2F1-2in

印象語と仕様データのマッピングに基づく情報検索サービス

Information Retrieval Service by Mapping Specified Terms with Impression Words

多川勇介[*1] Yusuke TAGAWA
相原康弘[*1] Yasuhiro AIHARA
王慧俊[*1] Keisyun OH
南裕也[*2] Yuya MINAMI
並河大地[*2] Daichi NAMIKAWA
金子雅志[*3] Masashi KANEKO
山口 高平[*1] Takahira YAMAGUCHI

[*1] 慶応義塾大学 Keio University

[*2]NTT サービスインテグレーション基盤研究所 NTT Service Integration Laboratories
[*3]NTT ネットワークサービスシステム研究所 NTT Network Service Systems Laboratories

The goal of the study is to enable search with impression words by mapping these words with specified terms. To take in service marketing to mapping approach, we intend to achieve sensitive retrieval by impression words.

1. はじめに

ハードウェアの発達により画像，音楽といったさまざまなコンテンツを扱うことが容易になった．それに伴い従来の文字列マッチによる検索とは異なる，文章化されていないコンテンツに対する検索エンジンの研究が盛んに行われている．その中の 1 つとして印象語による検索が挙げられる．印象語とは個人によって感じ方の異なる「きれい」「近い」といった感覚的な単語であり，印象語を検索に利用することによって直感的な検索が可能となる．印象語による検索の先行研究としては静止画に対する検索では印象語による絵画データベースの検索[栗田　95]や色彩分布と印象語に基づく絵画データの検索[八村　96]，音楽に対する検索では印象に基づく楽曲検索システムの設計・構築・公開[熊本　06]などがある．

本研究の目的は，印象語を仕様データにマッピングし，印象語による情報検索を可能にすることである．印象語と仕様データのマッピングを行うことで，精度の高い印象語による情報検索エンジンを実現させる．

2. システム概要

本節では印象語を用いた飲食店検索サービスシステムの提案について説明する．
システム概要を図1に示す．

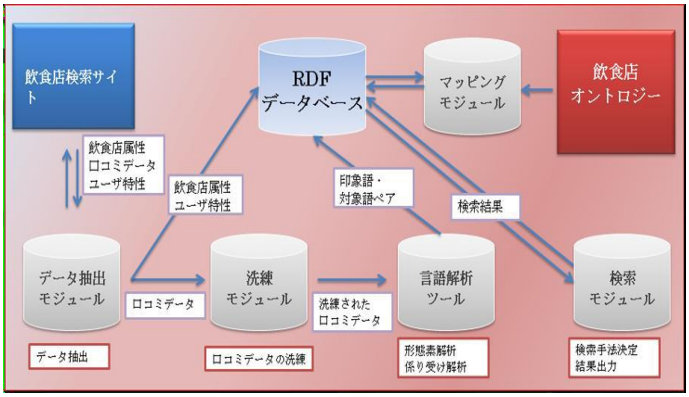


図 1　提案システム概要

提案システムは大きく 3 つに分けられる．

- 飲食店検索サイトからのデータ抽出及び洗練
- 印象語と飲食店データのマッピング
- 検索モジュール

この 3 項目についてそれぞれ説明をする．

3. 飲食店検索サイトからのデータ抽出及び洗練

API[*1]やクロール技術を用いた飲食店検索サイトからのデータ抽出と洗練については同研究室で開発したデータ抽出モジュールを用い，飲食店の仕様データを正規化した RDF データベースを作成する．

飲食店の仕様データは WebAPI[*1]から取得する．取得できるデータは約80項目あるが，アクセス方法や住所，営業時間などは自由記述のまま書かれており，そのままでは検索に用いることができない．そこで，正規化が必要な項目について仕様データの構造化を行っていく必要がある．

飲食店情報は東京都内だけで1万件以上あり，手作業でデータの構造化を行うには効率が悪い．そこで Java を用いて仕様データの構造化プログラムを作成する．

今回はアンケートの結果から，飲食店検索に用いられやすい仕様データのなかで，正規化が必要と判断されるアクセス方法，住所，営業時間についての正規化と構造化を行っていく．

3.1 アクセス方法の構造化

アクセス方法は以下のように店舗ごとに<access>タグの中に自由記述されている．

<access>　JR 新宿駅東口徒歩 5 分／西武新宿線西武新宿駅正面口徒歩 3 分</access>

この中で検索に用いる事ができる仕様データは「路線名」「駅名」「出口」「移動方法」「所要時間」である．しかし，このままではどこの駅から何分かかるかを判断できないので仕様データを抽出し，検索に使用できるように構造化する必要がある．

