2E1-4in

文書分類のためのフレーズパターンの生成

Phrase pattern generation for text classification

服部 一浩[*1] 横野 光[*2] 相澤 彰子[*2*1]
Kazuhiro Hattori Hikaru Yokono Akiko Aizawa

[*1]東京大学 The University of Tokyo
[*2]国立情報科学研究所 National Institute of Informatics

Phrase-based patterns are recognized as useful features in recent studies in text categorization and sentiment analysis. In this paper, we propose a new method to generate phrase patterns by extending a polynomial time conventionally studied in regular patterns in formal language. The paper also reports results of extrinsic evaluation where the extracted phrase patterns are used as features in a traditional text categorization task. It is shown that higher micro F_1 score is obtained by the pattern-based features comparing with n-gram-based features.

1. はじめに

文章中の数箇所をワイルドカードで置き換えることによって表現されるいわゆる文章テンプレートは, 自然言語生成の分野で使われ, また, ワイルドカードに入る語を見ることで関係抽出にも使われる. 次々に新しいテンプレートを増やす必要がある場合, 人手でテンプレートを作る作業は高コストであり自動的にテンプレートを抽出する必要がある.

我々は有用なテンプレートとは次の 2 つの性質を満たすものであると考えた. 1 つはワイルドカードを含む割合が適切であること. テンプレートの利用によっては全くワイルドカードを含まないものや, 逆に大部分をワイルドカードが占めているものはテンプレートとして不適切だと思われる. もう 1 つは, テンプレートがそのコーパスに特徴的な語や文法表現, 言い回しを含んでいること, 例えば, 自然言語生成ならば生成する文章が科学に関するものか人文に関するものかによって異なる様態のテンプレートがあると有用である. 関係抽出の為のテンプレートならば, 対象が新聞記事であるか, ブログ記事であるかによってその様態は異なるべきである. 特にこの 2 点目を考えると, テンプレートは文章を特徴づけるものと言え, 文書分類の素性になり得ると考えられる. 文書分類では通常, 文書を特徴付ける素性として bag-of-words や n-gram が用いられるが, それらよりも高度な素性を目指すものとしてフレーズパターンが提案され, 文書分類を始め極性判定など様々なタスクへの応用が示されてきた [1, 2, 3, 4] . これらは文書分類等を目的に構成されたものであるが, 語の順序を保ったまま表現され, ワイルドカードに相当するものを持つなど, 文章テンプレートとの類似性が見られる. ただしテンプレートは文章に対応し, フレーズパターンは文またはそれより短いフレーズに対応する.

そこで我々は 1. コーパスからパターンの抽出 2. パターンの組み合わせによるテンプレートの作成 という 2 段階を踏むことでフレーズパターンを元に文章テンプレートを作れるのではないかと考えた. 本稿ではこの 1 段階目の為の手法を提案し, 抽出されたパターンがコーパスをどれだけ特徴づけているかを文書分類タスクによって実験し, bag-of-words 素性, n-gram 素性, それぞれを用いた場合との比較を行う.

フレーズパターンを表現する為の形式として, Angluin が導入した正規パターン (regular pattern) [5] を用いることにした. これは, 正規パターンは語とワイルドカードからなる列で表現され, テンプレートへの転用が容易であると考えられるからである.

連絡先: khattori@nii.ac.jp (服部)

2. 関連研究

文章の特徴を捉えるための素性として bag-of-words は単語を独立に扱うが, 語順の情報を持たせた素性として様々なフレーズパターンが使われてきた.

Fei らは文章の極性判定にフレーズパターンを用いる手法を提案した [1]. 彼らの作成するパターンは, 単語にタグを振り, タグの列として表現するものであった. タグは極性判定の為の単語の意味クラスとして構成されており, パターンは n-gram と同様に連続した列でありワイルドカードを持たない.

Zhang らは文書分類にフレーズパターンを用いる手法を提案した [2, 3]. 彼らのフレーズパターンは PrefixSpan アルゴリズムを用いて作成され, 語の列によって表現される. 文中にパターンの語が順に出現する場合にパターンがマッチしていると見做す. 即ちワイルドカードを明示的に持たず, 代わりに語と語の間に常に 0 語以上にマッチするワイルドカードを持つものと考えられる.

