社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

半構造化テキストの分類のための ブースティングアルゴリズム

工藤　　拓†　　松本　裕治†

† 奈良先端科学技術大学院大学 情報科学研究科　〒 630–0192 奈良県生駒市高山町 8916-5
E-mail: †taku-ku@is.aist-nara.ac.jp, ††matsu@is.asit-nara.ac.jp

あらまし　近年, テキスト分類は, 単純なトピック分類から, 文のモダリティ, 意見性, 主観性といった書き手の意図に基づく分類へと, そのタスクの多様化が進んでいる. それに伴い, 単語の集合 (bag-of-words) を素性とする古典的手法では十分な精度を得にくくなっている. 精度向上には, テキストの構造 (構文/レイアウト) を考慮する必要があるが, 恣意的に選択された部分構造のみを用いた手法が多い. 本稿では, 構造を考慮したテキスト分類 (半構造化テキスト分類) に向け, 部分木を素性とする decision stumps と, それを弱学習器とする Boosting アルゴリズムを提案する. また, Tree Kernel を用いた SVM との関連性, 及び本手法の利点について言及する. 実データを用いた実験により, 提案手法の有効性を検証する.

キーワード　テキスト分類, 半構造化テキスト, Decision Stumps, Boosting

A Boosting Algorithm for Classification of Semi-Structured Text

Taku KUDO† and Yuji MATSUMOTO†

† Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630–0192 Japan
E-mail: †taku-ku@is.aist-nara.ac.jp, ††matsu@is.asit-nara.ac.jp

Abstract　The research focus in text classification has expanded from a simple topic identification to a more challenging task, such as opinion/modality identification. For the latter, the traditional bag-of-word representations are not sufficient, and a richer, structural representation will be required. Accordingly, learning algorithms must be able to handle such sub-structures observed in text. In this paper, we propose a Boosting algorithm that captures sub-structures embedded in text. The proposal consists of i) decision stumps that use subtrees as features and ii) Boosting algorithm in which the subtree-based decision stumps are applied as weak learners. We also discuss a relation between our algorithm and SVM with Tree Kernel. Two experiments on the opinion/modality classification tasks confirm that subtree features are important. Our Boosting algorithm is computationally efficient for classification tasks involving discrete structural features.

Key words　Text Classification, Semi-Structured Text, Decision Stumps, Boosting

1. はじめに

SVM や Boosting といった機械学習手法が, テキスト分類に応用され, 多くの成功をおさめている [7], [11]. テキスト分類では, 一般に単語の集合 (bag-of-words 以下 BOW) を素性として用い, 政治, 経済, スポーツといったカテゴリにテキストを分類する. 個々の単語が, これらのカテゴリを特徴付けるのに十分な情報を与えるため, BOW といった単純な素性でも十分な精度を達成できる.

一方, テキストマイニングの分野では, 「お客様の声」や「現場の声」といったテキストデータから, 早期に危機の茅や要求を発見, 分析するための要素技術が求められている. 例えば, コールセンターログからの知識発見, 自由記述アンケートや製品レビュー記事からの不満要素の発見といったタスクがそれにあたる. これらのタスクでは, 旧来の文書分類に典型的なカテゴリは用いられず, 主観的/客観的に述べているのか, (ある製品を) 誉めている/けなしているのか, 背景/結果/結論を述べているのか, といった書き手の意図に関するカテゴリが使われる. さらに, 分類単位が, 文書といった大きな単位から, パッセージ, 文のような小さな単位に変化している. これらは, テキスト分類には変わりないが, BOW といった素朴な素性では, 高い精度が得られにくい. 単純には, 固定長の N-グラム, 係り受け関係 (係り元, 係り先のペア) といった, 有効そうな素性を発見的に追加することで, ある程度精度向上が期待できる. しかし, これらの手法は, タスク依存性が強く, テキスト分類全般に適用できる汎用的な手法とは言い難い.

本稿では, テキストが単語の集合であると仮定したり, 部分構造を発見的に追加する従来法に対し, テキストが持つ「構造」を直接考慮する学習, 分類アルゴリズムを提案する. ここでの構造とは, 単語の系列, 係り受け構造といった**ラベル付き順序木**を指す. この構造は, 自然言語処理において一般的なものである. さらに HTML/XML 文書もラベル付き順序木として定式化できる.