連絡先：多川勇介，山口高平，慶応義塾大学理工学部
〒223-8522　神奈川県横浜市港北区日吉　3-14-1
{y_tagawa,yamaguti}@ae.keio.ac.jp

*1　HotpepperAPI：
http://webservice.recruit.co.jp/hotpepper/reference.html
*2　Mecab:http://mecab.sourceforge.net/

具体的な流れは,
- 文字列の全半角の統一などの前処理
- 駅名の抽出
- 出口の抽出
- 移動方法, 所要時間の抽出

の繰り返しである. 出口, 移動方法と所要時間は書かれていない場合や, ひとつの駅に複数書かれている場合がある. 書かれていない場合は空白にし, 複数書かれている場合は駅名ノードを複数作成することで解消する.

また, 構造化を行う際に, 駅名をキーとして次の駅名が出るまでの文字列を 1 つのアクセス方法候補として構造化を行っていく. 1つのアクセス方法に「路線名」「駅名」「出口」「移動方法」「所要時間」のセットが複数記載されている場合にはそれぞれを1アクセス方法として構造化する.

● 前処理

API から得られるデータには半角と全角が混在したまま書かれている場合が多い. 文字の統一は以降で行う正規表現の処理の際のための前処理として数値やアルファベットをすべて半角に統一する.

● 駅名の抽出

飲食店のアクセス方法にはかなりの確率で駅名が含まれている. まず形態素解析[*3]を行い,「固有名詞＋駅」となっている部分から駅名を抽出する. この時,「代々木西口徒歩 3 分」というように「駅」を省略している場合があるので形態素解析に引っかからなかった場合には今回都道府県別に作成した駅名データベースとの照合を行う. 都道府県別にした理由は店舗の住所情報から, 店舗の所在都道府県の駅名データベースのみを参照することで検索精度を向上させ, 駅名抽出の時間短縮を図るためである.

● 出口の抽出

次に出口の抽出である. 先に駅名を抽出し, その後方にある出口を正規表現を用いて抽出する.

"[A-Z]\\d*(口\|出口)\|(東\|西\|南\|北\|中央\|正面)口\|\\d 番(口\|出口)"

よくある基本的な名称の出口はこれで抽出できるが, 抽出されなかった場合にはさらに形態素解析を用いて「固有名詞＋口」となるものを抽出する.

● 移動方法, 所要時間の抽出

最後に移動方法, 所要時間の抽出であるが, これは正規表現で抽出する.

"((徒歩\|電車\|バス\|車)\\d+(分\|秒))\|(直結)\|(\\d+分)\|(すぐ)"

アクセス方法の記述を分析したところほとんどの場合がこの正規表現で取ることができる.

上記の3ステップを踏んで構造化したものが下図である.

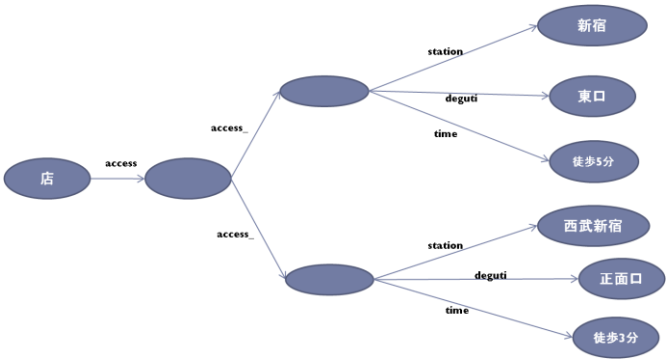


図 2　構造化されたアクセス方法

以上のようにして東京都と神奈川県内約 17000 件の店舗のなかから無作為に 50 店舗抽出し構造化したところ, 正しく構造化できた割合は 90％であった.

3.2 住所の構造化

店舗の住所は以下のように店舗ごとに<address>タグの中に自由記述されている.

<address>神奈川県相模原市相模大野3－16－1　レガロビルB1,1F</address>

住所についても今後の発展を考えて構造化をしていく.

今回は「都道府県市区町村」「番地」「建物名」「階数」に分けて抽出し, 構造化する.

具体的な流れは
- 文字列の全半角の統一などの前処理
- 階数の抽出
- 番地の抽出
- 都道府県市区町村の抽出
- 建物名の抽出

である. このような順にした理由はなるべく簡易的なプログラムの書き方でより多くの記述に対応できるようにするためである. 住所の正規化については既存のプログラムが公開されているが, 今回はデータベースとの照合を行わずに幅広い表記ゆれに対応できるように工夫した.

