RO-014

マッシュアップを対象とした WebAPI 検索システム

Proposal of WebAPI Search System for Mash-up

井上 紘希† 小板 隆浩‡
Koki Inoue Takahiro Koita

1. はじめに

1.1 背景

インターネットの普及に伴い，様々な Web サービスへのアクセスが可能になった．その中で，Web ビジネスを展開する企業が情報の共有化による Web サービスの向上を目的に WebAPI を公開するようになった．WebAPI が公開されることで，WebAPI を利用して多くの Web サービスの中から，任意の Web サービス同士を組み合わせて新しい 1 つの Web サービスを作り出すマッシュアップという技術が登場し，注目を集めている．マッシュアップを用いることで，開発環境の構築コストやサービスの開発コストを抑えて，新たな Web サービスを開発することができる．その有用性から，現在では，多くの Web サービスでマッシュアップが用いられている．マッシュアップによって Web サービスは新しい価値を持つことが可能となる．このためマッシュアップが盛んに行われるようになり，マッシュアップは広く利用されるようになった．マッシュアップの例を図 1 に示す．

図 1 マッシュアップの例

1.2 目的

ユーザが望んでいる機能を持った Web サービス同士を組み合わせるといった，ユーザの求めるマッシュアップを行うためには，ユーザが必要とする Web サービスの WebAPI を検索する必要がある．しかし，インターネット上に数多く存在する WebAPI の中から，ユーザが仕様を理解し，目的に合う WebAPI の組み合わせを探し出すことは，WebAPI の入出力に関する情報の組み合わせを考慮しながら検索する必要があり，ユーザの負担となる．さらに，WebAPI の関連性を示す情報がないため，マッシュアップに利用するユーザの目的に合う 1 つの WebAPI を数多くの WebAPI の中から選択した後，ユーザの目的に合い，尚且つ入出力情報が一致するものを，もう一度数多くの WebAPI の中から検索する必要があり．WebAPI の組み合わせを検索することは困難な作業となっている．

†同志社大学大学院　理工学研究科　情報工学専攻
‡同志社大学　理工学部　情報システムデザイン学科

WebAPI 検索は今後の WebAPI の増加に伴いさらに困難な作業となり，マッシュアップを行うユーザの大きな負担となっていく．本研究では，WebAPI 検索を対象とし，ユーザが組み合わせる WebAPI を容易に検索できる環境の実現を目的とする．

2. 問題定義

2.1 既存の WebAPI 検索

ProgrammableWeb[1]は，WebAPI に関する情報が集められている，マッシュアップポータルサイトである．ユーザはマッシュアップポータルサイトを利用することで，検索エンジンを用いて WebAPI を検索する場合に比べ， WebAPI の検索を行うことが容易となった．そのため，ユーザは WebAPI 検索を行う際，積極的にマッシュアップポータルサイトを用いている．ProgrammableWeb では，統一フォーマットで情報が整理されていて，キーワードだけでなくカテゴリやタグで検索を行うことができる．図 2 に ProgrammableWeb に登録されている Google Maps[2]の情報例を示す． 図中の Highlights とは，WebAPI の概要情報を表している．

Google Maps: Highlights

Summary	Mapping services
Category	Mapping
Tags	mapping places viewer display
Protocols	JavaScript
Data Formats	XML , VML , JSON , KML
API home	https://developers.google.com/maps/
This is 1 of	107 APIs by Google

図 2 ProgrammableWeb の情報例

2.2 既存の WebAPI 検索の問題

ProgrammableWeb の WebAPI 検索を利用すれば，WebAPI の名前だけでなく，カテゴリやタグでの検索も可能なので，１つのタグから共通の機能を持つ WebAPI を検索することが可能である．検索した WebAPI の情報と共に，マッシュアップに使われているサービスの情報も得ることができる．しかし，WebAPI の組み合わせ検索を行うには，WebAPI の入出力に関する情報が必要になるが，ProgrammableWeb ではその情報を得ることができない．Google Maps の場合，地図上に表示するための位置情報を文字列で入力するのか，数値で入力するのかなどの，入出力の値に関する情報を得ることができない．そのため，WebAPI の組み合わせ検索を行うための情報が，登録されている１つのマッシュアップに使われているという情報だけになってしまう． １つの WebAPI に関する情報を検索することはできても，WebAPI の入出力に関する情報を得ることができないので，WebAPI の入出力を考慮した