以下, 次章では提案手法の詳細, 3 章でその実装方法について説明する. 4 章で Tree Kernel を用いた SVM と提案手法の関連性に言及する. 5 章で評価実験とそれらの結果を報告し, 6 章でまとめる.

2. 提 案 手 法

詳細に入る前に, 木に対するいくつかの定義, 表記について言及する.

2.1 準　　備

［定義 1］ ラベル付き順序木
ラベル付き順序木は, すべてのノードに一意のラベルが付与されており, 兄弟関係に順序が与えられる木である. □

［定義 2］ 部分木
t と u をラベル付き順序木とする. ある木 t が, u にマッチする, もしくは, t が u の部分木である ($t \subseteq u$) とは, t と u のノード間に, 1 対 1 の写像関数 ψ が定義できることである. ただし, ψ は, 以下の 3 つの条件を満たす. (1) ψ は, 親子関係を保存する. (2) ψ は, 兄弟の順序関係を保存する. (3) ψ はラベルを保存する. □

本稿では, ラベル付き順序木を単純に順序木, もしくは木と呼ぶ. さらに, 木 t 中のすべての部分木の集合を $\mathcal{S}(t)$ ($\mathcal{S}(t) = \{t' | t' \subseteq t\}$), さらに 木 t のノードの数を $|t|$ と表記する.

2.2 ラベル付き順序木の分類問題

ラベル付き順序木の分類問題 (木の分類問題) とは, L 個の学習データ $T = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^{L}$ から, 分類関数 $f(\mathbf{x}) : \mathcal{X} \to \{\pm 1\}$ を導出することである. ただし, $\mathbf{x}_i \in \mathcal{X}$ はラベル付き順序木であり, $y_i \in \{\pm 1\}$ は, 木 $\mathbf{x}_i$ に付与されたラベルである (ここでは, 2 値分類問題を考える). 通常の学習, 分類問題との違いは, 事例 $\mathbf{x}_i$ が, 数値ベクトルで表現されるのではなく, ラベル付き順序木として表現される点にある.

2.3 Decision Stumps for Trees

Decision Stumps は, 1 つの素性の有無に基づき分類を行う, 単純なアルゴリズムである. Decision Stumps は, 深さ 1 の Decision Tree (決定木) と同一である. Schapire らは, 個々の単語を 1 つの素性とする Decision Stumps を用い, テキスト分類を行っている [11]. 本稿では, ラベル付き順序木の分類問題に対し, 以下のような部分木の有無に基づく Decision Stumps を考える.

［定義 3］ Decision Stumps for Trees
t および $\mathbf{x}$ を, ラベル付き順序木, $y \in \{\pm 1\}$ を, クラスラベルとする. 木を分類するための Decision Stumps を以下のように定義する.

$$h_{\langle t,y \rangle}(\mathbf{x}) \stackrel{\text{def}}{=} \begin{cases} y & t \subseteq \mathbf{x} \\ -y & otherwise. \end{cases}$$
$$= y \cdot (2I(t \subseteq \mathbf{x}) - 1) \quad \square$$

分類器のパラメータは, 木 t と ラベル y の組 $\langle t, y \rangle$ である. 以後, この組を **ルール** と呼ぶ.

Decision Stumps の学習は, 学習データ $T = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^{L}$ に対するエラー率を最小にするルール $\langle \hat{t}, \hat{y} \rangle$ を導出することで実現できる.

$$\langle \hat{t}, \hat{y} \rangle = \underset{t \in \mathcal{F}, y \in \{\pm 1\}}{\operatorname{argmin}} \frac{1}{2L} \sum_{i=1}^{L} (1 - y_i \cdot h_{\langle t,y \rangle}(\mathbf{x}_i)) \quad (1)$$

ただし, $\mathcal{F}$ は, 全部分木の集合 (**素性集合**) である ($\mathcal{F} = \bigcup_{i=1}^{L} \mathcal{S}(\mathbf{x}_i)$). ここで, ルール $\langle t, y \rangle$ に対する $gain(\langle t, y \rangle)$ を以下のように定義する.

$$gain(\langle t, y \rangle) \stackrel{\text{def}}{=} \sum_{i=1}^{L} y_i \cdot h_{\langle t,y \rangle}(\mathbf{x}_i) \quad (2)$$

gain を用いると, (1) の最小化問題は, 以下の最大化問題と等価となる.

$$\langle \hat{t}, \hat{y} \rangle = \underset{t \in \mathcal{F}, y \in \{\pm 1\}}{\operatorname{argmax}} gain(\langle t, y \rangle)$$

本稿では, エラー率最小のかわりに, gain の最大化を用いて, Decision Stumps の学習を定式化する.

