効率的なテキストストリーム処理のための自己適応的分類器

A Self-adaptive Classifier for Efficient Text-Stream Processing

吉永 直樹*1 喜連川 優*1,2
Naoki Yoshinaga1 Masaru Kitsuregawa2,1

1 東京大学生産技術研究所
2 Institute of Industrial Science, the University of Tokyo
2国立情報学研究所
2 National Institute of Informatics

Abstract: This paper presents a self-adaptive classifier for efficient text stream processing. Our method speeds up a classifier trained with many conjunctive features while the classifier solves the classification problems in processing a given text stream. The key idea is to keep and reuse classification results for fundamental classification problems to solve the forthcoming classification problems. We explore two enumeration schemes, the least frequently used (LFU) and the least recently used (LRU), to select fundamental classification problems while processing the text stream. Experimental results with a Twitter stream on the day of the 2011 Great East Japan Earthquake confirmed that the proposed method accelerated a classifier for a state-of-the-art deterministic dependency parser by a factor of up to 5.

1 はじめに

Twitterに代表されるマイクロブログの普及により、世間一般の人が自身の体験や考えを気軽に発信する時代となっている。人々がマイクロブログを通じて発信する情報は、いまま実世界のあらゆる時間・空間を網羅するようになりつつあり、これをリアルタイムで解析することもすれば、自然災害の状況把握に基づく減災、また企業や自治体の公的機関が提供する商品やサービス、施策の問題への迅速な対応など、より良い社会の実現に繋がることが期待できる。

しかししながら、現在の自然言語処理技術でマイクロブログのテキストストリームを解析しようとする場合、テキストストリームの質（内容）・量（単位時間あたりの流量）が時間的に変化するという性質から、解析の頑健性とリアルタイム性を両立させることは困難となっている[1, 2]。現在のところ、既存研究では、表記ゆれの正規化[3]や未知語処理[4]など、主に質の多様性に焦点を当てて研究が行われている。

本稿では、テキストストリームを言語解析する上で問題となる上記の問題のうち特に「量」の変化に焦点を当て、自然言語処理で広く用いられる組み合わせ性に基づく線形分類器を、解析対象の分類問題に対し適応的に高速化させる手法を提案する。提案手法はテキストストリームの流量が増えるときに、地震、大雪など自然災害の発生[5]やイベントの実況など、特定の話題に関する投稿が増えるという観察に基づき、類似する内容の文が多数発信されるという仮定のもとで分類問題に共通する基本分類問題を動的に列挙し、その分類結果を再利用することで分類器を適応的に高速化する。

実験では、東日本大震災発生当日のテキストストリームを係り受け解析し、その際に生成される分類問題（のストリーム）に対して提案手法を適用することで、提案手法の有効性を評価する。

2 関連研究

単語分割や品詞解析、基本句同定や係り受け解析、意味理解となど多くの基礎言語解析において、その基本となる解析単位は文であり、効率的な解析アルゴリズムの多くは文単位での解析を高速化することを目的としている[6, 7, 8]。以下では、与えられたテキスト（文の集合）に対して適応的に解析を高速化する手法を紹介する。

Yoshinaga と Kitsuregawa[9, 10] は、自然言語処理で多用される組み合わせ素性に基づく分類器による分類を、事前に列挙した基本分類問題の分類結果を利用

*1 https://2011.twitter.com/ja/tps.html
バーサプトロンやSVMなどマージンに基づく二値分類器では与えられた分類問題において、ラベル \( y \in \{+1, -1\} \) を以下の式で用いて決定する（簡単のため、バイアス項は省略して議論を進める）。

\[
m(x; \phi, w) = w^T \phi(x) = \sum w_i \phi_i(x) 
\]

（1）

\[
y = \begin{cases} 
+1 & (m(x; \phi, w) \geq 0) \\
-1 & (m(x; \phi, w) < 0).
\end{cases}
\]

（2）

ここで \( m(x; \phi, w) \) は分離平面からのマージン、\( \phi_i \) は素性関数であり、\( w_i \) は学習により得られた \( \phi_i \) の重みである。