WebAPI の組み合わせの検索が困難となることが問題となる．

ProgrammableWeb に登録されている WebAPI とマッシュアップの情報は，ユーザが任意で登録を行っているため，ProgrammableWeb 上に全ての WebAPI などに関する情報が存在するわけではない．そのため，ユーザがマッシュアップを行う際に必要な，WebAPI 自体の情報や，WebAPI を組み合わる際に必要な情報が不足している．特にマッシュアップに関する情報は，1 つの Web サービスを改良したものをマッシュアップとして登録されていることが多く，WebAPI の組み合わせの情報量が少なくなり，目的の組み合わせを発見することが困難となることが問題となる．

2.3 必要要件

マッシュアップをするためには複数の WebAPI が必要である．マッシュアップのための WebAPI 検索では，マッシュアップに利用できる複数の WebAPI による組み合わせを発見することが重要となる．ProgrammableWeb での WebAPI 検索では WebAPI を組み合わせるための入出力に関する情報が不足しているので，WebAPI の組み合わせ検索を行うことが難しい．しかし，WebAPI の関連性を示すことができれば，マッシュアップに利用できる WebAPI の組み合わせを絞り込むことができ，その中からユーザの目的と WebAPI の入出力の両方に一致する WebAPI を探し出すことは，数多くの WebAPI の中から検索するよりも容易となる．従って，組み合わせる WebAPI を容易に検索できる環境を実現するためには，WebAPI の関連性を提示することが必要となる．

ProgrammableWeb 上に存在する WebAPI に関する情報は，ユーザによって任意で登録されたものであり，マッシュアップに使用されている WebAPI の組み合わせも全体の一部である．WebAPI 検索の範囲を ProgrammableWeb からインターネット全体へ拡大することで，ProgrammableWeb 上にないマッシュアップと WebAPI の組み合わせを発見することができ，WebAPI に新たな関連性を持たせることができる．WebAPI に関連性を持たせるとこができれば，組み合わせる WebAPI を容易に検索出来る環境が実現できる．

3. 提案手法

3.1 提案手法概要

WebAPI の組み合わせを検索するためには，WebAPI 検索の範囲を ProgrammableWeb からインターネット全体へ拡大し，ProgrammableWeb に不足している WebAPI の関連性を補足することが必要となるので，WebAPI の組み合わせの検索を支援するシステムとして，検索エンジンの Suggest 機能を用いて WebAPI の関連性を探索する手法を提案する．Suggest 機能とは，ユーザが入力した検索ワードに対して自動で検索頻度の高いワードを補ってくれる機能である．WebAPI 名を検索エンジンの Suggest 機能にクエリとして送信し，返信される Suggest 機能の結果の中から他の WebAPI 名を探索することで，WebAPI に関連性を持たせることができる．WebAPI に関連性を持たせることで，WebAPI の入出力に関する情報の不足により WebAPI の組み合わせ検索が困難になっている点を補い，WebAPI の組み合わせの検索が可能になる．また，検索エンジンの Suggest 機能を使用することで，インターネット全体から情報を取得することができる．

3.2 提案手法全体像

提案手法では 3 つのことを行う．それぞれの手法の詳細を以下に示す．

(1)WebAPI 情報の取得：

検索エンジンの Suggest 機能を利用するために必要となる WebAPI 名や，WebAPI の組み合わせの考慮に必要となる WebAPI が持つ機能などの概要情報は，ProgrammableWeb で一つ一つの WebAPI 毎に情報がまとめられていて，情報の形式が統一されているので，WebAPI 自体の情報を得るのに十分である．本研究の提案手法で使用するための WebAPI の概要情報を ProgrammableWeb から取得する．

(2)Google Suggest の利用：

本研究の提案手法では検索エンジンの Suggest 機能として Google Suggest[3]を使用する．ProgrammableWeb から取得した概要情報の中の APIname をキーワードとして，Google Suggest にクエリを送信し，返信される Suggest 機能の結果を取得する．

(3)結果の出力：

取得した Suggest 機能の結果の中から”Google maps”，”Twitter” など，WebAPI 名があった場合のみその結果を取得する．Suggest 機能で取得した WebAPI は検索頻度が高いものなので，マッシュアップのための WebAPI 検索で検索されている可能性が高く，その WebAPI を組み合わせたマッシュアップが存在する可能性も高い．従って，Suggest 機能で取得した WebAPI は関連性がある WebAPI となる．ここで取得した関連性のある WebAPI の情報を WebAPI 毎にまとめて出力する．