2.4 Boosting の適用

部分木を素性とする Decision Stumps は, 1 つの部分木の有無に基づき分類を行うため, それ単独では精度が悪い. しかし, Boosting [5] を用いてその精度を向上させることができる. Boosting は, 逐次 **弱学習器** (ここでは, 部分木を素性とする Decision Stumps) を構築し, それらの重み付き多数決によって, 最終的な分類器 f を構成する. k 番目の弱学習器は, それぞれ異なった事例分布 (重み) $\mathbf{d}^{(k)}$ を用いて学習される. $\mathbf{d}^{(k)} = (d_1^{(k)}, \ldots, d_L^{(k)})$, (ただし $\sum_{i=1}^{N} d_i = 1, d_i^k \geqq 0$). 分布 $\mathbf{d}^{(k)}$ は, 分類が困難な

事例に高い重みが与えられるように逐次更新される．弱学習器が無作為分類よりも精度が良いことさえ満足すれば，単一の弱学習器より，Boosting の汎化能力が高くなることが理論的に証明されている [5]．Decision Stumps を弱学習器とする Boosting を構築するには，分布 $\mathbf{d}$ を考慮するよう 式 (2) を再定義すればよい．

$$gain(\langle t, y\rangle) \stackrel{\text{def}}{=} \sum_{i=1}^{L} d_i \cdot y_i \cdot h_{\langle t,y\rangle}(\mathbf{x}_i)$$

AdaBoost [5] は Boosting の代表的なアルゴリズムであるが，本稿では，漸近特性の良さから，Breiman らによる Arc-GV [3] を用いる．

3. 実　　装

本章では，最適なルール $\langle \hat{t}, \hat{y}\rangle$ を発見するための高速なアルゴリズムを提案する．最適ルール発見問題は以下のように定式化される．

［問題 1］ 最適ルール発見問題

学習データ $T = \{\langle \mathbf{x}_1, y_1, d_1\rangle, \ldots, \langle \mathbf{x}_L, y_L, d_L\rangle\}$ (ただし，$\mathbf{x}_i$ は順序木，$y_i \in \{\pm 1\}$ は木に対応するクラスラベル，d_i ($\sum_{i=1}^{L} d_i = 1$, $d_i \geqq 0$) は，木の重み) が与えらえた時，gain を最大にするルール $\langle \hat{t}, \hat{y}\rangle$ を発見せよ．(ただし，$\langle \hat{t}, \hat{y}\rangle = \text{argmax}_{t\in\mathcal{F}, y\in\{\pm 1\}} d_i y_i h_{\langle t,y\rangle}(\mathbf{x}_i)$, $\mathcal{F} = \bigcup_{i=1}^{L} \mathcal{S}(\mathbf{x}_i)$)．

単純な方法は，全部分木 $\mathcal{F}$ を陽に列挙し，そこから最大の gain を与える部分木を選択することであろう．しかし，部分木の個数は木のサイズに対し指数的に増えていくために，こういったしらみつぶしの方法は実問題には適用困難である．実際に，木の集合から全部分木を完全に列挙する問題は，NP 困難であることが分かっている．

本稿では，最適ルール発見問題のための効率良いアルゴリズムを提案する．提案手法は，以下の 3 つの戦略から構成される．基本的には，分枝限定法 (branch-and-bound) の考え方と同一である．

