Wikipediaからの要約生成パターンの抽出

田中 翔平 † 岡崎 直観 ‡ 石塚 満 ††

† 東京大学工学部電子情報工学科 ‡ 東京大学大学院情報学環 †† 東京大学大学院情報理工学系研究科

1 はじめに

オンライン百科辞典として有名なWikipedia[1]は，様々な事柄に関する解説を蓄積している．現在の英語版Wikipediaには，約300万件の膨大な記事があるが，全ての事柄を網羅しているわけではない．更に，日々増え続ける新しい情報を手作業で記事にまとめるのは，大変な労力が要る．そこで，本研究は，Web上の情報に基づいて，Wikipediaを模した記事の編纂作業を自動化することを目標とする．

2 手法

2.1 手法の概要

本研究では，記事の自動生成の対象となる事柄のことを，「エンティティ」と呼ぶ．このエンティティにはそれぞれ，記事に収録されやすい関連情報が存在する．例えば，ある俳優に関する記事を作成するときは，その俳優の出身地や出演作品などの情報をカバーしていることが望ましい．このような，あるエンティティに関して収録すべき周辺情報を，本研究では「fact」と呼ぶ．これらを踏まえると，あるエンティティに関する記事を自動的に作成するには，Webからエンティティに関するfactをできるだけ多く，かつ効率よく収集する必要があると言える．この時，利用する検索クエリの素となるいくつかのフレーズを，「テンプレート」と呼ぶ．このテンプレートが求まれば，記事作成に必要な情報をテンプレートに従ってWeb上から収集し，既存の要約手法などを用いて記事を作り出すことができる．

テンプレートは，エンティティの周辺情報として必要なfactを多く収集でき，かつ，検索クエリの数をできるだけ少なくするものが望ましい．そこで，本稿では，このテンプレートを最適化する手法を中心に述べる．テンプレートを最適化するためには，Web検索においてクエリとしてどのフレーズを用いるかを最適化しなければならず，これをWeighted Max-SAT問題として定式化して解く手法についてを説明する．

2.2 記事作成の手順

作りたい記事となるエンティティを，Wikipediaの既存の記事におけるタイトルと置き換えると，タイトルに関連するfactは記事内のWikilinkと考える事ができる．更に，factがどのような種類の内容を記述したものか，すなわちタイトルとどのような関係のfactであるかは，タイトルとWikilinkの間の単語の並び(パターン)で知ることができる．このタイトルとfactの間の関係の種類は，記事の分野(カテゴリ)毎にある程度限定されるため，それを記述したものである単語パターンは，カテゴリ毎に偏りがあると考えられる．そこで，既存のWikipediaの記事からテンプレートを作成し，それを利用して記事を作成したいエンティティに関するWebソースの文章を新たに作成するために，次のようなシステムを考える．

1. 入力として記事にしたいエンティティの名前と，そのエンティティが属すべきカテゴリを与える．
2. 与えられたカテゴリに属する既存のWikipedia記事をリストアップし，それぞれの記事の文章の中から，各記事のタイトルとWikilinkの間にある単語パターンを抽出する．こうして抽出された単語パターンの中から，テンプレートとして利用できそうなものを探す．
3. 選んだ単語パターンそれぞれに対して，検索エンジンを利用して“(エンティティ名) (パターン)”という検索を行い，文書をいくつか得る．
4. 得た文書を既存の手法により要約し，一つの文章を作成する．

さて，このシステムの性能を考える上で一番問題となるのが，テンプレートとして利用するパターンの選定である．パターンは網羅的にfactを収集することが望ましいが，Web検索におけるコストを考えると，できるだけパターンの数を少なくしたい．更に，収集したいfactがほとんど述べられていない文書や，エンティティと関係の無い記述が多い文書を収集してしまうパターンは，テンプレートとして相応しくない．これら全てを考慮してテンプレートを作成するために，次に提案する手法によりパターン選定を行う．