自然言語処理では文書分類など一部のタスクを除き、素性関数として言語的制約を表現する指示関数（二値関数）を用いることが多い。本稿では \( |\{0,1\} \) を返す二値関数のみを素性関数として想定し、以後マージン \( m(x; \phi, w) \) を以下の式で表現する。

\[
m(x; \phi, w) = \sum_{i \in \{ \phi(x) = 1 \}} w_i
\]

（3）

で表現する。また、素性関数 \( \phi_i \) を単に素性と呼び、\( x \) について \( \phi_i(x) = 1 \) となるとき、\( x \) が素性 \( \phi_i \) を含む、あるいは \( x \) で素性 \( \phi_i \) が発火する、と表現する。文法 \( \phi(x) \) の発火素数数として \( O(|\phi(x)|) \) で計算することができる。

既存手法では、分類タスクを共通して出現する基本分類問題 \( x \) に対して分類結果 \( M_{x_0} \equiv m(x; \phi, w) \) を事前に計算しておく、これを入力の分類問題の部品結果として用いることで分類を高速化する。

\[
m(x; \phi, w) = M_{x_0} + \sum_{i \in \{ \phi(x) = 1 \}, \phi_i(x) = 0} w_i
\]

（4）

where \( \forall i \in \{ \phi(x) = 1 \}, \phi_i(x) = 1. \)

式4が式3より高速に計算できるためには、基本分類 \( M_{x_0} \) の \( O(|\phi(x)|) \) より高速に取得できる必要があるが、素性に \( \phi \) を含む \( \phi_i \) が含まれる状況下では、その組み合わせ素性 \( \phi_{i,j} \) を構成する基本素性 \( \phi_{i,j} \) のみ確認すれば \( M_{x_0} \) を取得可能であるため、高速化できる。例えば、\( \phi = \{ \phi_1, \phi_2, \phi_3 \} \) とすると、\( m(x; \phi, w) \) の計算には組み合わせ素性を含む3つの素性の重みを加算する必要があるが、\( M_{x_0} \) を取得するには3つの基本素性 \( \phi_i \) のみを確認すればよい。一般的な議論をすると、元の組み合わせ素性を含む素性空間 \( \phi \) に対して、組み合わせ素性を除いて基本素性のみに縮小した素性空間 \( \phi' \) 考えれば、マージンを得るために確認する素性の数を多項式オーダから線形オーダに落とすことができる。

なお、式4による高速化の効果を最大とするためにでは \( x \) でのみ発火する素性 \( \phi_i(x) = 1, \phi_i(x) = 0 \)

3 背景知識

本節では、提案手法の土台となる線形分類器の高速化手法 [9] を説明する。この手法では、言語処理において用いられる組み合わせ素性に基づく線形分類器（または多項式カーネルに基づく非線形分類器 [15]）を高速化の対象としている。
の数がなるべく少なくなるような（直感的に言い換え
るとなるべくxに近い）基本分類問題x_eのマージン
M_{x_e}を部分結果として用いることが望ましい。しか
し、そのため、全てのありうるxについてM_e = m(x; \theta, w)
を事前に計算しようとするとO(2^\theta)の記憶領域が必要
となり、現実的ではない。そこで既存手法[9]では、分
類器を用いて解く言語処理タスクの入力を大量に用意
して実際に解く分類問題を列挙し、それらと共通する
部分分類問題を基本分類問題として採用する手法を提
案している。基本分類問題の選択にあたっては、その出
現頻度と式4において削減される計算コストの大きさ
に基づき、より有用性の高い基本分類問題を選択する。

抽出した基本分類問題x_eは発火する素性のインデッ
クス列（以下、素性列）で表現し、計算した分類結果
M(x_e)を合わせてトライに格納する。入力の分類問題
になるべく近い基本分類問題を式4で利用するため、各
基本素性の頻度を計算し、その頻度順で発火する基本
素性を並べてトライに保存・検索する素性列を構成す
る。これより、最も接頭辞検索により、入力の分類
問題に類似する基本分類問題をO(\phi'(x_e))で取得す
ることが可能となる。