● 前処理

アクセス方法の場合と同様に, 前処理として数値やアルファベット, ハイフンをすべて半角に統一する. また, 丁目などをハイフンに置き換える.

● 階数の抽出

まず階数を抽出する. これは階数の記述方法が多岐にわたり書かれている場所も様々であることから, まず階数を抽出して元の文から削除する.

番地の後ろに ***3-16-1-3F*** と書かれている場合や, *3～4階, 3, 4階, 3階4階*など書き方は様々である. このような記述方法に対して正規表現で階数を抽出する. 今回は特に最低階と最上階を抽出する.

● 番地の抽出

次に番地の抽出である. この時点で読み込む住所は元の住所情報から先ほど抽出した階数部分が削除された形である.

<address>神奈川県相模原市相模大野3－16－1　レガロビル</address>

この文から番地を正規表現で抽出する. 番地部分が「数値―数値―数値」という形式であることに着目し,

"((\\D+)(\\d*-\\d*-\\d*)\|(\\D+)(\\d*-\\d*)\|(\\D+)(\\d*))(.*)"

というように記述する. 今回使用した API のデータではすべての番地が数値とハイフンで書かれていたが, 漢字で書かれている場合にも「丁目」「番地」「号」をハイフンに置き換えることで応用できる.

● 都道府県市区町村の抽出

都道府県市区町村は多くの既存のプログラムでは住所データベースとの文字列マッチを行っているが, 今回は都道府県市区町村を番地の前方にある塊として抽出する. 先程の番地を抽出する正規表現の***(\\D+)***にあたる部分が都道府県市区町村である. このように抽出することでデータベースとの照合に比べ抽

*3 茶筌:http://chasen.naist.jp/hiki/ChaSen/

出スピードが早くすむ.

● 建物名の抽出

建物名は番地の後方にある文字列すべてを建物名として当てはめる.

上記のステップを踏んで構造化したものが下図である.

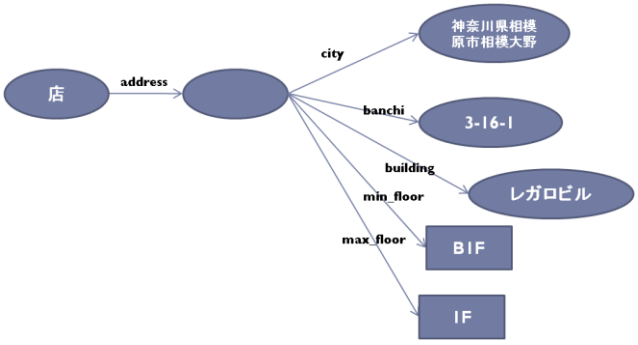
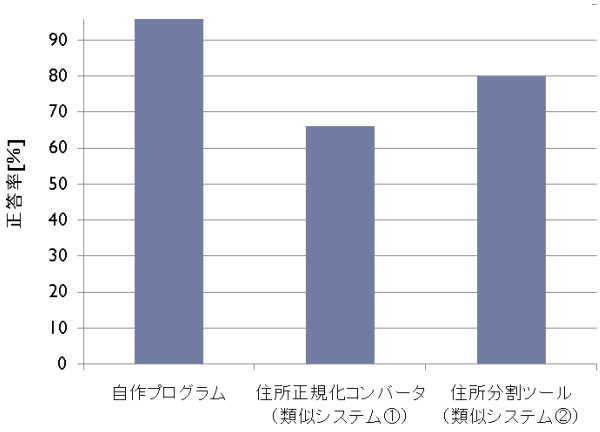


図 3　構造化された住所

以上のようにして東京都と神奈川県内約 17000 件の店舗のなかから無作為に 50 店舗抽出し，本プログラムと，既存の住所正規化プログラム[*4,5]で「都道府県市区町村」「番地」「建物名」「階数」が全て正しく取れているものを正答とした場合の正答率の表を次に示す.

表 1　住所の正規化の正答率



既存の住所正規化プログラムでは住所を頭からデータベースとの照合を行っているために表記ゆれに弱く，正答率が下がっている. しかし本プログラムでは 98%の正答率を誇り，ほぼ正しく構造化が出来ていることがわかる.

3.3 営業時間の構造化

店舗の住所は以下のように店舗ごとに<open>タグの中に自由記述されている.