（1） 木の集合から，全部分木を完全に重複なく列挙するための正準的な探索空間を定義する

（2） 1 で定義した探索空間を深さ優先探索し，最大の gain を与える部分木を発見する．

（3） 1, 2 だけでは，全部分木をしらみつぶしに列挙してることと変わりない．そこで，gain の上限値を見積もり，探索空間を枝刈りする．

この 3 つの戦略について，順に説明する．

3.1 部分木の列挙方法

Abe と Zaki は，木から，その部分木を，完全に重複なく列挙するアルゴリズム**最右拡張**をそれぞれ独立に提案している [1], [13]．最右拡張は，部分集合を列挙するアルゴリズム Set Enumuration Tree [2] の部分木への自然な拡張となっている．最右拡張では，まず，サイズ 1 の木が列挙される．そして，サイズ $(k-1)$ の木に 1 つのノードを追加することでサイズ k の木を構築する．この手続きを再帰的に適用することで全部分木を列挙する．しかし，任意の位置にノードを追加すると重複する木を生成してしまうため，ノードの追加に一種の制限を設ける．この制限が最右拡張の鍵となるアイデアである．以下に最右拡張の定義を与える．

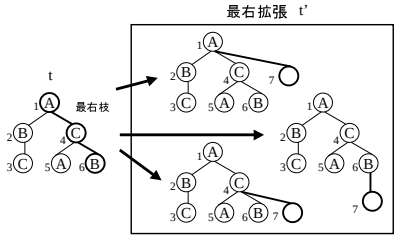

図 1　最 右 拡 張

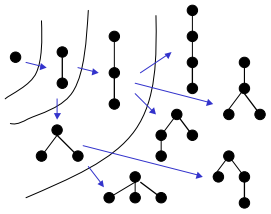

図 2　最右拡張の再帰 (探索木)

［定義 4］ **最右拡張** [1]

木 t に 1 つのノード s を追加して得られる順序木 t' を，木 t の 最右拡張と定義する．ただし，ノードの追加には以下の制約がある．(1) ノードは，t の最右枝 (ルートから常に右端のノードを葉まで選ぶ) に追加される．(2) 追加されるノードは，最右の兄弟 (末弟) である．□

図 1 に最右拡張の例を示す．便宜上，木 t は前順序 (pre-order) でノードに順序が付与されているものとする．図 1 の例では，位置 7 のノードが最右拡張で追加されるノードとなる．また，追加される位置の候補は 3 種類 (位置 1,4,6) あるが，ラベルの候補集合を考えると，ラベルの種類数 × 位置数 (図 1 では，$|\{A, B, C\}| \times 3 = 9$) 通りの木 t' が，最右拡張より生成される．

さらに，最右拡張を，再帰的に適用することで，一種の「探索木」を構築するとができる．図 2 に，その探索木の例を示す．図 2 中では，便宜的にラベルの種類を 1 種類のみとしている．この探索木を探索することで，全部分木を，完全に重複なく列挙できる．

3.2 探索空間の枝刈り

前章で定義した探索空間を分枝限定法 (branch-and-bound) の考えに従い枝刈りする．本章では，その具体的な手法について述べる．定理 1 (Morishita [9] の拡張) は，t の全上位木 $t', (\forall t' \supseteqq t)$ に対する $gain(\langle t', y\rangle)$ の上限値を与える．

［定理 1］ gain の上限値: $\mu(t)$

t の全上位木 $t', (\forall t' \supseteqq t)$, $y \in \{\pm 1\}$ について，ルール $\langle t', y\rangle$ の gain は $\mu(t)$ を上限値とする ($gain(\langle t' y\rangle) \leqq \mu(t)$)．ただし，$\mu(t)$ は以下で与えられる．

$$\mu(t) \stackrel{\text{def}}{=} \max\Big(2 \sum_{\{i|y_i=+1, t\subseteqq \mathbf{x}_i\}} d_i - \sum_{i=1}^{L} y_i \cdot d_i,\ 2 \sum_{\{i|y_i=-1, t\subseteqq \mathbf{x}_i\}} d_i + \sum_{i=1}^{L} y_i \cdot d_i\Big).$$

証明

$$gain(\langle t', y\rangle) = \sum_{i=1}^{L} d_i y_i \cdot y \cdot (2I(t' \subseteqq \mathbf{x}_i) - 1)$$