2.3 パターン選定

パターン選定の手順は以下のような流れになる．

1. 入力として与えられたカテゴリに属する全てのWikipedia記事に対し，前述のパターン抽出を行い，存在する記事のタイトルのリストと，各記事内に出現するWikilink(各記事のタイトルに関連するfact)のリストを得る．
2. 抽出した全てのタイトル，パターンに対し，Web検索エンジンを用いて“(タイトル) (パターン)”という検索を行い，検索結果上位10件の文書を得る．文書としては，元のWikipedia記事は採用しないものとする．
3. 得られた文書を解析し，どのfactが出現したか(どのfactを収集できるか)を調べる．また，文書内において”(タイトル) (パターン)”というフレーズが出てきたとき，その後にタイトルに関連するfactが出現したかどうかを調べる．これらの解析により，どのパターンからどのfactが収集でき，またどの程度正しくfactを収集できたかを知る事ができる．
4. 各パターンのfact収集能力から，テンプレートとして採用するパターンを選ぶ．まず，選ぶパターンは，全体でfactをより多く収集する組み合わせでなければならない．その一方で，選ぶパターンはfactをより正確に収集できるものでなければならず，正確性に欠くパターンは，テンプレートとしては相応しくない．この2つの条件を満たすような定式化は，以下のようになる．

$$
\begin{aligned}
&\textit{Maximize} && \sum_i X_i - \sum_j \lambda_j P_j \\
&\textit{Subject to} && X_1 = P_{1a} \vee P_{1b} \vee P_{1c} \cdots \quad (weight = 1) \\
& && X_2 = P_{2a} \vee P_{2b} \vee P_{2c} \cdots \quad (weight = 1) \\
& && \cdots \\
& && X_n = P_{na} \vee P_{nb} \vee P_{nc} \cdots \quad (weight = 1) \\
& && P_1 \in \{0, 1\} \qquad (\lambda_1 = c_1 w_p + c_2 E_1) \\
& && \cdots \\
& && P_m \in \{0, 1\} \qquad (\lambda_m = c_1 w_p + c_2 E_m)
\end{aligned}
$$

Pattern extraction from Wikipedia for summarization

†Shouhei TANAKA, Faculty of Engineering, University of Tokyo

‡Naoaki OKAZAKI, Interfaculty Initiative in Information Studies, University of Tokyo

††Mitsuru ISHIZUKA, Graduate School of Information Science and Technology, University of Tokyo

[1] http://www.wikipedia.org

ここで，P_j は j 番目のパターンのことであり，テンプレートとしてそのパターンを採用するならば 1，しないならば 0 となる．また，X_i は i 番目の fact に関する節であり，Web 検索でその fact を収集できるパターンの論理和で表現される．例えば節 X_1 は，1 番目の fact が，1a 番目のパターン，1b 番目のパターン，1c 番目のパターン…から収集できる，という意味である．これらのパターンの中でどれか 1 つがテンプレートとして選ばれれば，1 番目の fact は最終的に収集できるという事になり，節 X_1 は 1 になる．λ_j は j 番目のパターンに関する重み付けを意味し，パターン全体に共通の重み付けと各パターンに固有の重みを c_1，c_2 で決まる割合で足したものである．各パターンに固有の重み付けとしては，fact 損失数 E_j(“(タイトル) (j 番目のパターン)”の後に適切な fact が来なかった数) を採用する．この定式化により，満たす fact の数から，選んだパターンの fact 損失数に関連する値を引いたものを最大化することになる．このため，重み付けを適切に決めれば，損失が少なく，かつより多くの fact を収集するパターンを選び出すことができる．なお，この問題は Weighted Max-SAT 問題として解く事ができる．

3 実験

提案手法の有用性を評価するために，実験を行う．実験では既存の Wikipedia 記事を利用し，複数のカテゴリに対してテンプレートを作成し，fact の収集能力を調べることでその性能を評価する．なお，実験においては検索エンジンとして Yahoo! Search BOSS[2]を利用し，SAT solver として SAT4J[3]を利用した．

3.1 内容

Tennis players，American actors，Software companies という 3 カテゴリに対して，交差検定によるテンプレートの生成，及び評価実験を行う．具体的な手順は以下のとおりである．