4. 実装

実装環境を表 1 に示す．

表 1 実装環境

実装に用いた環境	
言語	PHP5
仮想サーバ	Apache2.2
データ保存用ファイル	Microsoft Excel 2010 形式

表 1 に示すように，システム実装の開発言語は PHP5，仮想サーバは Apache2.2[4]，データ保存用のファイルとして Microsoft Excel 2010 形式(CSV)のファイルを使用した．

実装概要を以下に述べる．

(1)WebAPI 情報の取得：

提案手法で WebAPI の概要情報を利用するために，ProgrammableWeb から WebAPI の概要情報を抽出するスクレイピング部分は PHP5 で実装した．ProgrammableWeb 上から WebAPI の概要情報を取得するために ProgrammableWeb のサイト構成から APIdirectory に保存されている情報のみを取得した．まず，PHP5 から提供されている PHP_simpleDOMparser を用いて APIdirectory の HTML（HyperText Markup Language）データを取得する．そして，取得した HTML データの構文を解析する．今回の WebAPI に関する情報は Body 内の table に表形式として保存されていたので，その表を取得する．その表の APIname の部分には WebAPI のより詳しい概要情報が提供されているリンクが貼ってあるので，それぞれのリンクを取得して，

リンク先の HTML に対しても同様に HTML 解析を行い，ページから WebAPI の概要情報のうち必要な部分のみを取得した．取得した WebAPI の概要情報を作成しておいた CSV ファイルに保存した．ProgrammableWeb に対するスクレイピングのシーケンスを図 3 に示す．

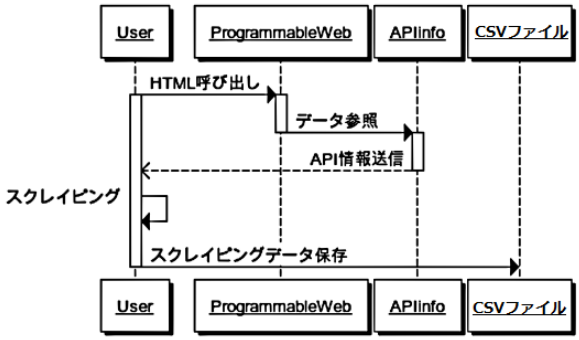

図 3　ProgrammableWeb に対する スクレイピングのシーケンス

(2)Google Suggest の利用：

ProgrammableWeb の概要情報を保存した CSV ファイルを読み込み，APIname を抽出する．抽出した APIname をキーワードとして Google Suggest へクエリを送信する．Google Suggest の出力形式は Java Script 形式，XML 形式，JSON 形式の 3 つから設定して出力することができるが，現在では XML 形式以外は無効となっており，エラーが返されるので，XML 形式で結果を取得する．simplexml_load_file 関数を使うことで，XML 形式のファイルをパースしてオブジェクトに代入することができ，自由に Suggest 機能の結果を扱うことができるようになる．

実際に WebAPI 名をキーワードとして Google Suggest へクエリを送信するだけでは，”～とは”，”～使い方” などの，その WebAPI 単体に関する Suggest 機能の結果しか得ることができない．従って，予め関連性を探索したい WebAPI を設定しておき，”WebAPI 名＋設定した WebAPI の頭文字” で Google Suggest へクエリを送信する．本研究では，ProgrammableWeb に登録されているマッシュアップに使われている WebAPI の上位 10 個を関連性探索のために設定する WebAPI とする．設定した WebAPI の例をあげると，Google Maps，Twitter などがあり，設定した WebAPI が Google Maps の場合は頭文字の”g” を追加して”WebAPI 名 + g”，Twitter[5]の場合は頭文字の”t” を追加して”WebAPI 名 + t” で Google Suggest へクエリを送信する．もし，使用頻度の低い WebAPI を関連性探索のために設定する WebAPI とすると，使用頻度の低い WebAPI 同士では関連性を提示することができても，そこから組み合わせるの可否を検索する作業において，実際のマッシュアップ例も少ない．この場合，インターネット上の情報も少なくなるので，関連性を提示した後のユーザの WebAPI 検索の負担が大きく，関連性の提示が WebAPI 検索支援につながりにくい．使用頻度の高い WebAPI を関連性探索のために設定するなら，使用頻度の低い WebAPI との組み合わせの場合でも，使用頻度の高い WebAPI はマッシュアップ例や，インターネット上の情報が多いため，関連性を提示した後のユーザの WebAPI 検索の負担が少ない．

