形態素解析を用いた検索支援キーワードマップの提案

広瀬研究室 4 年
C1190245 一居洋平

令和 4 年 1 月 11 日

概要

キーワードマップとは, 自分が調べたい情報を特定のキーワードを軸としてその関連するキーワードを広げた図や表のことである。SNS が発展した現代では誰でも情報が取得できる一方で, 検索するという行為を苦手とする人が多く存在する。そこで本研究では, 検索を苦手とする人が簡単に知りたい情報を取得可能にするために形態素解析を用いたキーワードマップ作成システムを作成, 提案する。

目次

第 1 章

はじめに

本章では研究の背景と目的について説明する。

1.1　背景

インターネットが急速に発展した現代では, 誰もがスマートフォンや PC などといった電子端末を持ち歩いている。知りたい情報が誰でも簡単に取得できるようになった一方で, 検索エンジンを用いたキーワード検索を苦手とする人が多くいることが問題となっている。20 代から 60 代の男女 4407 名に、普段インターネットを利用して検索をする際の, 検索方法や行動に関するアンケート調査を行った結果において, 検索エンジンを利用する際のキーワード検索方法は何かという質問に「1 回検索しても答えが出ないので, 複数回検索する」,「わからない」が 2.5 割を占めた。これらから, 知りたい情報を検索しても情報が得られないことや情報を得るために時間がかかってしまう。以上のことを踏まえて, 本研究では, 誰でも簡単に自分が知りたい情報を検索できるキーワードマップの作成を行う。

1.2　研究目的

本研究では, 利用者が入力した検索キーワードから, 関連するキーワードを正確に取得し, キーワードマップを生成する Ruby プログラムを作成する。

第 2 章

関連研究と類似サービス

本章では,Web 検索機能における情報の関連研究と既存のキーワードマップ生成サービスを調査する。

2.1　関連研究

形態素解析を用いたシステムの研究事例について紹介する。

2.1.1　形態素解析を用いたトレンド情報の抽出

寺内, 吉田らの研究 [1] では, 特定分野として経営工学に関わる研究に焦点を当て, その中から「経営情報学会」の論文を論文や図書・雑誌などの学術情報を検索できるデータベース・サービス『CiNii』から数年分収集した。トレンド情報の検出手法として形態素解析エンジンの MeCab を用いて, 論文のトレンド情報と捉えられる語句を選別することで抽出している。

2.1.2　Twitter データのクラスタ分析

こちらの研究 [2] では,2016 年 7 月に行われた東京都知事選挙に関する Twitter データのうち, 選挙期間中に投稿されたもの収集し, クラスタ分析を行うことによって, 選挙に関する投稿について分析をした。収集したデータの前処理として日本語形態素解析エンジンの MeCab を利用し, 形態素解析を行っている。

2.2　キーワードマップ生成サービス

キーワードマップとは, ある特定のキーワードを中心として, そのキーワードから派生する関連キーワードを放射状ツリーで階層的につなげた図や表のことである。特定のキーワードの関連キーワード群を視覚的に把握することが可能であり, 検索キーワードの選定やコンテンツ作成などに用いられている。キーワードマップ生成サービスについて調査し, キーワードマップの課題点を探る。

2.2.1　keysearch Beta

keysearch Beta は無料で利用可能なキーワードマップ作成ツールである。 [3] 検索フォームにワードを入力し検索することで, 主に Google サジェストから検索したキーワードと関連するキーワードを自動で取得し, キーワードマップを作成することができる。また作成したキーワードマップから関連するワードのみを抽出

し,CSV や SVG 形式で保存することも可能である。keysearch Beta で作成されたキーワードマップは入力したキーワードのセカンドキーワードまでを抽出した二階層になっており, ひと目で情報を整理することができるという特徴がある。しかし, 入力したキーワードのセカンドキーワードまでしか表示されないことから, 関連キーワードを網羅することができず, キーワード選定には向いていない。