ここで, $y = +1$ の場合に着目すると

$$\begin{aligned} gain(\langle t', +1\rangle) &= 2\Big(\sum_{\{i|y_i=+1, t' \subseteqq \mathbf{x}_i\}} d_i \quad - \sum_{\{i|y_i=-1, t' \subseteqq \mathbf{x}_i\}} d_i \Big) \\ &\quad - \sum_{i=1}^{L} y_i \cdot d_i \\ &\leqq 2 \sum_{\{i|y_i=+1, t' \subseteqq \mathbf{x}_i\}} d_i - \sum_{i=1}^{L} y_i \cdot d_i \\ &\leqq 2 \sum_{\{i|y_i=+1, t \subseteqq \mathbf{x}_i\}} d_i - \sum_{i=1}^{L} y_i \cdot d_i \end{aligned}$$

($\because$ t の全上位木 $t', (\forall t' \supseteqq t)$ について $|\{i|y_i = +1, t' \subseteqq \mathbf{x}_i\}| \leqq |\{i|y_i = +1, t \subseteqq \mathbf{x}_i\}|$ が成立するため.)
同様に,

$$gain(\langle t', -1\rangle) \leqq 2 \sum_{\{i|y_i=-1, t \subseteqq \mathbf{x}_i\}} d_i + \sum_{i=1}^{L} y_i \cdot d_i$$

以上より, t の全上位木 $t' \supseteqq t$, $y \in \{\pm 1\}$ について

$$gain(\langle t', y\rangle) \leqq \mu(t) \quad \square.$$

定理 1 より, gain の上限値が与えられるので, 分枝限定法を用い, 探索空間を枝刈りすることができる. 具体的には, まず, 最右拡張が定義する探索空間を深さ優先探索しながら, 準最適 gain (これまで探索した中で最大の gain) τ と, それに対応する準最適ルールを保持する. もし, $gain(\langle t, y\rangle) > \tau$ となるようなルールが探索中に見つかれば, 準最適ルール, 準最適解をそれぞれ更新する.

ここで, 木 t について, もし $\mu(t) \leqq \tau$ ならば, t の全上位木 t' の gain は, τ 以下であるため, t が張る部分空間は安全に枝刈りできる. 逆に $\mu(t) > \tau$ ならば, $gain(\langle t', y\rangle) > \tau$ となる上位木 t' が存在する可能性があるため, 枝刈りできない. また, $\mu(t) > \tau$ となり, ノード s が, 木 t に最右拡張により追加される場合でも, $\mu(s) < \tau$ ならば, 拡張された木 t' が張る部分空間を枝刈りできる.

4. SVM との関連性

SVM は, Kernel 関数と組み合わせることで, 任意の構造を持った事例を学習, 分類できる. ラベル付き順序木を分類するための Kernel として Tree Kernel [4], [8] があり, 全部分木を個別の素性とするような写像関数を与える. 部分木を最小単位の情報として用いていることから, 部分木を素性とする Decision Stumps と Tree Kenrnel に基づく SVM は, 素性空間という観点において, 本質的に同一であるといえる.

2 つのアルゴリズムの違いは, マージンのノルムである (Boosting は l_1 ノルム SVMs は l_2 ノルム). 文献 [10] では, このノルムの違いを, モデルのスパース性という観点で議論している. 詳細は [10] に譲るが, SVM は, **事例スパース**の解を導出し, できるだけ少数の事例でモデルを表現しようとする. SVM の分離平面 ($\mathbf{w}$) は, 学習事例の線形結合で与えられ, 非 0 の係数が付いた事例が, **サポートベクター**と呼ばれるのは上記の理由からである. 一方, Boosting は, **素性スパース**の解を導出する. つまり, できるだけ少数の素性で $\mathbf{w}$ を表現しようとする.

SVM と Boosting を精度という観点で比較することは, タスク依存性が強いため, ここでは行わない. しかし, Boosting を用いる提案手法は, スパース素性の性質から, 以下のような「現実的な」利点がある.

- 解釈のしやすさ

できるだけ少数の素性で分類し, 分類自身が陽に実行されるため, どのような素性が分類に使用されているか, 分類に寄与する素性は何か, といった分析が行いやすい. SVM では, 素性空間が陰に表現されるため, そのような分析を与えにくい.