Tsur らはフレーズパターンを用いて皮肉文の検出を行った [4]. 彼らのパターンは n-gram のいくつかの単語を任意の 1 単語とマッチするワイルドカードに置き換えたもので表現され, n-gram と同様に長さ n のパターンは文中の連続した n 語と部分マッチするものを考える.

本稿で提案するパターンは, 文章のテンプレートとしての利用を目指す為に, ワイルドカードを有し, パターンの長さ以上の長さの文にマッチするようなものが望ましい. 自然言語では例えば形容詞の有無などの為に長さが可変であるからである. この点において, Zhang らのパターンが最も近いが, 彼らのパターンでは連続する語が表現できず構造が不足している.

そこで本稿では Angluin によって導入された形式言語の一つである消去不能正規パターン [5] を用いることにした. これは Gold によって提示された枠組み [6] である正提示から帰納的推論可能な言語クラスに属する. その推論の戦略として, 極小言語を求める多項式時間アルゴリズムが Shinohara によって示されている [7]. Arimura らはさらに正規パターンを拡張した k-multiple を導入し, この極小言語を求める多項式時間アルゴリズムを示した [8]. 提案手法は Arimura らによる推論アルゴリズムを元にして, フレーズパターンを生成する.

3. 正規パターン言語と k-mmg

この章ではまず Angluin によるパターンおよび正規パターンとそれらの言語の定義 [5] を述べ, 次に Arimura らによるその拡張である k-multiple[8] の定義を述べる.

3.1 正規パターン

集合 A の要素からなる長さ 1 以上の列の集合を A^+ と表記する. シンボルの有限集合 Σ (ただし $|\Sigma| \geq 2$) と変数の無限集合 X (ただし $\Sigma \cap X = \emptyset$) とがあるとき, $(\Sigma \cup X)^+$ の要素をパターンという. 更に, パターン中に出現する変数がすべて異なるものを, 正規パターンという. 以下, 本稿で扱うパターンはすべて正規パターンであるとし, 変数をすべて $\diamond$ で表記することにする. パターン中に出現するある 1 つの変数をパターンで置き換える操作を代入という. 代入によって変数を空列で置換する操作を許すパターンを消去可能パターンというが, 本稿ではこれを許さないとする (消去不能パターン). パターンは長さ 1 以上の列であり, 空列はパターンではない. パターン p から代入を繰り返すことでパターン q が得られるとき, p は q の汎化であるといい, $q \preceq p$ と表記する. また p は q よりも general であるといい, 逆に q は p よりも specific であるという. 汎化関係 $\preceq$ はパターン上の半順序で, 推移律が成り立つ. $q \preceq p$ かつ $p \preceq q$ のとき, 同値関係 $p \equiv q$ にあるとして p と q を同一視する. $q \preceq p$ かつ $q \not\equiv p$ のとき, $q \prec p$ と書く.

Σ^+ の要素を文字列と呼ぶ. $\Sigma^+ \subset (\Sigma \cup X)^+$ である為, 文字列も正規パターンである. また, 文字列は $\preceq$ の極小元である.

パターン p について $L(p) = \{s \in \Sigma^+ : s \preceq p\}$ を p が生成する言語と定義し, パターン言語と呼ぶ. 特に正規パターンが生成する言語を正規パターン言語と呼ぶ. 言語の定義から $q \preceq p \implies L(q) \subseteq L(p)$ が成り立ち, 特に $|\Sigma| \geq 3$ のとき, $q \preceq p \iff L(q) \subseteq L(p)$ が成り立つ [5].

3.2 極小言語

与えられた文字列の集合 S とパターン p について $S \subseteq L(p)$ が成り立つとき, パターン p は S を被覆するという. S を被覆するパターンが生成する言語の内、包含関係において極小であるような言語を極小言語といい, 極小言語を生成するパターンを極小汎化という. $p \preceq q \iff L(p) \subseteq L(q)$ が成り立つとき, パターンが極小であることと, 生成される言語が極小であることは同値である.

3.3 k-multiple

ここでは正規パターンの拡張として Arimura らによって導入された k-multiple[8] を説明する.