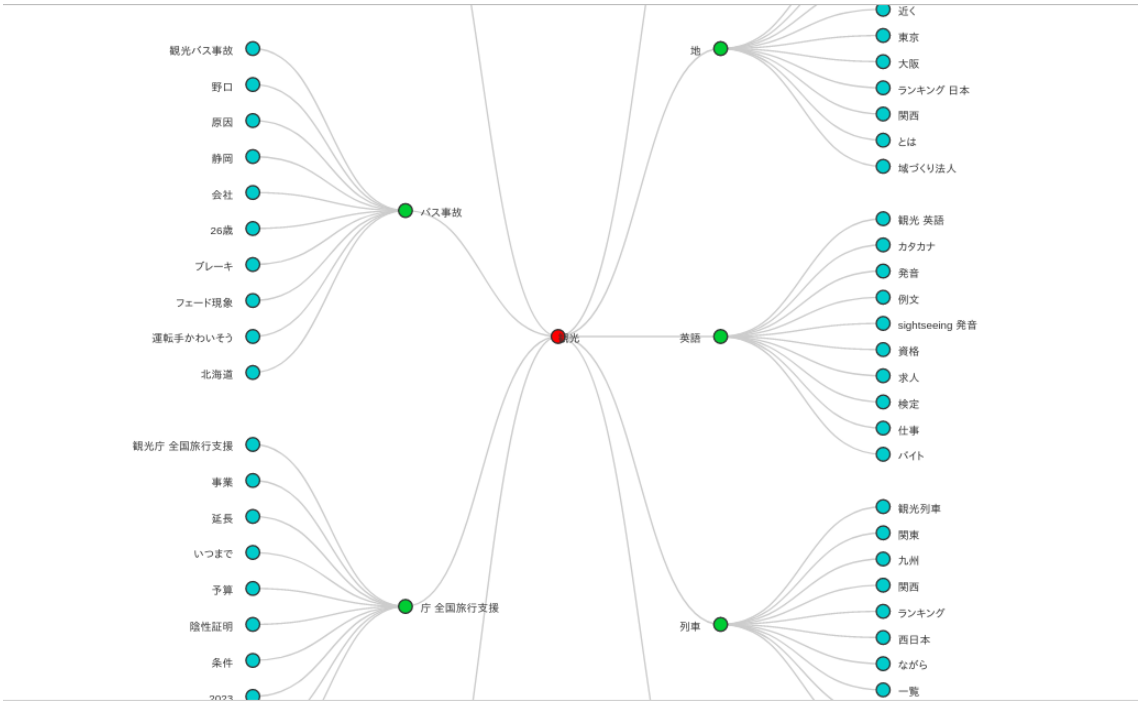

図 2.1　keysearch beta のキーワードマップ

2.2.2　OMUSUBI

OMUSUBI は無料で利用可能なキーワードマップ作成ツールである。 [4] 検索フォームにキーワードを入力することで,Google,Youtube.Amazon,Bing 検索などのサジェストを自動取得し, それぞれの検索エンジンに沿ったキーワードマップを作成することができる。OMUSUBI で作成されたキーワードマップはスクロールで大きさの調整やドラッグすることで上下左右への移動, ノードをクリックすることでそのキーワードを中心に配置することができ, キーワードの関連性を階層ごとに把握しやすいことが特徴として挙げられる。しかし, キーワードマップが表示されないことやノードが重なりキーワードマップとして見づらいことなど不便な部分もある。

2.3　キーワードマップ生成サービスの課題点

課題点として, 一つ目は正確なキーワードを入力しなければ, 取得したい情報が得られないという点である。検索フォームに入力されたキーワードの表記揺れによって, サービスの利用者が取得したい情報とは異なる情報のキーワードマップが生成されることがあるため, 一度の検索で入力するキーワードの表記揺れに対応した検索システムが必要となる。二つ目は, 文章のような長いキーワードを入力した場合, キーワードマップが生成

図 2.2　OMUSUBI のキーワードマップ

されないという点である。キーワードマップ生成サービスは,Google などの検索サジェストからキーワードを取得しているため, 検索キーワードが検索サジェストに一致しない場合, 関連キーワードを取得することができない。よって, あらかじめ入力された検索キーワードを整理し, 検索サジェストに一致しやすくする必要がある。

第 3 章

提案

本章では, 2.3 節で挙げた課題点を踏まえてそれを解決するためのシステムの提案を行う。

3.1　本研究の提案

本研究では, 検索キーワードを入力することで, 誰でも利用することが可能であり, 表記揺れや文章などに対応したキーワードマップ生成することが可能なシステムを構築する。

