏、松崎氏より貴重な助言や指導を頂いた。ここに感謝の意を表する。

参考文献

[1] Akaike, H. 1973. Information theory and an extension of the maximum likelihood principle, Second International Symposium on Information Theory, pp. 267-281.

[2] Andreas Stolcke and Stephen M. Omohundro. 1994. Best-first Model Merging for Hidden Markov Model Induction. Technical Report TR-94-003, Computer Science Division, University of California at Berkeley and International Science Institute.

[3] Chen, S. F. Goodman, J. 1996, 1998. An empircal study of smoothing techniques for language modeling. In Proceedings of the 34th Annual Meething of the Association for Computational Linguistics (ACL-96) pp.310-318.

[4] Dekang Lin and Patrick Pantel. 2002. Concept Discovery from Text. In Proceedings of Conference on Computational Linguistics 2002. pp. 577-583. Taipei, Taiwan.

[5] Dominic Widdow and Beate Dorow. 2002. A Graph Model for Unsupervised Lexical Acquisition. Proceedings of the 19th International Conference on Computational Linguistics, 1093-1099.

[6] Eugene Carniak, Immediate-head parsing for language models. In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, 2001.

[7] E. W. D. Whittaker and P. C. Woodland. 2001. Efficient Class-Based Language Modelling For Very Large Vocabularies. In Acoustics, Speech, and Signal Processing, Proceedings. pp. 545-548 vol. 1

[8] Hang Li. 2002. Word Clustering and Dismbiguatiaon based on Co-occurrence Data, Natural Language Engineering, 8(1), 25-42

[9] Hang Li and Naoki Abe. 1998. Word Clustering and Disambiguation Based on Co-occurrence data. Proceedings of the 18th International Conference on Computational Linguistics and the 36th Annual Meeting of Association for Computational Linguistics, 749-755.

[10] Inderjit S. Dhillon, Subramanyam Mallela and Rahul Kumar. 2002. Information Theoretic Feature Clustering for Text Classification. The Nineteeth International Conference on Machine Learning, Workshop on Text Learning.

[11] Martin, S., Liermann, J. and Ney, H. 1998. Algorithms for bigram and trigram word clustering. Speech Communication 24(1998) 19-37

[12] Matsuzaki, Takuya, Yusuke Miyao and Jun'ichi Tsujii. 2003. An Efficient Clustering Algorithm for Class-based Language Models. In the Proceedings of the Seventh Conference on Natural Language Learning (CoNLL) at HLT-NAACL 2003. pp. 119-126.

[13] Ney, H., Essen, U., Kneser, R., 1995. On the estimation of small probabilities by leaving one out. IEEE Trans. Pattern Anal. Machine Intell. 17(12), 1202-1212.

[14] Peter Cheeseman and John Stutz. 1996. Bayesian Classification (AutoClass): Theory and Results. In U. Fayyad, G. Piatetsky-Shapiro, P.Smyth and R. Uthurusamy(Eds.), Advances in Knowledge Discovery and Data Mining, 153-180. AAAIPress.

[15] P.F.Brown, V.J.Della Pietra, P.V.deSouza, J.C.Lai, and R.L.Mercer. 1992. Class-based n-gram models of natural language. Computational Linguistics, 18(4),1992.

[16] J. Rissanen. 1989. Stochastic Complexity in Statistical Inquiry. World Scientific Publishing Co., Singapore.

[17] 2003 年度未踏創造事業「汎用的データにおける確率モデルの抽出及びその利用」