パターンの集合であってその大きさが 1 以上 k 以下であるようなものを k-multiple という (k は自然数). k-multiple 上の半順序 $\sqsubseteq$ を次で定める.

$$P \sqsubseteq Q \iff \forall p\, \exists q.\, p \preceq q$$

$P \sqsubseteq Q$ かつ $Q \sqsubseteq P$ のとき, P と Q とは同値関係 $P \equiv Q$ にあるとし, $P \sqsubseteq Q$ かつ $P \not\equiv Q$ のとき, $P \sqsubset Q$ と書く. k-multiple P の言語 $L(P)$ を

$$L(P) = \bigcup_{p \in P} L(p)$$

で定める. 一般に $P \sqsubseteq Q \implies L(P) \subseteq L(Q)$ が成り立ち, 特に $|\Sigma| \geq k+2$ のとき, $P \sqsubseteq Q \iff L(P) \subseteq L(Q)$ が成り立つ [9].

3.4 k-mmg

3.2 章で述べた極小言語を k-multiple に対しても全く同様に定義できる.

文字列集合 S について, $S \subseteq L(P)$ かつ $S \subseteq L(Q) \implies Q \not\sqsubset P$ であるような k-multiple P を k-minimal multiple generalization (k-mmg) という. 文字列集合 S が与えられた時に k-mmg を S の大きさと k に関して多項式時間で求めるアルゴリズム `MMG` [8] が Arimura らによって示されている.

4. `MMG` アルゴリズムを用いたパターン生成

提案するパターンの生成には Arimura らによって示された `MMG` アルゴリズムを利用する. `MMG` アルゴリズムとは文字列の集合 S と自然数 k について, S を被覆する極小言語を表現する高々k 個の正規パターン, すなわち k-mmg を計算するものである. アルゴリズムの概要は次の通りである. S を被覆する h-mmg を $h = 1$ から初め, h が k 未満である間, 正規パターンをより specific な複数の正規パターンに分割 (`division`) することで, 徐々に h を増やしていき, 最終的に k-mmg を得る. k-mmg は必ずしもちょうど k 個の正規パターンを含む必要はない.

本稿で目標に掲げたようなコーパスに特有なパターンの生成に対して, 極小言語を計算する `MMG` アルゴリズムは次の理由から適している. コーパスが十分に与えられていれば, ちょうどコーパスだけを表現するようなパターンが生成されるべきであり, 逆にコーパスに含まれない表現まで過剰に表現されてはいけない. 実際には限られたコーパスからパターンを生成する場合, コーパス以外の文字列を表現することを許すべきであり, 生成される言語が小さすぎる現象を過汎化という. 即ち, 生成される言語はコーパスを包含するが, パターンの汎化の度合いを制限した上で, 生成される言語が小さくなるようなパターンを考える必要がある. k-mmg ではパターンの個数の上限 k が生成されるパターンの汎化の度合いに作用し, k が大きくなるほど, 得られるパターンは specific になる. 極端な例として文字列集合 S に対して $k \geq |S|$ とすると, S の k-mmg は S そのものとなる.

自然言語処理に `MMG` アルゴリズムを適用する際に, 文字列集合 S には文の集合や, 文節などフレーズ単位に区切って得た集合など考えられる. より短い文字列から作る場合, 集合のバリエーションが増えるためパターンが豊富となるが, 当然得られるパターンも短くなるため, 本来のテンプレートへの転用という目標から離れてしまう. 今回はコーパスの文章を文節に区切ることで得た文節の集合を S とし, シンボル集合 Σ を S が含む単語の集合とした. また Σ の大きさは十分に大きいと仮定する.