3.2　課題点を踏まえた改善案

2.3 節で挙げた課題点を踏まえて, キーワードマップ生成に関する改善案を以下のように定義した。

- 検索キーワードの表記揺れ
  入力された検索キーワードの表記揺れは検索サジェストを取得する前に検索キーワードのあいまい検索を行うことによって, 正確な情報を取得可能にする。
- 検索キーワードの整理
  文章のような長いキーワードを入力した場合, 検索サジェストと一致しやすくするためにキーワードの形態素解析を行うことによって, そのキーワードのうち検索に適した要素のみを抽出する。
- キーワードマップの網羅性
  多くのキーワードの選定を可能にするためには作成されたキーワードマップ内の関連キーワードを多く取得する必要がある。そこで, 関連キーワードをより多く取得するために検索キーワードから取得したサジェストからそれを検索キーワードとして再度検索サジェストを取得する。この動作を数回繰り返すことによって, よりキーワード選定に向けたものにする。

3.3　あいまい検索の活用

あいまい検索は, 検索キーワードとして入力された情報と類似した情報を検索結果として取得することが可能な検索機能である。あいまい検索の処理方法は, 取得した検索キーワードを言葉として意味を有する部分文字列を選別し, それらを出現回数や出現集中度を考慮したスコアを算出するという構造となっている。

3.3.1　検索キーワードの形態素解析

検索キーワードの整理には形態素解析を用いる。形態素解析は自然言語処理の一部であり, 自然言語を言葉が意味を有する表現要素の最小単位である形態素まで分割し, それらを品詞として分類, 判別する文字列抽出法である。この技術は検索エンジンの処理やニュースアプリにおける画面上のタイトルの文字組みなどで活用されている。主な形態素解析エンジンとしてオープンソースの形態素解析ツールである「MeCab」や Python 上における形態素解析に有用な「Janome」などが挙げられる。

3.4　システム概要

本研究では, 検索キーワードを入力することで, その検索キーワードの情報を中心としたキーワードマップの生成が可能なシステムを作成する。このシステムは Ruby を利用して作成され, 検索キーワードに関する情報は Google サジェストから取得している。検索キーワードの表記ゆれや長い検索キーワードなどを入力した場合でも, 情報を取得し誰でも簡単に利用可能で, より多くの情報を取得することでキーワードの選定に向けたものとする。

第 4 章

システムの設計

本章では, システムの設計を行う。

4.1　システムの構成

本システムは,

4.2　使用技術/開発環境

本節では本システムで使用する技術についてまとめる。

4.2.1　Ruby

Ruby はまつもとゆきひろ氏によって開発されたオブジェクト指向のプログラミング言語である。国産のプログラミング言語としては日本初の国際電気標準会議において国際規格に認証されている。特徴としてクラス定義, ガベージコレクション, 強力な文字列操作や正規表現処理, マルチスレッド, 例外処理機能, イテレーターとクロージャ,Mixin, 演算子オーバーロードなどの機能がある。Ruby は本システム内の検索キーワードの正規表現処理と検索サジェストの取得するプログラムとして利用する。本研究で用いる Ruby のバージョンは ruby 2.7.0 を用いる。

4.2.2　MeCab

MeCab は京都大学情報学研究科と日本電信電話株式会社コミュニケーション科学基礎研究所の共同ユニットプロジェクトを通じて, 開発されたオープンソースの形態素解析エンジンである。言語, 辞書, コーパスに依存しない汎用的な設計方針を採用しており, 数多くのプログラミング言語で使用することが可能である。MeCab による形態素解析の実行結果は, 入力した自然分を形態素にまで分解した後にそれぞれ表層形, 品詞, 品詞細分類 1, 品詞細分類 2, 品詞細分類 3, 活用型, 活用形, 原形, 読み, 発音という流れで出力される。本システムでは,MeCab による形態素解析の出力結果を Ruby のプログラム内で利用するために MeCab-Ruby を用いる。