(3)結果の出力：

取得した Suggest 機能の結果に対して strstr 関数を使用し文字列検索を行うことで，設定した WebAPI が Suggest 機能の結果の中にあるのかマッチングを行う．マッチングの結果，設定した WebAPI が見つかった場合，その時 Google Suggest へクエリとして送信した WebAPI の概要情報を，設定した WebAPI 毎にまとめた CSV ファイルへ出力していく．CSV ファイルへ出力する時，Suggest 機能の結果の上から何番目で設定した WebAPI が見つかったのかの情報も，Rank として同時に出力する．これにより，どの WebAPI がより関連性を持っているかの順位をつけることができる．Google Suggest へのクエリの送信から結果を取得するまでのフローチャートを図 4 に示す．

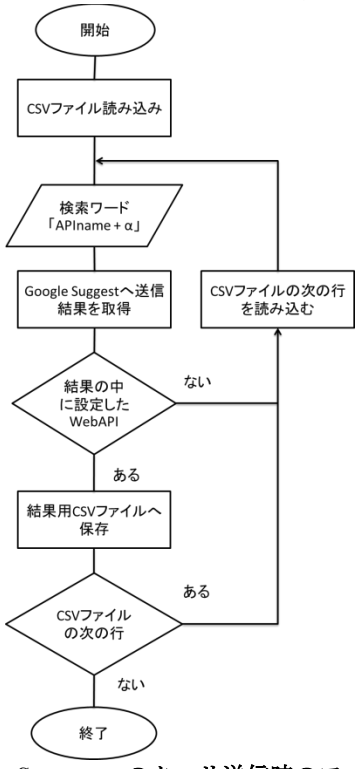

図 4　Google Suggest へのクエリ送信時のフローチャート

5．評価と考察

提案システムの評価を示すとともに，評価に対する考察を行う．ProgrammableWeb との比較評価と，提案手法で得られた関連性の検証を行った．

5.1　比較評価

提案手法を用いることで，情報範囲を拡大し，ProgrammableWeb に登録されていない WebAPI の組み合わせを探索することができているのかを検証した．ProgrammableWeb に登録されているマッシュアップに使われている上位 10 個の WebAPI を評価対象として設定し，設定した WebAPI 毎に 10 個ずつ，Suggest 機能の結果の中で取得した順位が上のものから順番に選択する．選択した WebAPI との組み合わせが ProgrammableWeb に登録されていない新しい組み合わせなのかを検証した．Google Maps を例に検証の結果を表 2 に示す．色付きで表示されている 7 個の WebAPI の組み合わせが，ProgrammabelWeb 上にない新しい組み合わせとなった．全体で見ると，検証を行った WebAPI の組み合わせ 100 個中 88 個が新しい組み合わせ

となり，情報範囲の拡大により ProgrammabelWeb に不足している WebAPI の組み合わせを取得することができた．

表 2　Google Maps との組み合わせ検証例

Rank	APIname	Category
1	Genome Maps	Science
1	Google Fusion Tables	Database
1	Nabaztag	Games
1	Sfax	Fax
1	Winnipeg Transit	Transportation
2	Brain Maps	Reference
2	EarthTools	Tools
2	IP Location	Internet
2	jsFiddle	Tools
2	The Old Bailey	Government

5.2　関連性の検証

ProgrammableWeb のマッシュアップ登録数と提案手法で得られた WebAPI の関連性の数を比較した結果，ProgrammableWeb より多くの関連性を得られた Twitter と Youtube[6]と Facebook[7]，得られた関連性の数が最も少ない Google Maps を対象として，得られた関連性で容易な WebAPI の組み合わせ検索を実現できているのかを検証した．検証の結果を表 3 に示す．この結果により，得られた関連性の数が最も少なかった Google Maps の検証と，得られた関連性の数が多かった Facebook の検証でマッシュアップを発見することができ，得られた WebAPI の関連性から WebAPI の組み合わせ検索を行うことができた．Twitter と Youtube の検証では，企業の Twitter アカウントや，Youtube に投稿されている動画などしか発見することができず，WebAPI の組み合わせ検索を行うことはできなかった．

