

Wikipedia から興味ある情報を手早く検索できるサービス

市村 哲¹

概要：Wikipedia は世界中のボランティアによって共同執筆されている無料で有用なインターネット百科事典であるが、収録されている情報が非常に膨大であり、欲しい情報を見つけ出すことが難しいという問題がある。著者らは、多くのページを閲覧しなくても手軽に Wikipedia から興味ある情報を見つけ出せるようにすることを研究の目標に定め、Wiki ゲーター (Wikigator) を開発した。DBpedia 経由で取得した Wikipedia のデータセット文章を機械学習し、ユーザの好みを診断するための選択肢を自動作成し複数回に分けて提示する。ユーザがどの選択肢を選んだかに基づいて推薦すべきページを徐々に絞込むようになっていく。

Quick Wikipedia Search Based on User's Preference

Satoshi Ichimura¹

1. はじめに

Wikipedia[1] は無料で利用できるインターネット百科事典である。Wiki システムを採用しており、関連項目をリンクによって結びつけることや、Web ブラウザを用いて簡単に執筆・編集できることが特徴である。世界中のボランティアによって共同執筆されている。また Wikipedia の記事にはカテゴリが付与され、各カテゴリは階層構造化されている。収録された情報は Wikipedia サイトに設けられた検索機能によってテキスト検索できる他、Google 検索などの一般的な検索サイトを使用した際に検索結果として Wikipedia 内の情報が提示されることも多い。

しかしながら、Wikipedia に収録されている情報は非常に膨大であり、欲しい情報を見つけ出すことが難しいという問題がある。特に、知りたいことがまだ曖昧であるような場合は検索キーワードを正しく指定できないため、欲しい情報にたどり着くことができず、自分に興味があるページにたどり着くためには、多くのページをひとつひとつ閲覧する必要がある。例えば、旅行に興味はあるが、どこに行きたいか、何を見たいか、どのような遊びをしたいかがまだ決まっていない段階では、的確なキーワードを入力し

て旅行情報を検索することができず、使いにくい面がある。

以上の問題に着目し、多くのページを閲覧しなくても手軽に Wikipedia から興味ある情報を見つけ出せるようにすることを研究の目標に定め、この目標に沿って「Wiki ゲーター (Wikigator)」を開発した。Wiki ゲーターでは、DBpedia[2] 経由で取得した Wikipedia のデータセット文章を機械学習し、ユーザの好みを診断する幾つかの質問を自動的に作成する。ユーザは、質問に回答する際システムが提示した選択肢から好みのものを選ぶだけでよい。各質問にユーザがどのように回答したかに基づいてユーザの好みに適した情報ページが推薦されるようになっている。

2. 関連研究

Wikipedia に収録されている情報は非常に膨大であり、自分に興味があるページにたどり着くことが難しいという問題がある。同様な問題がインターネット上の他のサービスについても生じており、これらの問題に対する関連研究について以下に述べる。

大山ら [3] は、ゲームレビューサイトに登録されている大量のレビュー文章を Word2Vec[4] を用いて機械学習し、ユーザがキーワードを入力した際に、類似度が高いゲーム名を出力するシステムを提案している。Word2Vec は大量のテキストデータを解析して単語をベクトル化する手法である。単語をベクトル化することで、ベクトル間の距離を

¹ 大妻女子大学社会情報学部情報デザイン専攻
12 Banchi, Sanbancho, Chiyoda-ku, Tokyo, 102-8357 Japan

比較して、その単語に近い単語、すなわち類似した単語を出力することや、単語と単語の類似度を求めることができる。一般的に Word2Vec は中間層と出力層の 2 層からなるニューラルネットワークで構成されている。