- 高速な分類

スパースな素性空間は, 高速な分類を可能にする. Tree Kernel の計算量は, $O(|N_1||N_2|)$ である (ただし N_1 と N_2 は内積をとる 2 つの木である). さらに Kernel 法に基づく分類アルゴリズムはサポートベクターの数 L' に依存する. つまり, SVM の分類コストは, $O(L'|N_1||N_2|)$ となり非常に大きい. この問題は, 実際の応用において障壁になっている.

5. 実験と考察

5.1 実験環境, 設定

実験は, 以下の 2 種類のタスクで行った.

- PHS レビュー分類 (**PHS**)

PHS ユーザに, 良い点/悪い点を区別してレビューを投稿するよう指示した掲示板のデータである. 分類単位は「文」, 文数は 5,741, カテゴリは「良い点」「悪い点」の 2 つである.

- 文のモダリティ分類 (**MOD**)

被験者 1 名が, 田村ら [14] の大分類に従い, 99 年毎日新聞の社説からランダムに選んだ 60 記事に, 文単位のモダリティ付与したデータである. 分類単位は「文」, 文数は 1,710, カテゴリは, 「意見」「断定」「叙述」の 3 つである.

3 つのタスクのデータの一例を図 3 に示す.

文の表現方法として, 以下の 3 つを用いた

- bag-of-words (**bow**), ベースライン

構造は考慮せず, 単語の有無を素性とする. 具体的には, まず, ChaSen(注1)を用いて単語を切り出す. ただし, 単語の表層形ではなく, 原形を用いた. 提案手法では, サイズ 1 の構造のみが素性集合となる. これは Boostexter [11] と同一である.

- 係り受け (**dep**)

単語単位の係り受け構造として文を表現する. 具体的には, CaboCha(注2)を使い, 文節単位の係り受け

(注1): http://chasen.aist-nara.ac.jp/
(注2): http://cl.aist-nara.ac.jp/~taku-ku/software/cabocha/

PHS　良い点: メールを送受信した日付、時間が表示されるのも結構ありがたいです。
悪い点: なんとなく、レスポンスが悪いように思います。
mod　断定: 「ポケモン」の米国での成功を単純に喜んでいてはいけない。
意見: その論議を詰め、国民に青写真を示す時期ではないのか。
叙述: バブル崩壊で会社神話が崩れ、教育を取り巻く環境も変わった。

図 3　データの例

構造を導出した後, 単語 (原形) 単位の係り受け構造に変換する. 基本的に, 文節中の単語は直後の単語に, 文節中の最後の単語は係り先文節の主辞となる単語に係るものとする. また, 文頭, 文末にはダミーのノードを置く.

- N-グラム (**ngram**)

各単語 (原形) の係り先が, 直後の単語であるとみなした係り受け木である. 任意の部分木は, 単語 N-グラムになる.

これらのデータ, 文の表現方法を用い, 提案手法 (Boosting) と, Tree Kernel による SVM の比較実験を行った. Tree Kernel には, 文献 [8] で提案されている部分木のサイズに対する事前分布や, 親子間の伸縮等は考慮せず, 全部分木に展開するオリジナルの Tree Kernel を用いた. これは, 学習に使われる素性を同一にし, 比較を公平にするためである. 複数クラスの分類には, 2 値分類器を多値分類問題へ拡張する手法の 1 つである one-vs-rest を用いた. 評価は, 5-fold 交差検定により得られた F 値の平均にて行う. SVM のソフトマージンパラメータ, 及び Boosting の繰り返し回数は, それぞれ適当に変化させながら, テストデータに対する精度を最良にする値を採用した.(注3). 即ち, それぞれ最良かつ公平な条件での比較を行う.

5.2 実験結果

表 1,2 に, PHS,MOD の実験結果を示す. B は提案手法 (Boosting), S は SVM を用いた結果である.