4 提案手法

前節で述べた高速分類手法[9]では、部分結果とし
て利用する基本分類問題は事前に列挙することを想定
していた。本研究では、基本分類問題を動的に列挙す
ることで、テキストストリームに対して適応的に分類
を高速化する手法を提案する。これを実現するにあた
り解すべき問題は、1) マージンを保持する基本分類問
題をどう選ぶか。また、2) 基本分類問題をどのような
データ構造で管理するか、という2点である。以下
で、これらの課題に対する本研究での解を順に述べる。

4.1 基本分類問題の動的列挙

既存手法[9]では分類問題の頻度と計算の削減コス
トを考慮して基本分類問題を選択していたが、テキス
トストリームから分類問題を列挙する場合には頻度が
動的に変化するため、どの分類問題が（テキストストリ
ーム全体を通じて）有用な基本分類問題となっている
か事前に知ることはできない。従って、テキストストリ
ームの変化に合わせて必要な分類問題は追加し、不
要なものは削除するなどして基本分類問題の集合に適
応的に更新する必要がある。

Algorithm 1 に我々の提案する基本分類問題の動的
列挙に基づく適応的な分類手法を示す。提案手法では

Algorithm 1 基本分類問題の動的列挙に基づく分類

Input: x, \theta, \phi', w, \lambda, k
Output: m(x) \in \mathbb{R}, \lambda
1: initialize: x_e s.t. \phi'(x_e) = 0, m(x) = 0
2: repeat
3: \hat{i} = \arg\max_{i \in (j, \phi'_{j}(x_e) = 1, \phi(x_e) = 0)} FREQ(x_e)
4: \phi'_i(x_e) \leftarrow 1
5: if x_e \notin \lambda then
6: M_{x_e} \leftarrow m(x) + \sum_{j \in \lambda} w_i
7: if \|\lambda\| = k then
8: \lambda \leftarrow \lambda \cup \{x_e\}
9: \lambda \leftarrow \lambda \cup \{x_e\}
10: m(x) \leftarrow M_{x_e}
11: until \phi'(x_e) \neq \phi'(x)
12: return m(x), \lambda

観測された分類問題x_eから分類器の学習時に高頻度で
観測された基本素性\phi'を順に取り出し、基本分類問題
x_eを事前に構成する（3-4行目）。このようにして
構成されたx_eがその時点で保持する基本分類問題集
合\lambda_eに含まれていた場合には以前計算したマージン
M_{x_e}を返し、そうでない場合は式4を用いてマージン
M_{x_e} = m(x; \phi', w)を計算した後、x_eを基本分類問題
として\lambda_eに追加する（9-10行目）。

ここで重要のは、基本分類問題数が予め定められた
数に達した場合に、\lambda_eに保存された基本分類問題
から最もその後の分類の高速化に貢献しないと考えられ
る基本分類問題を取り除く関数USELESSである。本稿
では、一般的なキャッシュアルゴリズムにおけるキャッ
シュの管理ポリシーを参考に以下の2つの基準を提案・
比較する。

Least Frequently Used (LFU) テキストストリーム
中で観測された分類問題の頻度を計測し、高頻
度で参照される基本分類問題のみを保持するよう
に低頻度の基本分類問題を削除する。

useless_{lfu}(\lambda_e) = \arg\min_{x_e} FREQ(x_e) (5)

テキストストリーム中での基本分類問題の頻度は
space savingアルゴリズム[16]を用いて近似的
に計測し、頻度計測の対象から外れた基本分類問
題を削除する。

Least Recently Used (LRU) テキストストリーム
の流量が増えるときには、特定の話題に偏って情
報が発信されることが多い。この点を考慮する
と、より最近アクセスされた分類問題を基本分類
問題として保持することも有効と考えられる。

useless_{lru}(\lambda_e) = \arg\min_{x_e} TIME(x_e) (6)
4.2 ダブル配列を用いた基本分類問題の管理

基本分類問題を管理するデータ構造としてはダブル配列 [17]に基づくトライを利用する。トライは基本分類問題のキーに対してそのインデックス \(i(1 \leq i \leq k)\) を値として保持し、これを前節で述べた基本分類問題の列挙時に利用する。基本分類問題をダブル配列に基づくトライで管理することで、Algorithm 1 における検索・削除操作（それぞれ、5, 8 行目）が \(O(|\phi(x)|)\) ではなく、\(O(1)\) で行えることが注目される。