栗原ら [5] は、映画レビューサイトに登録されている大量のレビュー文書を自然言語検索するために、Word2Vec と Doc2Vec[6][7] を併せて用いる手法を提案している。Word2Vec が入力として単語列を受け取るのに対し、Doc2Vec は単語列と文章 ID を併せて受け取るため、単語のベクトル化だけでなく任意の長さの文章のベクトル化が可能である。栗原らは映画ごとにレビュー文章をまとめて Doc2Vec に入力して機械学習すると共に、Word2Vec を用いて質問文に類語を追加してクエリ拡張している。これによりクエリ一文に記述した語句が入っていないような場合でも、意味的に近い口コミ文を検索できるようになったと報告している。Doc2Vec も入力層、中間層、出力層からなるニューラルネットワークで構成されている。

難波 [8] は、異なる特許の請求項の同一性を自動判定するために、BERT (Bidirectional Encoder Representations from Transformers) [9] を用いる方法を提案している。BERT は Doc2Vec と同じく任意の長さの文章のベクトル化が可能な手法であるが、Wikipedia などの大量の教師なし文章データから事前学習モデル (pre-trained models) を作成しておき、各種タスクに応じて少ない文章データを追加学習 (転移学習) するだけで精度良く文章分類タスクなどを実行できる利点を有している。主題と構成要素 (文章構造) の類似性を重視して特許請求項の同一性を判定する方法を提案している。

著者らは過去において、観光地に関する膨大な口コミデータを Doc2Vec を用いて機械学習しユーザの好みに応じた観光地を推薦する旅ゲーター [10] を開発した。ただし対象は事前に取得した観光地情報のみが推薦の対象となっていた。

膨大な情報から興味ある情報を選び出すことは一般的に容易なことではない。これに関し選択行動についての社会学的研究が存在する。

Iyengar(アイエンガー) [11] は、選択肢数と満足度の関係について複数の実験を行い、選択肢数の増加が選択結果の満足度を低下させ、結果として選択行動そのものを放棄させる場合があることを示している。スーパーマーケットのジャム試食コーナーにおいて、6 種類のジャムを陳列した場合と、24 種類のジャムを陳列した場合とを比較した実験では、6 種類の場合は 40 % の人が試食コーナーで立ち止まってその中の 30 % の人が購入した一方で、24 種類の場合は 60 % 人が立ち止まったもののその中の 3 % の人しか購入しなかった。結果としてジャムの購入件数は 6 種類の場合が、24 種類の場合の約 6 倍であったと報告している。

また大学生を対象にチョコレートの選択行動を観察した

実験では、30 種類から最も好みのものを選択した場合と、6 種類から最も好みのものを選択した場合とを比較した場合に、6 種類から選んだ方が、選択したチョコレートに対する満足度が高く、選択したチョコレートをより美味しいと感じた人が多かったと報告している。Iyengar は選択肢数の増加によって選択行動が放棄されたり、選択結果満足度が低下する現象を「選択のオーバーロード」と呼んでいる。そして、人が自信を持って選べ、選んだ結果について満足度が高いのは選択肢数が 5~9 (7 ± 2) の場合が多いと述べている。

加えて Schwartz(シュワルツ)[12] は、選択肢が多ければ多いほど不幸を感じやすくなってしまう人間の心理作用のことを「選択のパラドックス」と呼んでいる。選択肢が多いと、苦勞して多くの情報を比較して 1 つを選んだとしても、「他の選択もあったのでは」という迷いが大きくなってしまい、正しい選択をしたかどうか自信が持てなくなり満足感が低下する。そのため、多くの中から選択する度に不幸を感じてしまうと述べている。

3. 提案

多くのページを閲覧しなくても手軽に Wikipedia から興味ある情報を見つけ出せるようにすることを研究の目標に定め、この目標に沿い Wiki ゲーターを開発した。DBpedia 経由で取得した Wikipedia のデータセット文章を機械学習し、ユーザの好みを診断する幾つかの質問を自動的に作成する。ユーザは、質問に回答する際システムが提示した選択肢から好みのものを選ぶだけでよい。各質問にユーザがどのように回答したかに基づいてユーザの好みに適した情報ページが推薦されるようになっている。

具体的には、Wiki ゲーターの Web ページを表示すると選択肢 (ボタン) が表示される。ユーザは自分が好みの選択肢を選んでボタンを押す。すると、次の選択肢が現れるので、再度好みの選択肢を選んでボタンを押す。この操作を数回繰り返すと最後におすすめ情報が表示される。