MeCab による形態素解析の例

```
~$ mecab
この文章を形態素解析します
この 連体詞,*,*,*,*,*, この, コノ, コノ
文章 名詞, 一般,*,*,*,*, 文章, ブンショウ, ブンショー
を 助詞, 格助詞, 一般,*,*,*, を, ヲ, ヲ
形態素 名詞, 一般,*,*,*,*, 形態素, ケイタイソ, ケイタイソ
解析 名詞, サ変接続,*,*,*,*, 解析, カイセキ, カイセキ
し 動詞, 自立,*,*, サ変・スル, 連用形, する, シ, シ
ます 助動詞,*,*,*, 特殊・マス, 基本形, ます, マス, マス
EOS
```

4.2.3 Graphviz

Graphiviz(Graph Visualization Software) は AT&T 研究所によって開発されたオープンソースのグラフ画像作成ツールである。データ記述言語の一種で, グラフをデータ構造としてプレーンテキストで表現する言語である DOT 言語を用いて, テキストファイルを png や jpeg などの画像ファイルに変換することで, グラフ画像を生成することができる。本システムでは, 入力された検索キーワードと取得した検索サジェストのデータをキーワードマップの画像として生成するために用いる。また,Ruby のコードから Graphviz によるグラフ生成を可能にするために Gviz を用いて画像を生成する。

第 5 章

システム開発

4 章を元にシステム開発を行う。

5.1　検索キーワードの入力と形態素解析

検索キーワードを入力し, その検索キーワードの形態素解析を行う。gets メソッドを使用し, キーワードマップとして生成するための核となる検索キーワードを文字列としてキーボードからのデータ入力を可能にする。取得したデータをもとに MeCab::Tagger クラスのインスタンスを取得し,parse メソッドを使用することで形態素解析の結果を文字列で取得する。形態素解析の結果はそれぞれ表層形と形態素解析の結果である文字列が品詞, 品詞細分類 1, 品詞細分類 2, 品詞細分類 3, 活用型, 活用形, 原形, 読み, 発音という順番で出力され, 最後に EOS と出力される。検索キーワードとして利用する箇所は原形・読み・発音の三種類である。これらを組み合わせて表記揺れや誤表記に対応した検索キーワード生成する。これらの要素を文字列から抽出し, 利用するために表層形とそれ以外の文字列というように二種類の文字列に分割する必要がある。全ての文字列を分割し配列に要素を入れるために split メソッドを用いる。

検索キーワードの形態素解析

```
sentence = gets.chomp
tagger = MeCab::Tagger.new
keyword = tagger.parse sentence
array = keyword.split
array2 = keyword.split
```

5.2　検索キーワードの構築

本節では形態素解析した文字列を検索キーワードとして再構築するコードの処理について説明する。5.1 節で行った処理から検索キーワードとして利用するために検索キーワードを形態素解析した情報の入った配列 array から不必要な情報を抽出する必要がある。不必要となる情報は形態素解析の結果の品詞, 品詞細分類 1, 品詞細分類 2, 品詞細分類 3, 活用型, 活用形の部分と文字列の EOS である。EOS は delete メソッドを用いて, 文字列の EOS をマッチさせることで配列 array から除外する。形態素解析の結果の品詞, 品詞細分類 1, 品詞細分類 2, 品詞細分類 3, 活用型, 活用形の部分は正規表現を用いる。原形は MeCab の辞書とマッチした表層形

と同じ文字列で出力されるため, 正規表現で表層形の文字列を指定して原形以降の文字列を抽出し, それ以前の文字列は配列 array から除外する。同じ文字列と区切り文字は split メソッドと uniq メソッドを利用することで除外する。また, 検索キーワードに存在しない単語や MeCab の辞書に未登録の単語を入力した場合は形態素解析の結果として原形が出力されないために表層形と同じ文字列をその要素に入れる処理を行う。

これらの処理を行った配列 array から表層形の文字列を除外する処理を行う。表層形の文字列は配列 array の奇数番目に位置する要素であるために each_slice メソッドと map メソッドを利用して, 奇数番目の要素を抽出し, 除外する。inject メソッドと map メソッドを利用して配列 array 内に残った要素同士で総当りで組み合わせの配列を生成し, その配列内の要素を join メソッドによって結合することによって表記揺れや誤表記に対応された検索キーワードが生成される。