表 1　結果 (PHS)
Table 1　Results (PHS)

	F 値
B/bow	76.6
B/dep	79.0
B/ngram	79.3
S/bow	77.2
S/dep	77.2
S/ngram	79.4

表 2　結果 (MOD)
Table 2　Results (MOD)

	断定	意見	叙述
B/bow	71.2	62.1	83.0
B/dep	87.5	80.5	91.9
B/ngram	87.6	78.4	91.9
S/bow	72.1	59.2	82.5
S/dep	81.7	26.1	88.1
S/ngram	81.7	26.1	88.1

5.3 考　　察

5.3.1 構造を考慮する有効性

全タスクにおいて, 構造を用いないベースライン (bow) に比べ, 構造を考慮する提案手法が顕著に精度が高いことが確認できる. 係り受け (dep) と N-グラム (ngram) を比較すると, MOD タスクの「意見」カテゴリにおいて, 係り受け構造を用いるほうが精度が良いが, 全般的に顕著な差は確認できない.

(注3): これらのパラメータは, 正則化の強さをコントロールするという意味で同種のものである

5.3.2 Tree Kernel を用いた SVM との比較

bow を素性とした場合, Boosting と SVM には顕著な差は確認されない. dep, ngram を用いた場合でも, 提案手法が同一かそれ以上の精度を示している. しかし, SVM の場合, カテゴリによっては著しく精度が悪い. このような顕著な精度低下の要因として, Tree Kernel をはじめとする Convolution Kernel 全体が持つ欠点が挙げられる. 言語データといった疎データに Convolution Kernel を用いると, 素性数が指数的に増えるため, 自分以外との内積と比較して, 自分自身との内積が極めて大きくなる傾向にある. これにより, 学習事例に酷似した事例のみを分類し, 残りはデフォルトクラスに分類するような丸暗記学習が行われやすくなる. これを回避する方法として, 1) 大きな部分構造の重みを減衰させる [8], 2) 正規分布の分散と同形の平滑化を与える [6], 3) カイ 2 乗値により素性選択する [12], といった手法が提案されている. これらの手法を用いることで, 提案手法より高い精度を得る可能性がある. しかし, これらの手法の目的は, あくまでも精度向上であり, 提案手法の持つ利点 「高速分類」, 「解釈のしやすさ」の実現ではない. また, 学習の素性空間が大きく変わってしまうため, 素性空間が同一という条件での公平な比較ができない. さらに, 減衰率といった設定すべきパラメータの数が増えるという意味で問題が複雑になってしまう.

5.3.3 提案手法の利点

4 章にて, 提案手法の「現実的な」利点について述べた. ここでは, 実験結果 (PHS) から, それらの利点を検証する.

提案手法は, 必要最小限の素性を自動的に選択する能力を備える. PHS タスクにおいて, 実際に使われた素性 (サポート素性) の数は 1,793 であり, 人手による分析に耐えうるサイズであった. 学習データから抽出した全 1-gram, 2-gram, 3-gram の異なり数がそれぞれ 4,211, 24,206, 43,658 であることから, いかに少数の素性で分類を行っているかが分かる. もし, SVM のサポートベクターの集合から, サポート素性を列挙すると, 数十万から数百万の素性数になると予想される.

サポート素性の分析の例として, 図 4 に, 個々のサポート素性 $t \in \mathcal{G}$ と, Boosting が算出した素性重み λ_t の一部を示す.

(A)「(〜し) にくい」を含む素性

「(〜し) にくい」は, 一般に否定的な意味の形容詞であり, 多くの素性は負 (悪い点) の重みが与えらえている. しかし, 「切れにくい」のみ正の重みとなり, ドメイン知識 (PHS) をよく反映している.

(B)「使う」を含む素性

「使う」は, 評判には中立な意味の単語だが, 周辺のコンテキストで重みが変化している (「使いたい」

→正,「使いやすい」→正). さらに, 過去形 (「使いやすかった」) になったり, 他の要素との比較 (「(〜の/〜する) 方が使いやすい」) になると, 負の重みが与えられており, 興味深い.

(C)「充電」を含む素性

ドメイン知識を反映した素性 (「充電時間が短い」→正,「充電時間が長い」→負) が抽出されている. また, これらは係り受け構造を考慮した素性である(注4).