しかしながら, `MMG` アルゴリズムをそのまま用いるのには難点が 2 つある. 1 つは, `MMG` アルゴリズムは得られるパターンが有する変数の割合を調整することが出来ず, 変数が大半を占めるようなパターンや, 逆に全く変数を含まない単なる文字列なども得られてしまう点である. 例えば, 文字列集合 S について $s \in S$ のみ他には使われていない文字で構成された文字列であるとすると $(S \setminus \{s\}) \sqsubseteq \{p\}$ とあるような p が存在する可能性があり, 2-mmg として $\{s, p\}$ を計算過程で経由する場合, `MMG` は出力に変数を全く含まないパターン s を含む. このように文字列としてただ一つ他と異なる文字列を被覆していたパターンは, 一回の分割で文字列になる. 我々の予備実験ではこの現象のため出力の大半を文字列が占め, S を適切に選ぶ必要が生じた. またもう 1 つの難点として, S に対する k の設定

が容易ではないことがある．一般に k は小さいほど出力されるパターンは general になり，大きくなるにつれて specific になる．この 2 点に対する解決として，次のような変更を施した．

まず，$k \geq |S|$ と設定する．この場合，本来の MMG アルゴリズムの出力は $|S|$ 自体になるが，その過程においては h-mmg $(1 \leq h \leq k)$ が $h = 1$ から順次，計算される．h が大きくになるにつれて含まれるパターンは specific になってゆく．この過程で得られた $P = \bigcup_{1 \leq h \leq k} h$-mmg の内，変数の割合に閾値を設けて，適当なものだけを選択する．0 以上 1 以下の 2 つの実数 ρ^-, ρ^+ を用いて，パターン p が持つ変数の割合 $\mathrm{vr}(p)$ = 変数の数 / パターンの長さ を算出し，$P' = \{p \in P : \rho^- \leq \mathrm{vr}(p) < \rho^+\}$ とする．P' は，もはや S 全体を被覆していない可能性がある．P' が生成する言語の小ささは ρ^-, ρ^+ によって決定される．このようにして生成されたパターンの集合 P' をアルゴリズムの出力とし，mmg パターンと呼ぶことにする．

元の MMG アルゴリズムは多項式時間で計算可能であり，従って mmg パターンの計算も多項式時間である．実用上 $|S|$ の上限を 1000 と設けた．実際には，正例の文節からランダムに定数個選び取ってくることによって文字列集合 S とした．

更なるアルゴリズムの変更として，h-mmg からパターンを取り出す際に，複数連続した変数を一重に簡単化する処理 (var_simplify) を行う．例えば $\Sigma = \{0, 1\}$ のとき，文字列 001 にパターン $0 \diamond \diamond 1$ はマッチしないが var_simplify はこのパターンを $0 \diamond 1$ に写し，マッチを許すようになる．この処理は一般には生成する言語を大きくし，従ってパターンのマッチする頻度を増やす．語の順序や連続性を崩すことはしないのでパターンが表現する自然言語文のトピックまたは属するドメインが変わるほどの操作でないと考えられる．

Algorithm 1 は以上を擬似コードとして示したものである．tightest, division, divisible は，Arimura らの提案した MMG アルゴリズム [8] で $k \geq |S|$ と設定したものである．tightest は正規パターンと被覆する文字列集合のペア (p, C) を受け取って，C を被覆する条件の下で p から貪欲に極小な正規パターンを発見して返す手続きである．division は正規パターンを二つ以上の正規パターンに分割する手続きで，正規パターンとそれが被覆する文字列集合のペア (p, C) を受け取ってペアの集合 $\{(p_i, C_i) : i = 1, 2, \ldots, m,\ p_i \prec p,\ L(p_i) \cap C = C_i \subset C\}$ を返す．ただし m は 2 以上の自然数で，かつ $\bigcup_i C_i = C$ を要請する．このような集合は必ずしも存在しない．divisible は division の返す値が存在するかどうかを判定する述語である．

Algorithm 1 MMG

Require: set of strings, S
  $P \leftarrow \emptyset$
  $p \leftarrow \mathrm{tightest}(\diamond, S)$
  $cp \leftarrow \{(p, S)\}$
  while $cp \neq \emptyset$ **do**
    $(p, C) \in cp$
    $cp \leftarrow cp \setminus \{(p, C)\}$
    $p' \leftarrow \mathrm{var\_simplify}(p)$
    if $\rho^- \leq \mathrm{vr}(p') < \rho^+$ **then**
      $P \leftarrow P \cup \{p'\}$
    end if
    if p is divisible **then**
      $cp \leftarrow cp \cup \mathrm{division}(p, C)$
    end if
  end while
  return P