検索キーワードの生成のコード

```
array.delete("EOS")
i = 0
(array.length/2).times do
if /#{array[i]}/ =~ array[i+1]
   array[i+1] = array[i]+$'
   array[i+1] = array[i+1].split(/[,|\/]/)
   array[i+1] = array[i+1].uniq
   i += 2
else
   array[i+1] = [array[i]]
   i += 2
end
end

i = 0
array = array.each_slice(2).map(&:last)
if array.length > 1
array = array.inject(:product).map(&:flatten)
array.length.times do
array[i] = array[i].join()
    i += 1
end
else
    array= array[0]
end
```

5.3　検索サジェストの取得

本節では検索サジェストの取得とそれのセカンドキーワードの取得について説明する。本システムでは検索サジェストとして Google サジェストを取得する。Google サジェストの取得には Google Suggest API を利用する。Google Suggest API は下記の URL 末尾にサジェストを取得したい検索キーワードを入力すると,XML 形式で検索サジェストが取得できる API である。

Google Suggest API

```
url = "https://www.google.com/complete/search?hl=ja&output=toolbar&q=#[検索キーワード]"
```

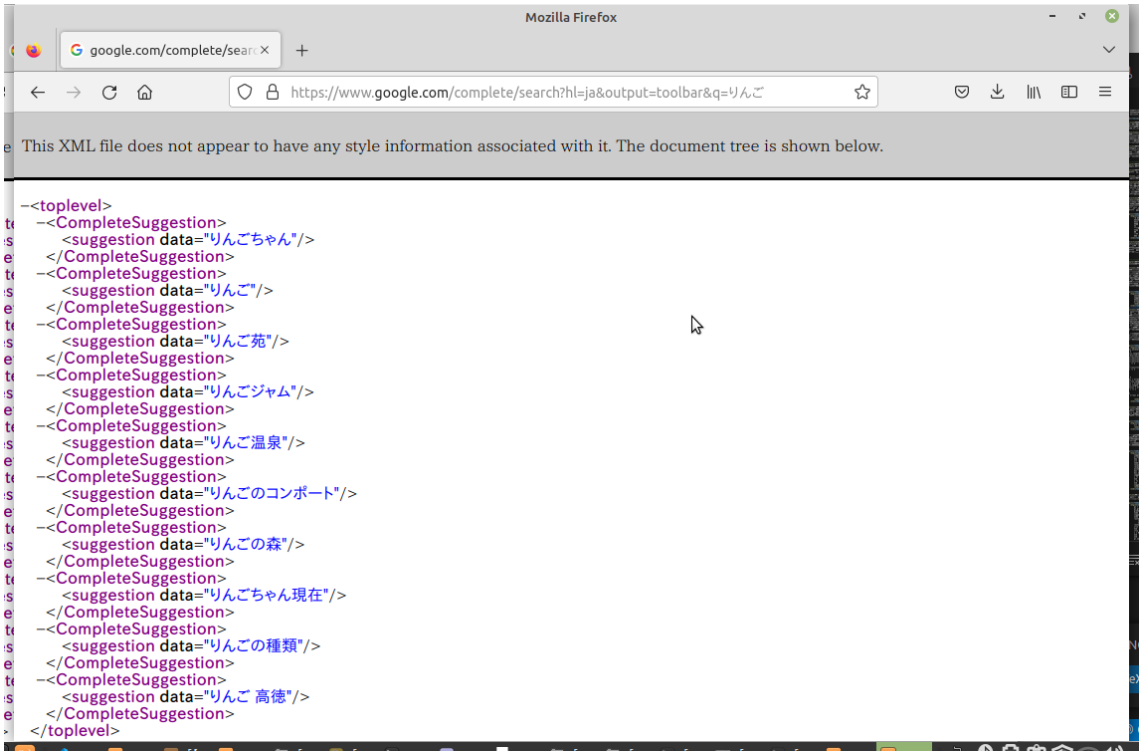

図 5.1　Google Suggest API のサジェスト取得画面

XML 形式で取得したデータは URI.open メソッドを用いて, 文字列として ruby 内に読み込む。Ruby で XML から目的の要素を取り出す処理を行うために remxl ライブラリを利用する。DOM の子要素を取り出す elements メソッドで不必要な要素を指定することで検索サジェストのみを抽出し, 配列 suggest にその結果を格納する。これらの処理を 5.2 節で生成した検索キーワードの要素の数だけ繰り返し, 全ての検索サジェストを抽出する。同じ検索サジェストは uniq メソッドを用いることで配列 suggest から除外する。