次に典型的なユースケースについて図を用いて説明する。たとえば関東近郊の観光地について調べたいとする。そして手始めに、Wikipedia サイトに設けられた検索枠に自分が知っている観光地「江ノ島」を入力して検索したと仮定する。すると、Wikipedia 内の「江ノ島」(<https://ja.wikipedia.org/wiki/江ノ島>) がヒットするが、そのページの最下段には「江ノ島」が「神奈川県観光地」「神奈川県指定史跡」「日本百景」等のカテゴリに属していることが表示される (図 1)。

ユーザは Wikipedia に登録されているカテゴリを検索枠に入力することで Wiki ゲーターを利用することができる。ここではページの最下段に表示されたカテゴリの中から「神奈川県観光地」を選んで、Wiki ゲーターに入力したとする。すると Wiki ゲーターは「神奈川県観光地」

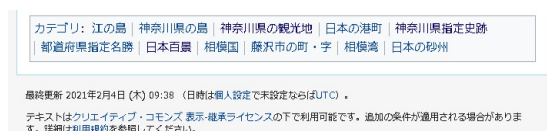


図 1 Wikipedia 記事が属するカテゴリ (江ノ島の例)

カテゴリに登録されている 40 件の Wikipedia ページの内容 (正確には, Wikipedia ページに対応した DBpedia ページの内容) を機械学習し, ユーザの好みを診断する複数選択肢を自動的作成してユーザに提示する。

ここでは Wiki ゲーターによって「海岸」「神奈川」「城」「碑」「公園」の 5 つの選択肢が表示されたと仮定する (図 2)。これらの中から, ユーザが「碑」を選んだとすると, Wiki ゲーターによって次の選択肢「翁」「石」「駅長」が表示される (図 3)。そしてユーザが「翁」を選んだ場合には「相翁松の碑」の Wikipedia ページがお勧めページとして推薦される (図 4)。ここでもし「石」を選んだ場合は「蛙石」ページが, 「駅長」を選んだ場合は「松本駅長殉難碑」ページが推薦される。

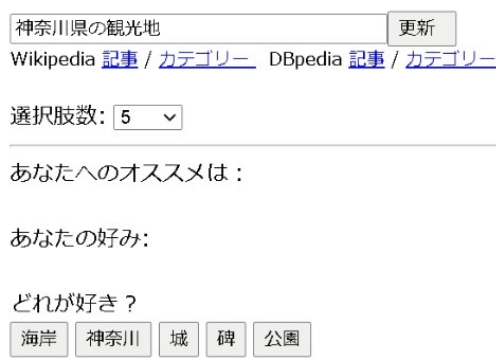


図 2 システム動作画面 (ステップ 1)

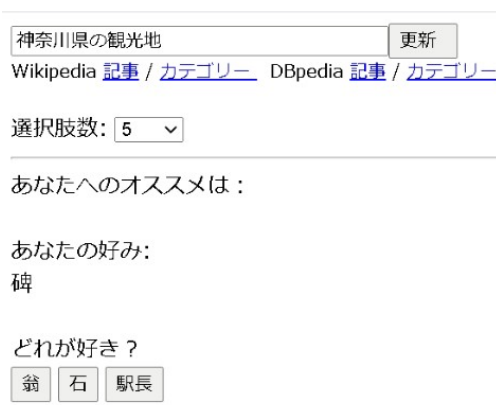
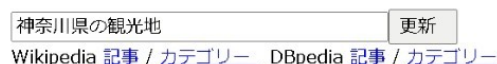


図 3 システム動作画面 (ステップ 2)

なお, Wikipedia ページ上で「神奈川県観光地」をクリックすると, このカテゴリに関する情報ページ (<https://ja.wikipedia.org/wiki/Category:神奈川県の観光地>)



選択肢数: 5

あなたへのオススメは:

相翁松の碑

相翁松の碑 (あいおいまつのひ) とは, 神奈川県小田原市江之浦にある石碑である。相翁松碑 (しょうおうしょうひ) とも表記する。1907 年 (明治 40 年) に, 坪野平太郎、阪谷芳郎、添田壽一らによって建立された。

あなたの好み:

碑 翁

図 4 システム動作画面 (ステップ 3)

地) が表示されるが, 上位カテゴリが「日本の観光地 (都道府県別)」であり, 下位カテゴリが「神奈川県の温泉」「神奈川県の自然景勝地」「神奈川県の寺」等であることが表示される。これらの上位カテゴリ名または下位カテゴリ名を Wiki ゲーターに入力して観光地を調べることもできる。

4. 実装

4.1 DBpedia と SPARQL

Wiki ゲーターの実装で使用した DBpedia[2] と SPARQL[13] について述べる。

DBpedia は Wikipedia のほぼ全てのページから情報を抽出してリンクトデータ (LOD : Linked Open Data) として公開するプロジェクトである。現時点で公開されている日本語版 DBpedia は 2020 年 7 月 1 日時点の日本語版 Wikipedia から情報を抽出して構築されたものである。

DBpedia から情報を検索する方法としては SPARQL (スパークル) を用いる方法がある。SPARQL は RDF (Resource Description Framework) で記述された XML などを取り扱うためのコンピュータ言語の一つである。SQL に類似した記法によって DBpedia 内のデータを検索するためのクエリを記述することができる。

Wikipedia の各記事にはカテゴリが付与されているが, 例えば「日本百名山」について知りたい場合には 日本百名山カテゴリに登録されている 数々の Wikipedia ページを調べるとよい。しかしながらページを一つ一つ見て回るのは手間と労力がかかる。このような場合, 公開されている SPARQL Query Editor (<http://ja.dbpedia.org/sparql>) に以下のような検索文を入力して実行することで, Wikipedia の日本百名山カテゴリ (<https://ja.wikipedia.org/wiki/Category:日本百名山>) に登録された数々の Wikipedia ページに対して検索を行ってその結果を一括取得することができるため, Wikipedia ページを一つ一つ見て回る手間が省ける。

```
SELECT DISTINCT * WHERE {
  ?s dct:subject category-ja:日本百名山.
```

```
?s rdfs:label ?label.
?s rdfs:comment ?comment.
}
```

または,

```
SELECT DISTINCT * WHERE {
  ?s dbpedia-owl:wikiPageWikiLink category-ja:日本百名山.
  ?s rdfs:label ?label.
  ?s rdfs:comment ?comment.
}
```

上記の SPARQL 検索が実行されると, DBpedia の「日本百名山」カテゴリ (<http://ja.dbpedia.org/page/Category:日本百名山>) に登録されている全ての DBpedia ページからラベル情報とコメント情報が抽出され, 以下のような検索結果が表示される (Wikipedia 内の記事のタイトルが DBpedia のラベル情報に対応し, 記事概要がコメント情報に対応している)。

(1) <http://ja.dbpedia.org/resource/トムラウシ山>. ”トムラウシ山 (トムラウシやま) は, 北海道中央部, 上川管内美瑛町と十勝管内新得町の境にそびえる大雪山系南部の標高 2,141 m の山. 「大雪の奥座敷」と称される。地元では昔から...

(2) <http://ja.dbpedia.org/resource/八幡平>. ”八幡平 (はちまんたい) は, 奥羽山脈北部の山群である。標高 1,614 m. 岩手県, 秋田県にほぼ等面積で広がる。広い高原上のあちこちに様々な形の火山起源の小さなピークがそびえ...