<open>11 :30 ～15 :00 (LO14 :30)17 :00 ～翌 1 :00 (LO24 :30) 日曜&祝日 12:00 ～24:00</open>

営業時間は記述のされ方がかなり多岐に渡っているのでまず大きく3つのパターンにわけ，それぞれのパターンに合わせて構造化していく.

また，前処理として「翌 1:00」という記述の場合には24を足し，「25:00」として構造化を行っていく.

下図のような構造化を目指す.

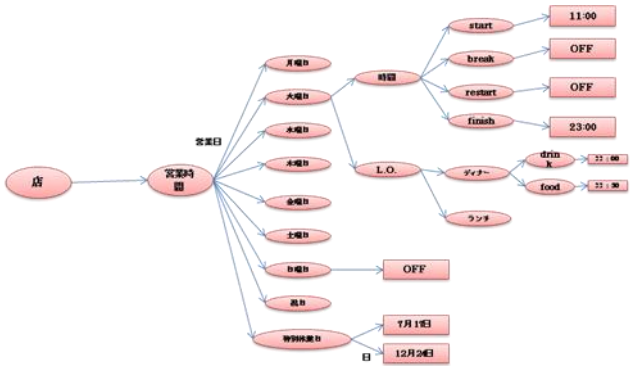


図 4　構造化された営業時間

また，ラストオーダーに関しては未完成であり，今後の課題である.

● パターン1：時間帯や曜日によって分けて記述していないもの

例：***11:30～翌5:00***

この場合には正規表現でコロンの前後の数値を時刻として取り出し，すべての曜日の開店時間と閉店時間に入れる.

● パターン2：最初に曜日によって分けて記述しているもの

例：***月～土/17 :30～23 :30 (L.O.23 :00) 日・祝/17 :30～23 :00 (L.O.22 :30)***

この場合には曜日と，その後方で直近にある時刻を開店時間と閉店時間として割り振っていく.

● パターン3：最初に時間帯によって分けて記述している

例：***ランチ 月～金 11:30～14:00 ディナー 月～土 17:00～23:30 日・祝 16:30～22:30***

この場合にはまず「ランチ」や「ディナー」をキーとして，その後方にある曜日や時間を塊として分割する. それからパターン2と同じように処理していく.

以上のようにして約 17000 件の店舗情報のうちパターンの判別可能率は 90%程で，構造化の正答率は 7 割強という結果になった. 営業時間に関しては記述の形式が多岐に渡るのでプログラムの場合分けの量に比例して正答率も上がっていく.

4. 印象語と飲食店情報のマッピング

印象語から飲食店情報の仕様データへのマッピングを行う. 今回は特にサービスマーケティングの観点からマッピングを行う.

マーケティング・ミックスとは，マーケティング戦略において，望ましい反応を市場から引き出すために，ツールを組み合わせることである.

製造業の分野では，ジェローム・マッカーシーが 1961 年に提唱した分類「4P（Product：製品，Price：価格，Place：流通，Promotion：プロモーション）」を用いてマーケティング・ミックスが語られることが多いが，サービス分野に置いては，4P に加え，さらに 3 つの P（Physical evidence：物的証拠，Process：プロセス，People：人）で分類される 7P によってマーケティング・ミックスが検討されるのが一般的である. この7P を飲食店の仕様データと結びつける.

そして印象語はサービスの質を測定する手法の一つである

*4　住所正規化コンバータ：
http://www.addressmatch.jp/anormapp/demo.jsp
*5　住所分割ツール：
http://www.un-exp.com/add2possummary.html

SERVQUAL の一つであるの飲食店向け DINESERV[Stevens 10]の 5D を用いて印象語と5D を結びつける.

さらに, 7P と5D を結びつけることで間接的に印象語から飲食店の仕様データとのマッピングが可能になり, 印象語から飲食店検索が可能になる.

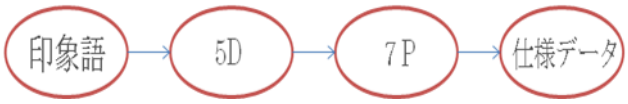


図 5　印象語から仕様データへのマッピング

このようにマッピングを行っていく. 現段階では方針のみであるが, マッピングの実装中である. 実際にマッピングしたものの詳細は全国大会当日に発表の予定である.

また, マッピングの有用性を図るためのアンケートなどを用いた評価実験も今後の課題である.