セカンドキーワードの取得は検索サジェストの取得で行った処理と同様に取得した検索サジェストをもとに行うことによって取得する。

検索サジェスト取得の処理

```
saggest = Array.new
array.length.times do
key = array[i]
url = "https://www.google.com/complete/search?hl=ja&output=toolbar&q=#
{Addressable::URI.encode(key)}"
response = URI.open(url)
response.charset
encoded = response.read.encode('UTF-8', response.charset)
doc = REXML::Document.new(encoded)
doc.elements.each('toplevel/CompleteSuggestion/suggestion') do |e|
saggest[j] = e.attributes['data']
j += 1
end
i += 1
end
suggest = suggest.uniq
```

5.4　キーワードマップの生成

本節では 5.3 節で取得した検索サジェストをデータをもとにキーワードマップを生成するコードの処理について説明する。キーワードマップの生成は Gviz を用いて行う。キーワードマップの設定として global layout メソッドを利用することでキーワードマップのレイアウト・ファイル名・フォントサイズ・ノードの重なりなどの設定を行うことができる。ノードとはグラフ内における丸や点で表される頂点となるものである。キーワードマップをより見やすくするために, レイアウトは環状型にしてノード同士が重ならない設定にする。

キーワードマップの中心となる検索キーワードとそれに関連するキーワードの設定を行う。作成するキーワードマップは検索キーワードを中心のノードとして環状型に関連キーワードのノードが展開されるマップである。中心に位置するノードに検索キーワードを表示するために,node メソッド内の label に 5.1 節で入力された変数 sentence を設定する。同様に関連キーワードとなるノードは前節で取得した検索サジェストの要素が入った配列 suggest を設定する。route メソッドを用いて, 検索キーワードとなる中心ノードから関連キーワードのノードへと接続するエッジを表示させる。エッジはグラフ内におけるノード間の接続を表す役割を持った線である。これらの設定を検索サジェストの全要素で行い, 同様の設定を検索サジェストと前節で取得した全てのセカンドキーワードをもとに行う。全ての処理を終えるとキーワードマップが生成される。

キーワードマップの設定コード

```
gv = Gviz.new
gv.graph do
global layout:'circo', label:"#{sentence}", fontsize:20, size:15, overlap:false
node :"sentence", {label:sentence}
suggest.length.times do
node :"suggest#{i}", {label: suggest[i]}
route :"sentence" => [:"suggest#{i}"]
suggest2[i].length.times do
node :"suggest#{i}q#{j}", {label: s[i][j]}
route :"suggest#{i}" => [:"suggest#{i}q#{j}"]
j += 1
    end
i += 1
j = 0
end
end
gv.save( :keywordmap, :png )
```

5.5 検索サジェストが取得できない場合の改善案

本節では検索サジェストが取得できない場合に行う検索キーワードの整理について説明する。文章のような長い検索キーワードは Google Suggest API から検索サジェストを取得することができない場合がある。この場合の改善案として, 検索キーワードの形態素解析を行った際に取得した品詞の種類から名詞である単語のみを抽出し, それを検索キーワードとして再度検索サジェストの取得処理を行う。

検索キーワードの整理

```
if suggest.empty?
    v, x, z = 0, 0, 0
    array3 = []
    array3[v] = []
    (array2.length/2).times do
    array3[v][x] = array2[z]
    x +=1
    z +=1
    array3[v][x] = array2[z]
    array3[v][x].sub!(/,.*/m,"")
    z +=1
    x = 0
    v +=1
    if v < array2.length/2
    array3[v] = []
    end
    end
    array3.select! {|dt|
        dt.include?"名詞"
    }
    array3 = array3.transpose[0]
    array3 = array3.join(" ")
end
```