5. 評価実験

5.1 コーパス

コーパスには Reuters-21578 の ApteMod を用いた．一つの文章には 135 個のトピックの内から 0 個以上のトピックが振られている．8762 の訓練事例と，3009 のテスト事例から構成されており，ともに含まれるカテゴリは 89 個であり，これを用いた．このコーパスの特徴として，正例が極端に小さいことがあり，76 のカテゴリの正例は 100 未満，30 のカテゴリの正例は 10 未満であった．

5.2 得られたパターン

トピック acq の正例事例に含まれる 1488 の文節から次のパターンが得られた．

1. ◇ `acquisition` ◇
2. ◇ `said it` ◇ `to acquire` ◇
3. ◇ `sales` ◇ `mln dlrs`
4. ◇ `of` ◇ `outstanding` ◇
5. ◇ `said` ◇ `agreement` ◇ `of` ◇

これらは，文節に対して完全マッチするパターンである．パターン 1 は文中に “acquisition” が出現する文にマッチする．ただし，ワイルドカードには 1 語以上が代入される制約のため，文節の頭か末のみに “acquisition” が出現する文節にはマッチしない．トピック acq はまさしく Acquisition に関する文章に振られるタグである為パターン 1 は トピック acq を特徴付けていることが期待できる．パターン 2 は加えて周辺の表現までパターンとして抽出できた例である．パターン 3 は acq と earn のトピックに集中してマッチし，テスト事例の内，パターン 3 がマッチした文節は 41 で、内 26 が acq, 14 が earn であった．パターン 4,5 は一見特徴語を含まないが，テスト事例中において，パターン 4 のマッチした文節の数は 73, 内 acq は 59, パターン 5 のマッチした文節の数は 28, 内 acq は 23 となっており，トピックを特徴づけるパターンであると言える．

5.3 評価指標

パターンそのものを評価するために，我々は良いパターンとは文をより特徴付けるものだと考え，テキスト分類のタスクによって，得られたパターンの評価を行った．

カテゴリ毎に SVM を用いて二値分類器を作り，精度，再現率，F_1 スコアを計算する．全てのカテゴリでの結果から micro F_1 スコアと macro F_1 スコア [10] とを計算してこの二つで比較する．一般に，macro F_1 は正例の小さいカテゴリでの結果に寄り，micro F_1 は正例の大きなカテゴリに寄る傾向にある．特に Reuters-21578 は先述したように，ほとんどのカテゴリの正例が小さいために，macro F_1 は micro F_1 より小さな値になる．

5.4 分類手法

今回提案する mmg パターンを素性とする分類器の他に，比較対象として，テキスト分類に一般的とされている bag-of-words 素性及び n-gram 素性を用いた分類器を作り結果を比較する．分類器はサポートベクターマシンのパッケージである SVM には SVM^{light} [11] を用い，次元数 2 の多項式カーネルをカーネルとした．mmg パターンの生成において $rho^- = 0.3, rho^+ = 0.8$ とした．bag-of-words 素性は stop words を取り除き，訓練事例中に 3 回以上出現した語のみを選択して用いる．n-gram 素性

には, uni-gram, bi-gram, tri-gram を用いる. またどの素性でもカイ二乗値によって素性選択を行った. 過去の bag-of-words 素性と SVM による実験 [12] では上位 10,000 の素性だけを用いるのが最良だと報告されている. 今回の実験でもすべて上位 10,000 だけを用いることとした.

5.5 結果

表 1 に, それぞれ bag-of-words 素性を用いた分類器, n-gram 素性を用いた分類器, `MMG` で得られたパターン (mmg パターン) を用いた分類結果の macro F_1 スコアと micro F_1 スコアとを示す.

mmg パターンによる分類は bag-of-words 素性を用いた分類には届かなかったが, micro F_1 は, n-gram 素性を用いた分類よりは良い結果が得られた. macro F_1 は他よりも悪い結果であるが, これは正例の小さなカテゴリの結果が他よりも悪いことを意味している. 図 1 は, このことについて詳細な分析を与えるもので, 各カテゴリについての二値分類器の性能をプロットしたものである. 横軸はカテゴリが持つ正例の数で, 縦軸は分類結果の F_1 スコアである. 正例の大きなカテゴリについては, mmg パターンは他と同様に分類できている傾向が見られるが, 正例の小さなカテゴリについては上手く分類が出来てないことが分かる.