従来、ダブル配列は検索は高速であるものの、動的な更新は低速なデータ構造であると認識されている。しかしながら近年の研究 [19] は、更新に実装しやすければ、更新についてもハッシュやハットトライ [20] などに匹敵する性能が可能となっている。我々は [19] に改良を施した動的ダブル配列を用いて基本分類問題の管理を行う。

ダブル配列を用いて基本分類問題の管理を効率的に行うためには、検索・更新・削除の際にアクセスするノードの数をなるべく少なくすることが重要である。そこで、以下のアルゴリズム・データ構成的な工夫により、トライのノード数を可能な限り削減することを試みた。

可変長バイト符号化を用いた素性表現の圧縮

4 低頻度の素性に関しては、その素性に関する全ての素性の重みを正に分けて加算したものの方が正確な上限・下限を与える場合があるので、両者を比較してより正確な上限・下限を利用することになる。ただし、トライから順に通ることで、同じ素性を二度チェックする必要がなくなるため、ダブル配列から不要となった基本分類問題を削除する際には、その後すぐに別の基本分類問題を格納することを考慮し、削除によって空いた領域を詰め込み直すこと [19] は行われない。

5 評価実験

本節では、係り受け解析の分類器に提案手法を適用し、Twitter のテキストストリームを解析対象として提案手法の有効性を検証する。以下の実験には Intel® Core™ i7-3720QM 2.6GHz CPU と主記憶 16GB を備えたサーバ上で行った。

提案用のテキストストリームとしては Twitter REST API を用いて当研究室で収集したツイートから、東日本大震災発生当時である 2011 年 3 月 11 日の 12:00 以降のツイートを選んで用いた。係り受け解析の入力は文であるため、句点等を手がかりに文分割を行った。

本節では、係り受け解析器の解析速度を計測することが望ましいのが今回用いた係り受け解析器では分類器による分類時間が大半の時間を占めること [9]、またより直接的に高速化の効果を計測するため、日本語係り受け解析器 J.DepP を用いて上記のツイートを時系列順に係り受け解析し、実際に J.DepP が解いた係り受け解析の分類問題を時系列順で列挙して、これをストリームとみなし提案する分類器で分類し、その効果を確認することとした。

J.DepP は関々野 [23] が提案した線形時間アルゴリズムに基づき、分類器を用いて与えられた二つの文節間の係り受け関係を的確に判定しながら文全体の係り受け構造を決定する、なるべく高精度の解析
表1: 評価に用いたテキストストリーム

| 手法       | 分類速度 | メモリ | 速度向上比 |
|------------|---------|-------|-----------|
| ペースライン [9] | 0.0325  | 31.5  | 1.00      |
| 提案手法 (LFU) | k = 210 | 0.0251 | 38.8 | 1.29     |
| 提案手法 (LRF) | k = 222 | 0.0103 | 369.1 | 3.49 |
| 提案手法 (LRF) | k = 226 | 0.0074 | 3845.9 | 4.37 |

図1: 東日本大震災当日のテキストストリームにおけるツイート数/分の変化

図2: 東日本大震災当日のテキストストリームにおける分類問題数/分の変化

実験結果から、選択した手法は、APIの制限のため、全体のツイートの一部をサンプルしたものであるが、それでも地震が発生した14時46分18秒を境として、災害によりツイート数が極端に増えていることが確認できる。また、ツイート数よりも、分類問題数の方が地震の発生前後で単位時間あたりの数の増加が大きいこととは興味深い。今回用いた係り受け解析器の分類問題数は、文中の文節の数に比例するため、震災発生前後で個々のツイート中の文の長さが増加したものと推測される。