5.6　生成したキーワードマップ

生成したキーワードマップは図 5.2 である。

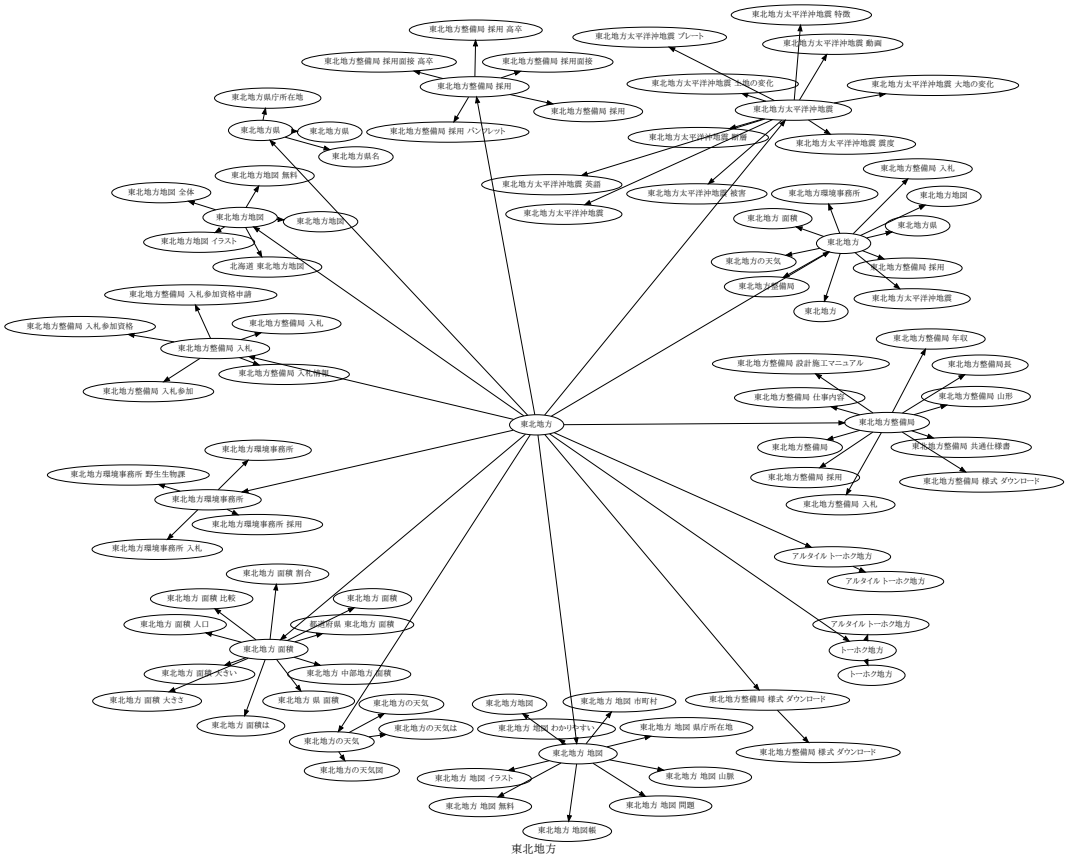

図 5.2　生成したキーワードマップ

第 6 章

結論

本章では, 本研究の結論と今後の展望について述べる。

6.1 結論

本研究では, 検索することが苦手な人に向けた形態素解析を用いた検索支援のキーワードマップを生成する Ruby プログラムを作成した。作成されるキーワードマップは検索に向かない検索キーワードを入力した場合でもキーワードマップとして生成されるため検索することが苦手な人でも求める情報を取得することができる。

6.2 今後の展望

今後の展望として, 検索サジェストを取得するためにキーワードマップの生成に時間がかかってしまう。また, 作成されたキーワードマップは視覚的に見づらい部分があるためにより見やすいマップの生成を行っていく。

参考文献

[1] 寺内 貴洋，吉田 典正, 齋藤 敏雄. “形態素解析を用いたトレンド情報の抽出と可視化システムの構築”
https://www.mlit.go.jp/common/000228861.pdf，(参照 2022-12-11).

[2] 和田 伸一郎. “インタラクティブなデータ・ヴィジュアライゼーション・ツールを用いた Twitter データクラスタ分析”
https://www.soumu.go.jp/main_content/000629037.pdf，(参照 2021-8-18).

[3] keysearch Beta. “keysearch Bet さ r 得る
https://kitsune-room.com/tools/keysearch/，(参照 2022-12-11).

[4] おむすび. “おむすびキーワード”
http://omusubisuggest.appspot.com/，(参照 2022-12-13).