この理由は次のように説明できる. mmg パターンとして得るためには, 文字列集合 S 中に文字列的に類似したものが複数存在しなければならない. 正例が大きい場合にはそのようなものが多くあるために, 豊富に mmg パターンが得られるが, 逆に正例が小さいものについては不利になる傾向がある.

	bag-of-words	n-gram	mmg
maF_1	47.28 %	45.28 %	34.72 %
miF_1	84.80 %	78.00 %	80.20 %

Table 1: 素性に対する SVM による分類の F_1 スコア

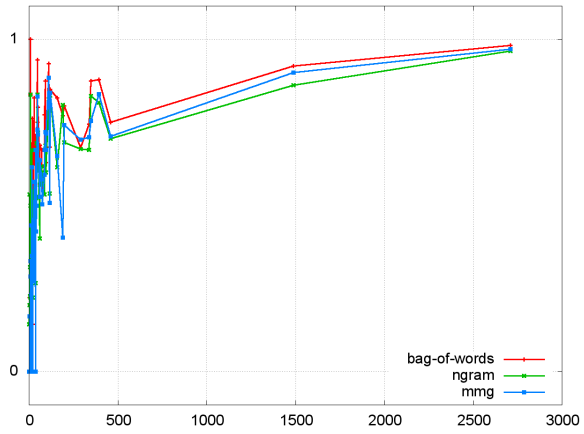

Figure 1: カテゴリの頻度に対する性能

6. おわりに

本稿では文章テンプレートの作成の足がかりとしてフレーズパターンを正規パターンで表現し, コーパスから生成する方法について述べた. 抽出したパターンを組み合わせることでテンプレートを生成し, 実際に自然言語生成のタスクに適用する工程は今後の研究としたい.

また, 抽出されたフレーズパターンが実際にコーパスのドメインをよく特徴づけていることが実験で明らかになったので既存研究と同様にフレーズパターン素性として用いた他のタスクへの適用も考えられる.

References

[1] Z. Fei, J. Liu, and G. Wu: Sentiment Classification Using Phrase Patterns in *CIT, 2004*, pp. 1147–1152

[2] B. Zhang, B. Hutchinson, W. Wu, and M. Ostendorf: Extracting Phrase Patterns with Minimum Redundancy for Unsupervised Speaker Role Classification in *Proc. HLT, 2010*, pp. 717–720

[3] B. Zhang, A. Marin, B. Hutchinson, and M. Ostendorf: Learning Phrase Patterns for Text Classification in *IEEE transactions on audio, speech, and language processing, 2013*, pp. 1180–1189

[4] O. Tsur, D. Davidov: ICWSM – A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Product Reviews in *Proc. AAAI, 2010*

[5] D. Angluin: Finding Patterns Common to a Set of Strings in *Journal of Computer and System Sciences, 1980*, pp. 46–62

[6] E. Gold: Language Identification in the Limit in *Information and Control, 1967*, pp. 447–474

[7] T. Shinohara: Polynomial Time Inference of Extended Regular Pattern Languages in *RIMS Symposia on Software Science and Engineering LNCS, Volume 147, 1983*, pp. 115–127

[8] H. Arimura, T. Shinohara, and, S. Otsuki: Finding Minimal Generalizations for Unions of Pattern Languages and its Application to Inductive Inference from Positive Data in *Proc. STACS '94, LNCS, Vol. 775 Springer, Berlin (1994)*, pp. 649–660

[9] J. Uemura, M. Sato: Compactness and Learning of Classes of Unions of Erasing Regular Pattern Languages in *LNCS, Volume 2533, 2002*, pp. 293–307

[10] D. Lewis, R. Schapire, J. Callan, R. Papka: Training Algorithms for Linear Text Classifiers in *SIGIR, 1996* pp. 298–306

[11] T. Joachims: Making large-Scale SVM Learning Practical in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[12] Y. Yang, and X. Liu: A Re-examination of Text Categorization Methods in *SIGIR, 1999*, pp. 42–49