(他 98 件 省略)

なお, Wikipedia の各カテゴリは階層構造を作っており, たとえば, <https://ja.wikipedia.org/wiki/Category:日本百名山> にアクセスすると, 「日本百名山」カテゴリは, 「日本の百選」カテゴリの下位カテゴリであることがわかる。 <https://ja.wikipedia.org/wiki/Category:日本の百選> にアクセスすると, 下位カテゴリには「日本百名山」の他「日本の歌百選」「日本 100 名城」などが存在することがわかる。

たとえば, 上記 SPARQL 検索文の「category-ja:日本百名山」の箇所を「category-jp:日本の歌百選」に変更すれば, 「花 (滝廉太郎)」「蛍の光」「めだかの学校」などの検索結果を得ることができる。

4.2 Wiki ゲーター

Wiki ゲーターの主な機能は以下のとおりである。

- (1) DBpedia から取得したデータセット文章のベクトル化
 - (2) データセット文章のクラスタリング
 - (3) 各クラスタの重要語抽出
 - (4) 選択クラスタの再帰的再分割
- 各機能の詳細について述べる。

(1) DBpedia から取得したデータセット文章のベクトル化: データセット文章のベクトル化は BERT を用いて行うようにした。具体的には, SPARQL 検索によって得られた検索結果 (DBpedia のラベル情報とコメント情報のペアの集合) について, コメント情報を BERT によってベクトル化し, そのベクトルにラベル情報を対応付けて記録するようにした。なお, BERT の事前学習モデルとしては東北大学乾研究室が公開しているものを利用し, 各コメント情報は 768 次元のベクトルとして表現されるようにした。

BERT に入力する文章が日本語の場合は, 分かち書き表記された文章を入力する必要がある, そのために MeCab[15] の分かち書き機能を利用している。MeCab を用いる際, 分かち書きされるべきでない固有名詞などを MeCab 標準辞書に多数追加した mecab-ipadic-NEologd [16] を利用した。

(2) データセット文章のクラスタリング:

(1) で作成された文章ベクトルを K 平均法 (K-Means) によって所定個のクラスタに分割する。K 平均法は非階層的クラスタリング手法の 1 つであり, 類似度の高いベクトルをまとめ, 指定した個数のクラスタに分類する方法である。今回の実装では gensim ライブラリ [17] に含まれる K-Means の実装を利用した。作成するクラスタ数 (選択肢数) はユーザが自由に選べるようにした。

(3) 各クラスタの重要語抽出: (2) で作成されたクラスタごとに, TF-IDF 法を用いて重要語 (ユーザに提示する選択肢) を抽出する。この際, 名詞のみを抽出して TF-IDF モジュールに入力するようにした。文章から名詞のみを抽出するためには MeCab の形態素解析機能を用いた。よって各クラスタを代表する重要単語は常に名詞となる。

TF 値は文書内でのある単語の出現頻度であり, その単語の出現回数が多いほど値は大きくなり, 出現回数が少ないほど値は小さくなる。IDF 値は文書集合の中のある単語が含まれる文書の割合の逆数であり, その単語が他の文章にも多く出現しているほど IDF 値は小さくなり, 他の文章にあまり出現していないほど値は大きくなる。TF 値と IDF 値を掛け合わせた TF-IDF 値が大きい単語ほど, 各クラスタ内で重要度が高い単語であるとみなせる。

TF-IDF 値を計算する際に gensim ライブラリに含まれる TF-IDF モジュールを利用している。(2) で作成された各クラスタから TF-IDF 値が最大となった単語 (各クラスタを代表する重要単語) を抽出し, ユーザに選択肢として示す。具体的には Web ブラウザ上に選択肢ボタンとして表示される。

なおストップワード除去処理のために, Web 検索研究

支援プログラミングライブラリ SlothLib[14] を用いたワード除去処理, および, 連続した数字を 0 に置き換える処理などを施した.

- (4) 選択クラスタの再帰的再分割: (3) で表示された選択肢のいずれかがユーザによって選ばれると, その選択肢に対応したクラスタが選ばれたとみなす. 選ばれたクラスタは, (2) に戻って K 平均法 (K-Means) によって所定個のクラスタに再分割され, さらに, 各クラスタに対して同様に (3), (4) が施される. そして最終的にユーザが選択したクラスタ内に 1 つしか情報ページが含まれなくなった時点で, そのページを推薦ページとして Web ブラウザに表示するようにした.