なお、全ツイート中で公式RT（コピー）が占める割合は分類問題数で47%程度であった。ツイート単位で解析結果をキャッシュするような単純なアプローチをとる場合には、オーバーヘッドを全て無視した場合でも、高々1.89(= 43221268/(43221268 - 20383651))倍程度の高速化に留まると考えられる。

表2に表1のテキストストリームを係り受け解析する際に生成された分類問題数を、時系列順に処理したときの分類速度を示す。提案手法については、解析中に保持する基本分類問題数を217から226まで2倍刻みに変化させて分類速度を測定した。表中、ベースラインは式3を用いて分類したときの分類時間、手法[9]は分類手法を学習した新記事コーパス（脚注10）から事前に基本分類を問題を静的に列挙し、式4で分類する分類器である。表から分かるように、提案手法を用いることでベースラインに対して、最大5倍の高速化を達成した。
図3: 東日本大震災当日のテキストストリームにおける1分間のツイートの処理に分類器が要する時間の変化

ここで、基本分類問題数を増やしていくと、基本分類問題を頻度（LFU）ではなくアクセス時間（LUF）に基づいて列挙する方が、高速化が顕著となることは興味深い。要因としては、時間的に近いツイート同士に類似した内容のテキストが多いためではないかと推察される。また、手法[9]で列挙された基本分類問題の総数は2,903,138であるが、提案手法では$k=16384$とこれと比べて大幅に少ない基本分類問題数で同程度の分類速度が得られていることは注目に値する。

図3に表1のテキストストリームを係り受け解析する際、1分間のツイートを処理するのに分類器が必要とした分類時間の推移を示す。表中の縦軸は、震災発生时刻までのベースライン手法の平均分類時間を1として分類時間の正規化したものである。ベースライン手法および既存手法[9]では、分類問題の増加（図2）に応じて分類時間も増大しているが、提案手法では分類時間の増加は大幅に抑えられている。このことから、提案手法はテキストストリームをリアルタイム解析をする上でより頑健であると結論づけることができる。

6 普び

本稿ではTwitterなどのテキストストリームを頑健にリアルタイム解析することを目的として、言語解析を用意と考えられる組み合わせ素性に基づく線形分類器を用いた決定的な言語解析は、品詞分類や単語分割[27]、構文解析[28]といった多くの言語処理タスクで最高精度を達成していること、また従来構造学習が有効とされる固有表現認識などにおいても、構造訳[29]（あるいはuptraining[30]）と呼ばれる手法で構造学習と同程度の精度を達成できるとの報告があることから、本研究の有効性が期待できる言語解析は、言語処理全般にわたると考えられる。

参考文献

[1] Kevin Gimpel, Nathan Schneider, Brendan O’Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In Proceedings of ACL-HLT, pp. 42–47, 2011.

[2] Jennifer Foster, Öziem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. #hardtoparse: POS tagging and parsing the Twitterverse. In Proceedings of the AAAI-11 Workshop on Analyzing Microtext, 2011.

[3] Bo Han and Timothy Baldwin. Lexical normalisation of short text messages: Makk sens a #twitter. In Proceedings of ACL-HLT, pp. 368–378, 2011.

[4] Ryoei Sasano, Sadao Kurohashi, and Manabu Okumura. A simple approach to unknown word processing in Japanese morphological analysis. In Proceedings of IJCNLP, pp. 162–170, 2013.

[5] Takeshi Sakaki, Fujio Toriumi, and Yutaka Matsuo. Tweet trend analysis in an emergency situation. In Proceedings of the Special Workshop on Internet and Disasters, 2011.

[6] Nobuhiro Kaji, Yasuhiro Fujiwara, Naoki Yoshinaga, and Masaru Kitsuregawa. Efficient staggered decoding for sequence labeling. In Proceedings of ACL, pp. 485–494, 2010.

[7] Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. Dual decomposition for parsing with non-projective head automata. In Proceedings of EMNLP, pp. 1288–1298, 2010.
[8] Alexander Rush and Slav Petrov. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of NAACL-HLT*, pp. 498–507, 2012.