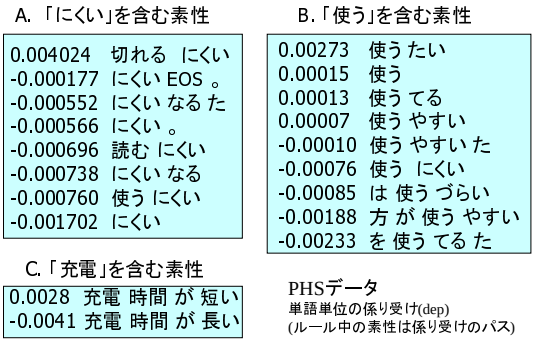

図 4 サポート素性の一部

さらに, 図 5 に分類の実行例を示す. 入力文「液晶が大きくて, 綺麗, 見やすい」に対し, どういう素性が適用されたか陽に出力し, 分析を容易にしている. このような分析は Tree Kernel では困難である.

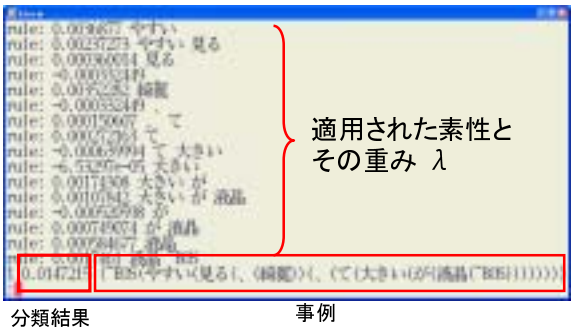

図 5 分類の実行例

(注: 前順序の S 式で木を表現しているため, 語順が逆になる)

最後に, 分類速度であるが, 提案手法の分類速度が 0.135 秒/1,149 事例に対し, SVM は, 57.91 秒/1,149 事例 であった(注5). 提案手法がおよそ 400 倍高速であることが確認された.

6. おわりに

本稿では, テキストの構造 (構文構造やレイアウト構造) を考慮したテキスト分類 (半構造化テキスト分類) に向け, 部分木を素性とする Decision Stumps と, それらを弱学習器として用いる Boosting アルゴリズムを提案した. さらに, 部分木列挙アルゴリズムを拡張し, 弱学習器の効率良い学習/分類手法を提案した. 実データを用いた実験にて, 文の構造を考慮する本手法の有効性を示した. さらに, 提案手法の 2 つの利点 (解釈のしやすさ, 高速分類) を実タスクにおいて確認した.

(注4):「充電時間がとても短い」は, 係り受けパターンならマッチできるが, N-グラムだとできない.

(注5): XEON 2.4Ghz, 主記憶 4.0Gbyte の Linux 上で測定した

謝 辞

文のモダリティ判別データを NTT コミュニケーション科学基礎研究所の平尾努氏に提供していただきました. ここに感謝いたします.

文 献

[1] Kenji Abe, Shinji Kawasoe, Tatsuya Asai, Hiroki Arimura, and Setsuo Arikawa. Optimized substructure discovery for semi-structured data. In *Proc. 6th European Conference on PKDD*, 2002.

[2] Roberto J. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*. ACM Press, 1998.

[3] Leo Breiman. Prediction games and arching algoritms. *Neural Computation*, 11(7):1493 – 1518, 1999.

[4] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Proc. of Neural Information Processing Systems (NIPS)*, 2001.

[5] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sicences*, 55(1):119–139, 1996.

[6] David Haussler. Convolution kernels on discrete structures. Technical report, UC Santa Cruz (UCS-CRL-99-10), 1999.

[7] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the ECML-98*, pages 137–142, 1998.

[8] Hisashi Kashima and Teruo Koyanagi. Svm kernels for semi-structured data. In *Proceedings of the ICML-2002*, pages 291–298, 2002.

[9] Shinichi Morhishita. Computing optimal hypotheses efficiently for boosting. In *Progress in Discovery Science*, pages 471–481. Springer, 2002.

[10] Gunnar. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, Department of Computer Science, University of Potsdam, 2001.

[11] Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[12] Jun Suzuki, Tsutomu Hirao, Hideki Isozaki, and Eisaku Maeda. String kernel with feature selection function (in Japanese). In *IPSJ SIG Technical Reports 2003-FI-72/2003-NL-157*, 2003.

[13] Mohammed Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining KDD*, pages 71–80, 2002.

[14] 田村直良 and 和田啓二. セグメントの分割と統合による文章の構造解析. **自然言語処理**, 5(1), 1996.