なお, 再帰的処理の途中で同じ名詞がクラスタを代表する重要名詞として繰り返し表示される可能性があるが, すでにユーザによって選択された名詞であることから繰り返し選択させるのは適当ではないと考えた. そこで, 最も TF-IDF 値が大きい名詞が表示済み名詞であった場合は, 2 番目に TF-IDF 値が大きい名詞 (それも表示済みであれば 3 番目に TF-IDF 値が大きい名詞) を選択肢として表示するようにした.

5. 評価

試作した Wiki ゲーターを Web サーバ上に構築し, 大学生 12 名 (女性) に使用させる実験を行った. 被験者にはシステムの使い方について以下の文面で説明した.

『テキスト枠に〇〇〇と入れて更新ボタンを押し, 表示されている複数の選択肢 (ボタン) から好きなものを 1 つ選んでください. 選択肢数は自分で変更できます. 「あなたへのオススメは:」に情報が表示されたら終わりです. 最初に戻ります. この作業を 3 分間繰り返し, 興味ある情報ページが幾つ見つかったか回答してください.』そして実験条件を揃えるため, 全員に対し〇〇〇には「観光地」について「美容」を入れるよう指示した.

また, Wiki ゲーターを使用した場合と使用しなかった場合を比べるため, 同じ被験者に対し, Wikipedia サイトの「観光地」(<https://ja.wikipedia.org/wiki/観光地>) について「美容」(<https://ja.wikipedia.org/wiki/美容>) のページからリンクをたどるなどして, 3 分間で興味ある情報ページが幾つ見つかったか回答させた. 閲覧の仕方や閲覧するページについては被験者の自由とした. そして, Wiki ゲーターと Wikipedia サイトを比較した感想などをアンケート調査した.

実験結果について述べる.

3 分間の作業で興味ある情報ページが幾つ見つかったかについては表 1 の通りとなった. 検索語が「観光地」の場合は Wikipedia の方が多く, 「美容」の場合は Wiki ゲーターの方が多い結果となった.

次に, 被験者から得られたコメントについて記す.

表 1 見つかった情報ページの数 (平均ページ数)

検索語	Wiki ゲーター	Wikipedia
観光地	3.3	4.6
美容	4.6	3.7

検索語が「観光地」の場合は以下のとおりである. どちらかと言うと Wikipedia を好感する意見が多かった.

- (1) Wikipedia では観光地一覧ページが見つかったので, 観光地を早く探せました
- (2) Wikipedia では一覧表示されたものを全体的に流し見する感じでした
- (3) Wikipedia は情報量が多いので読むのが大変だった
- (4) Wiki ゲーターの推薦結果として, 観光地ではなく用語の説明が出ることもあり時間をロスした
- (5) Wiki ゲーターは自分で探す必要がなくて楽だった

検索語が「美容」の場合は以下のとおりである. Wiki ゲーターを好感する意見がほとんどであった.

- (1) Wiki ゲーターでは興味のあるものをクリックしていただくだけで, 知らない言葉も知ることができた
- (2) Wiki ゲーターの方が知らない情報が多く出てきて, 興味のある情報を見つけやすかった
- (3) Wiki ゲーターは全く知らない情報を知れるのが便利でした
- (4) Wiki ゲーターでは初めて知る情報が多く, Wikipedia ではすでに知っていることを詳しく見る事が多かったです
- (5) Wiki ゲーターで, この言葉とこの言葉は結びつきがないのではないかと思うものでも, 選んでみると面白い結果が出てきて楽しかった
- (6) Wikipedia は情報量が多いので, 興味のある情報を見つけるのが大変でした
- (7) Wikipedia では観光地と違って一覧ページがなく, 時間内に興味のある情報までたどり着けませんでした

また, Wiki ゲーターについて以下のコメントが得られた.