[9] Naoki Yoshinaga and Masaru Kitsuregawa. Polynomial to linear: Efficient classification with conjunctive features. In *Proceedings of EMNLP*, pp. 1542–1551, 2009.

[10] Naoki Yoshinaga and Masaru Kitsuregawa. Efficient classification with conjunctive features. *Journal of Information Processing*, Vol. 20, No. 1, pp. 228–227.

[11] Vivek Srikumar, Gourab Kundu, and Dan Roth. On amortizing inference cost for structured prediction. In *Proceedings of EMNLP-CoNLL*, pp. 1114–1124, 2012.

[12] Gourab Kundu, Vivek Srikumar, and Dan Roth. Margin-based decomposed amortized inference. In *Proceedings of EMNLP*, pp. 905–913, 2013.

[13] Henning Wachsmuth, Benno Stein, and Gregor Engels. Constructing efficient information extraction pipelines. In *Proceedings of CIKM*, pp. 2237–2240, 2011.

[14] Henning Wachsmuth, Benno Stein, and Gregor Engels. Learning efficient information extraction on heterogeneous texts. In *Proceedings of IJCNLP*, pp. 534–542, 2013.

[15] Hideki Isozaki and Hideto Kazawa. Efficient support vector classifiers for named entity recognition. In *Proceedings of COLING 2002*, pp. 1–7, 2002.

[16] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of ICDT*, pp. 398–412, 2005.

[17] Jun ichi Aoe. An efficient digital search algorithm by using a double-array structure. *IEEE Transactions on Software Engineering*, Vol. 15, No. 9, pp. 1066–1077, 1989.

[18] Susumu Yata, Masaki Oono, Kazuhiro Morita, Masao Fuketa, and Jun ichi Aoe. An efficient deletion method for a minimal prefix double array. *Journal of Software: Practice and Experience*, Vol. 37, No. 5, pp. 523–534, 2007.

[19] 矢田晋, 田村雅浩, 森田和宏, 泷田正雄. 青江順一. ダブル配列による動的辞書の構成と評価. 第71回情報処理学会全国大会講演論文集, pp. 1263–1264, 2009.

[20] Nikolas Askitis and Ranjan Sinha. HAT-trie: A cache-conscious trie-based data structure for strings. In *Proceedings of the Thirtieth Australasian Conference on Computer Science*, pp. 97–105, 2007.

[21] Hugh E. Williams and Justin Zobel. Compressing integers for fast file access. *The Computer Journal*, Vol. 42, No. 3, pp. 193–201, 1999.

[22] Makoto Yasuhara, Toru Tanaka, Junya Norimatsu, and Mikio Yamamoto. An efficient language model using double-array structures. In *Proceedings of EMNLP*, pp. 222–232, 2013.

[23] 工藤野志. 日本語係り受け解析の線形時間アルゴリズム. 言語処理学会論文誌, Vol. 14, No. 1, pp. 3–16, 2007.

[24] 黒橋隆夫, 長尾真. 京都大学テキストコーバス・プロジェクト. 言語処理学会第3回国会次大会発表論文集, pp. 115–118, 1997.

[25] Masakazu Iwatate. *Development of Pairwise Comparison-based Japanese Dependency Parsers and Application to Corpus Annotation*. PhD thesis, 2012.

[26] 岩立将和, 浅原正幸, 松本裕治. ツーナメントモデルを用いた日本語係り受け解析. 言語処理学会論文誌, Vol. 15, No. 5, pp. 169–185, 2008.

[27] Graham Neubig, Yusuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of ACL*, pp. 529–533, 2011.

[28] Joakim Nivre and Ryan McDonald. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-HLT*, pp. 950–958, 2008.

[29] Percy Liang, Hal Daumé III, and Dan Klein. Structure compilation: trading structure for features. In *Proceedings of ICML*, pp. 592–599, 2008.

[30] Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. Uptraining for accurate deterministic question parsing. In *Proceedings of EMNLP*, pp. 705–713, 2010.