表 3　関連性の検証結果

WebAPI	対象WebAPI	検索結果	組み合わせ検索の可否
Google Maps	nabaztag[8]	マッシュアップ nabaztag-map[9]	可能
Twitter	LookStat[10]	提供企業のTwitterアカウント	不可能
Youtube	Geocaching[11]	Webサービスの紹介動画	不可能
Facebook	FatSecret[12]	マッシュアップ My Diet[13]	可能

5.3　考察

既存手法と提案手法を比較したことにより，提案手法で情報範囲の拡大を行い，ProgrammableWeb には無い新しい WebAPI の組み合わせを探索することができていることを示せた．既存手法よりも提案手法で得られた WebAPI の関連性の数が少なかった理由は，”APIname + g” なら”～guide”，”APIname + f” なら”～free”，”APIname + t” なら”～tutorial” など，検索ワード単体の情報を詳しく検索するためのキーワードが多く，関連性のある WebAPI を数多く探索することができなかったことが原因だと考えられる．しかし，その中で得られる WebAPI の関連性はより精度の高いものとなった．

前節で行った検証により，提案手法で得られた WebAPI の関連性の数に関係なく，組み合わせることができる WebAPI の関連性を探索できていることが示せた．提案手法で得られた WebAPI の関連性の数に関係なく，WebAPI の組み合わせを探索できたことで，提案手法で得られる情報の精度が高さを示した．提案手法で得られた WebAPI の関連性が，全て組み合わせることができる WebAPI ではないことが示されたが，その原因は設定した WebAPI の汎用性にある．提案手法で既存手法より多くの WebAPI の関連性を探索できた Twitter，Youtube，Facebook の 3 つ WebAPI は，他の WebAPI と組み合わせる使い方だけでなく，他の WebAPI がより詳しい情報を提供する場として利用することができるという汎用性を持つ．そのため，他の WebAPI との関連性が高くなり，Suggest 機能の結果で多くの WebAPI との関連性を持つようになった．さらに，得られた関連性が多い WebAPI は，関連性の精度が低くなる可能性があることを示した．

6．まとめ

本研究では，マッシュアップにおける WebAPI の組み合わせ検索を容易に行える環境の実現を目的として，検索エンジンの Suggest 機能を利用して WebAPI 同士の関連性を探索することで，ユーザの WebAPI 検索を支援するシステムを提案した．既存の WebAPI 検索手法では，1 つの WebAPI についての情報を検索することができたが，他の WebAPI との関連性を示す情報を得ることができず，組み合わせる WebAPI を検索することができなかった．そこで既存の WebAPI 検索の問題を解決するために，既存の WebAPI 検索手法に不足している WebAPI の関連性を，検索エンジンの Suggest 機能で探索し取得することで補うシステムを提案した．

評価では，既存の WebAPI 検索手法と提案手法で得られる WebAPI の関連性の数を比較し，新しく探索することができた WebAPI の関連性の精度について検証した．提案手法は，既存の WebAPI 検索手法では得られない，新しい WebAPI の組み合わせを探索できていることと，得られる WebAPI の関連性の総数が少なくても新しい WebAPI の関連性を探索することができることを示した．また，探索した新しい WebAPI の関連性が，WebAPI の組み合わせ検索に利用できる精度の高いものであることを示すことができた．

参考文献

[1]　Programmable web: http://www.programmableweb.com/.
[2]　Google Maps: https://developers.google.com/maps/.
[3]　Google Suggest : http://google.com/complete/search?q=&output=toolbar
[4]　Apache : http://www.apache.jp/
[5]　Twitter : https://dev.twitter.com/docs
[6]　Youtube : https://developers.google.com/youtube/
[7]　Facebook : http://developers.facebook.com/
[8]　nabaztag : http://blog.karotz.com/?p=1545&lang=en
[9]　nabaztag-map : http://www.nabzone.com/nabaztag-map/map.php
[10]　LookStat : https://www.lookstat.com/api.html
[11]　Geocaching : http://www.geocaching.com/live/apidevelopers/
[12]　FatSecret : http://platform.fatsecret.com/api/
[13]　My Diet : http://www.fatsecret.com/connected/facebook