5. 検索モジュール

4.で印象語から仕様データへのマッピングが可能になったことで印象語を用いた飲食店の検索が可能になる. 今回は特に, 外出先での検索をターゲットとして音声を用いた検索を提案する. 検索システムの全体像が図 6 である.

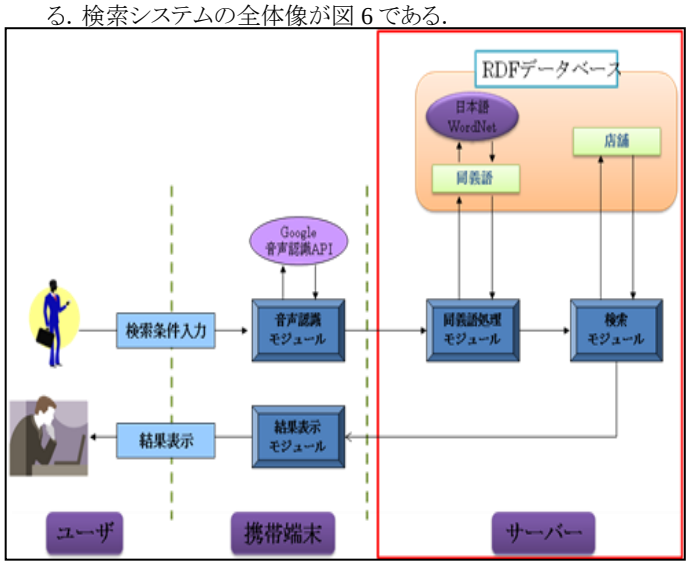


図 6　検索システム全体像

ユーザはスマートフォンを用いて印象語を含んだ検索条件を音声で入力する. そして入力された音声は Google 音声認識 API[*5]を用いて音声をテキストデータに変換する. そのテキストデータの同義語を同義語処理モジュールにおいて洗練し, 検索モジュールにおくる. そこで印象語から対応する仕様データが導かれ, その仕様データに合致する店舗を 3. で作成した RDF データベースより抽出し, 出力する. その際, より適合度の高い店舗が検索結果の上位に来るように重み付けをする. 図 7が Android 端末を用いた検索アプリのサンプル画面である.

メニューボタンを押し, 音声認識を立ち上げて印象語をそれぞれ単語単位で入力していく. そして検索ボタンで実際に検索モジュールに印象語に関連した店舗を検索し, 結果表示を行う.

図 7　印象語検索サンプル画面

6. 結び

本論文ではサービス・マーケティングの視点を取り入れて印象語と仕様データをマッピングする方法を提案した. マッピングを元に印象語検索システムを構築することによって, 検索結果の満足度の向上や, 精度向上が可能になるだろう.

今後の課題は, 実際にマッピングを完成させ印象語検索を可能にすることである. また, 印象語と仕様データのマッピングはユーザのセグメントによっても変えていく必要があるだろう. 例えば学生の「高級」と中年以上の社会人の「高級」とでは印象に対する定量値が異なるだろう. 真にユーザ特性を考慮した検索を行うことを考えた場合, ユーザの使用履歴による学習が必須となってくると考えられる.

また, マッピングの妥当性についても今後客観的なマッピングの評価を考慮していく必要があるだろう.

入力インタフェースについては現段階では音声認識を利用し, 単語ごとの入力としているが, 自然な話し言葉から印象語やその対象語を見つけ出し, 検索が行えるようにすることを目標としている.

本システムは印象語を用いた検索のひな型的な位置づけが出来ると考えており, 今後ユーザ特性を考慮したシステムを構築することを考えた際, ベースとして利用できると考えている.

参考文献

[栗田 92] 栗田多喜夫, 加藤俊一, 福田郁美, 坂倉あゆみ," 印象語による絵画データベースの検索", 情報処理学会論文誌(1992/11)

[八村 95] 八村広三郎, " 色彩分布と印象語に基づく絵画データの検索", 情報処理学会研究報告, (1995/9)

[熊本 06] 熊本忠彦, 太田公子,"印象に基づく楽曲検索システムの設計・構築・公開", 人工知能学会論文誌 21 巻第 3 号 K, pp.310-318(2006 年)

[Stevens 10] Pete Stevens, Bonnie Knutson, Mark Patton, Dineserv: A Tool for Measuring Service Quality in Restaurants, Cornell Hotel and Restaurant Administration Quarterly November 8,2010,p.56-60

*5　Google 音声認識 API : http://developer.android.com/reference/android/speech/package-summary.html