- (1) 興味のある分野の知らない知識を身に着けることができると思いました
- (2) わからない分野で, 何から調べていいかわからない時に便利と思う
- (3) 何かの事柄を調べるときにやみくもに調べるのではなく, 自分の興味のあるものをお勧めしてくれるので助かる
- (4) あまり知らない分野で検索キーワードが思い浮かばないときによい
- (5) 新しいことを始める時や新しい分野の勉強を始める時の情報収集などに使えるのではと思いました
- (6) 観光地のような特定の情報を見つけるのには向いていないと思ったが, 美容のような自分の興味のあるカテ

ゴリーの知識を増やしたいときに使えると思った

以上のように、Wiki ゲーターは、特定の情報を見つける用途より、興味のあるカテゴリの知識を増やしたい用途に便利であるという意見が多かった。今後はこの用途に特化した機能を追加するなどしてより有用なものにしたいと考えている。

6. 機能追加

ユーザからのコメントで「観光地を調べた際、行ったことがある場所が出てきてしまうことがあったので、幾つか推薦してもらいたかった」という感想があった。これに対応するために機能追加を行い、ユーザが希望する数のページを推薦できるようにした。

その方法としては、Wiki ゲーターを使って求まった最終的な推薦ページ（A 案）が持つ BERT ベクトルを使い、同一カテゴリ内でこのベクトルに最も近いベクトルを有する Wikipedia ページを 2 位候補とし、その次に近いベクトルを有するページを 3 位候補とするようにした。

ベクトルの近さの指標としてはコサイン類似度を用いることとした。図 5 において、A 案とのコサイン類似度が高い順に B 案、C 案、D 案... である。A 案を含めて幾つの案を表示するかをユーザが「推薦数」で選べるようにした。

Wikipedia [記事 / カテゴリ](#) DBpedia [記事 / カテゴリ](#)

選択肢数: 推薦数:

あなたへのオススメは:

A案: 相翁松の碑
相翁松の碑（あいおいまつのひ）とは、神奈川県小田原市江之浦にある石碑である。相翁松碑（しょうおうしょうひ）とも表記する。1907年（明治40年）に、坪野平太郎、阪谷芳郎、添田壽一らによって建立された。

B案: 山北の桜並木
山北の桜並木（やまきたのさくらなみき）は、神奈川県足柄上郡山北町にある桜の名所。御殿場線山北駅の先端、山北町鉄道公園付近から沼津方面へ約500メートルにわたり、御殿場線の掘割の土手両側に植えられた130本のソメイヨシノが咲く。知る人ぞ知る桜の名所であり、「かながわのまちなみ100選」にも選ばれた。鉄道ファンにとっては定番の撮影スポット。

図 5 代替案表示機能の追加

また、システムを利用する際に Wikipedia ページの最下段に表示されたカテゴリを Wiki ゲーターの検索枠にコピーする手間が必要であったため、カテゴリ入力をしやすくする工夫が必要と考えた。そこで、表示中のカテゴリの上位カテゴリと下位カテゴリを DBpedia から取得して表示する関連カテゴリ表示機能を追加することとした。カテゴリをクリックすると検索枠にそのカテゴリが入るように

なっている。

図 6 の例では、「生活」の上位カテゴリには「人間」または「社会」があり、下位カテゴリには「生活雑誌」「飲食」「行動」などがあることがわかる。ここで「飲食」をクリックしたとすると検索枠の「生活」が「飲食」に差し替えられるので、「飲食」カテゴリについての情報検索を行うことができる。

Wikipedia [記事 / カテゴリ](#) DBpedia [記事 / カテゴリ](#)

上位カテゴリ:
[人間](#) [社会](#)

下位カテゴリ:
[生活雑誌](#) [飲食](#) [行動](#) [親水公園](#) [DIY](#) [いたづら](#) [クオリティ・オブ・ライフ](#) [セルフケア](#) [ペット](#) [マナー](#) [レジャー](#) [住宅](#) [健康](#) [入浴](#) [労働](#) [収入](#) [嗜好品](#) [娯楽](#) [家事](#) [家庭](#) [家族](#) [家計](#) [日用品](#) [消費者](#) [睡眠](#) [福祉](#) [美容](#) [衛生](#) [趣味](#) [幸せ](#) [服飾](#) [生活科学](#) [相談](#)

選択肢数: 推薦数:

あなたへのオススメは:

あなたの好み:

どれが好き?