1. 指定されたカテゴリの記事をランダムに 10 分割する．
2. 9/10 の記事からパターンを抽出し，3 回以上出現したパターンの中から，提案手法または他のベースライン手法によりパターンを 6 個選ぶ．
3. 残り 1/10 の記事のタイトルと選んだパターンを組み合わせて，“(タイトル) (パターン)”Web 検索を行い，上位 10 件の文書を得る．選んだパターン全てについての結果から，その中に元の記事の fact がいくつ出現するか (Coverage)，“(タイトル) (パターン)” の後に元の記事内の fact がどの程度の割合で出現するか (precision) を調べる．これら 2 値から，擬似的な F 値

$$F' = \frac{2 \cdot Precision \cdot Coverage}{Precison + Coverage}$$

を計算する．

4. 10 個全ての分割に対して，2 と 3 の作業を行う．こうして得られた 10 回の評価の平均を，その手法の評価とする．

この実験では，提案手法の他に 3 つのベースラインを用意する．ベースラインの 1 つめは，それぞれのカテゴリの Wikipedia 記事群におけるパターンの出現回数の多い順に 6 個を取る方法で，もっとも単純な方法である．2 つめは，ランダムに 6 個を取る手法である．3 つめは，特定カテゴリに偏って出現するパターンを選出するために，各パターンのカテゴリ毎の出現回数から χ^2 値を計算し，その値が大きな順にパターンを選ぶ手法である．

[2]http://developer.yahoo.com/search/boss/
[3]http://www.sat4j.org

3.2 結果

各手法について，3 カテゴリの平均を取った結果を表 1 に示す．
なお，Weighted Max-SAT 問題では重みと係数を決めなければならないが，この実験では $c_1, w_p = 1$ とし，c_2 のみを変数としてパターンが 6 個選ばれるよう調整した．このように，係数の最適化を厳密には行っていないが，提案手法が出力するテンプレートが他のベースライン手法に比べて，より最適化ができており，fact 収集の性能が向上することが分かった．なお，テンプレートとして採用したパターンの具体例を挙げると，例えば Software Companies というカテゴリの場合，頻度の上位 6 件からテンプレートを作ると，「*“is a”, “was a”, “is an”, “acquired”, “is headquartered in”, “was acquired by”*」というパターンが選ばれる．それに対し提案手法では，「*“acquired”, “was acquired by”, “is”, “was founded by”, “also developed”, “is a vendor of”*」が得られる．この例を見ても分かるように，擬似的な F 値が向上するだけでなく，人間の目で見てもテンプレートとしてより最適なパターンが選ばれていることが分かる．

表 1: 3 カテゴリの平均

手法	Pre.	Cov.	F'
頻度	30.17	56.16	38.99
ランダム	39.97	43.09	37.53
χ^2 値	57.92	22.73	30.46
提案手法	46.94	53.49	49.48

4 関連研究

Wikipedia を要約例と見なし，それを元に Web から情報を収集して新しい記事を生成するという研究としては，C.Sauper[1] らの研究がある．この研究では，カテゴリ毎の節の名前 (セクション名) を抽出し，テンプレートとして利用している．このテンプレートをもとに Web 検索を行い，結果得られた複数の文章の中から適切なものを選ぶ．その過程を，線形計画問題として定式化し，これを解くことで最適化を行っている．この手法は文章選択という要約の部分を特に重視した方法であるが，検索クエリの妥当性は考慮されていない．

5 結論

本稿においては，Wikipedia の記事からテンプレートを学習し，それを元に新しい文章を作成するシステムと，そのシステム内でより良いテンプレートを生成するパターンを選定するための手法についてを述べた．また，作成したテンプレートについて評価実験を行い，それにより他のベースライン手法より良い結果を得られる事を示した．なお，提案手法では Weighted Max-SAT 問題を解いているが，これは多項式時間では解けない問題であり，現実的な時間では解けない可能性がある．ちなみに今回の評価実験では 1 カテゴリあたり約 2000 記事を用いているが，3.2GHz の CPU を搭載した PC で約 30 分で解けている．このように，現在対象としている記事数では現実的な時間で解けている．しかしながら，今後対象とする記事数が増えたり，パターンの候補の数が増えることも考えられるため，その場合計算時間がどうなるかという問題は今後の検討課題としたい．

参考文献

[1] Christina Sauper and Regina Barzilay Automatically Generating Wikipedia Articles:A Structure-Aware Approach In *Proc. of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 208-216, 2009.