図 6 関連カテゴリ表示機能の追加

上位カテゴリの取得は以下のような SPARQL 検索で行った。

```
SELECT DISTINCT * WHERE {  
  category-ja:生活 skos:broader ?o.  
  ?o rdfs:label ?label.  
}
```

また下位カテゴリの取得は以下のような SPARQL 検索で行った。

```
SELECT DISTINCT * WHERE {  
  ?s skos:broader category-ja:生活.  
  ?s rdfs:label ?label.  
}
```

7. まとめ

手軽に Wikipedia から興味ある情報を見つけ出せる Wiki ゲーターを提案した。システムから出される幾つかの質問に答えるだけで結果が得られるのが特徴である。被験者実験では、Wiki ゲーターは特定の情報を見つける用途より、興味のあるカテゴリの知識を増やしたい用途に便利であることが示される結果となり、興味深かった。

ユーザ評価後に追加した追加機能の評価については今後の課題としたい。

参考文献

- [1] 日本語版 ウィキペディア, <https://ja.wikipedia.org/wiki/メインページ> (2021)
- [2] DBpedia Japanese, <http://ja.dbpedia.org/> (2021)
- [3] 大山, 竹川, 平田: レビュー文を考慮したゲーム推薦システムの実現に向けた単語の類似度調整の取り組み, 情報処理学会エンタテインメントコンピューティングシンポジウム 2017 論文集, (2017)
- [4] models.word2vec - Word2vec embeddings, <https://radimrehurek.com/gensim/models/word2vec.html> (2021)
- [5] Kurihara, K., Shoji, Y., Fujita, S., Durst, M.J.: Target-Topic Aware Doc2Vec for Short Sentence Retrieval from User Generated Content, http://shoji-lab.jp/research/paper/iiWAS2019_shoji_TTA-D2V.pdf (2019)
- [6] Lau, J.H. and Baldwin, T.: An Empirical Evaluation of Doc2Vec with Practical Insights into Document Embedding Generation, <https://arxiv.org/abs/1607.05368> (2016)
- [7] models.doc2vec - Doc2vec paragraph embeddings, <https://radimrehurek.com/gensim/models/doc2vec.html> (2019)
- [8] 難波: 類似内容の特許請求項の自動対応付け, 情報処理学会研究報告 Vol. 2021-DC-120 No.7 (2021)
- [9] Devlin, J., Chang, M., Wei, L., Lee, K., Toutanova, K.: ERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, <https://arxiv.org/abs/1810.04805> (2021)
- [10] 市村: 口コミ解析と好み診断により手早く旅行先を推薦するサービス, 情報処理学会 DICOMO 2020, 1F-1, (2020)
- [11] Iyengar, S.: 選択の科学, コロンビア大学ビジネススクール特別講義, 文春文庫 (2010).
- [12] Schwartz, B.: 選択のパラドックスについて, https://www.ted.com/talks/barry_schwartz_on_the_paradox_of_choice, TED Global (July 2005).
- [13] Virtuoso SPARQL Query Editor, <http://ja.dbpedia.org/sparql> (2021)
- [14] SlothLib, Web 検索研究支援プログラミングライブラリ, 日本語ストップワード, <http://svn.sourceforge.jp/svnroot/slothlib/CSharp/Version1/SlothLib/NLP/Filter/StopWord/word/Japanese.txt> (2021) pp. 223-227 (2017)
- [15] 工藤: MeCab - Yet Another Part-of-Speech and Morphological Analyzer, <https://taku910.github.io/mecab/> (2021)
- [16] mecab-ipadic-NEologd : Neologism dictionary for MeCab, <https://github.com/neologd/mecab-ipadic-NEologd/blob/master/README.ja.md> (2021)
- [17] gensim, Free Python Library, <https://radimrehurek.com/gensim/> (2021)
- [18] <https://www.python.org/> (2021)