ユーザの嗜好と位置情報を利用した周辺の飲食店の検索と提案の検討

指導教員 渡辺 恭人
学籍番号 ０８４０１４６
八巻 優一

提出日：2012年1月25日

目次

１．背景・目的

１−１ 背景

日本の外食産業は経済成長とともに発展し、現在約６７万軒もの一般飲食店が存在するまでに成長した。そして、成長する過程で様々なニーズに対応するように分野を広げ、細分化していった。飲食店の評価を共有するサービスなどもあらわれ、外食産業の競争は過熱するばかりである。競争が過熱すると、顧客を取り込むための様々な工夫がなされ、消費者にとってはプラスになることが多い。しかし、その結果として飲食店の多様化が行き過ぎてしまい、消費者が店舗を選ぶ際の選択肢が増えすぎてしまった。

消費者がよりよい店舗を選ぶためには、より多くの店舗の情報を比較し、判断する必要がある。だが、消費者が飲食店を選ぶ際に、その都度情報を整理し選んでいては手間がかかりすぎてしまう。そこで有効活用したいのが口コミサイトのサービスである。口コミサイトとは、利用者が実際に商品や施設を利用した感想や評価を投稿し、多くの利用者と情報を共有することを目的としたサイトである。飲食店の口コミサイトでは、“食べログ”などが有名である。これらのサービスを活用することにより、簡単に評価の高い飲食店を選ぶことができる。

最近では、スマートフォンの普及により、口コミサイトと位置情報を組み合わせて利用したアプリケーションが多く開発されている。簡単に位置情報の取得や地図によるナビゲーションを行なえるスマートフォンならではの機能である。これらのアプリケーションのほとんどが、現在位置または指定された位置の周辺にある飲食店を検索する機能をもっている。詳細な検索の条件を設定することができるアプリケーションもあり、利用者の食の嗜好に合わせた検索も可能である。しかし、これらのアプリケーションは利用者が潜在的に抱えている需要に対応できていない場合が多い。

「利用者が潜在的に抱えている需要」とは、利用者の現在位置や検索を行う時間帯、利用者が持つ飲食物の嗜好など、利用者が検索を行う際に条件の入力をしていない需要や表面化していない需要である。たとえば、利用者が上記のアプリケーションで検索を行うとき、検索を行う時間や現在位置などの情報は利用者が入力しているわけではないが、検索結果として表示される店舗は開店中でなければならない。そうでなけ

れば、現在開店中の店舗を探す手間が増えてしまう。このように、利用者は入力を行わないが利用者にとって必要である需要が「利用者が潜在的に抱えている需要」である。この需要を満たすためには、利用者にかかわる情報を自動的に取得し、分析できるシステムが必要である。

これらの機能をもつサービスを提供するためには、利用者が常に身につけ持ち歩き、現在位置などの情報を簡単に取得することができるスマートフォン等の携帯端末を利用した形態が望ましい。本研究ではスマートフォン等を用いた、利用者が潜在的に抱えている需要を満たす飲食店の検索と提案の検討をおこなう。

１－２　目的

本研究では、利用者が潜在的に抱えている需要を満たす飲食店の検索機能を持つアプリケーションによって、利用者がより快適に飲食店を選択、利用できるようにすることを目的とする。

１－３　本論文の構成

本論文の２章では、飲食店と既存の飲食店検索サービス、口コミサービス等の分析について述べる。３章では２章での問題点を整理し、それらの解決へのアプローチと検索機能のモデルについて述べる。４章では、３章で述べたことをもとにアプリケーションの設計を行う。５章では４章での設計に基づいた実装について述べる。６章では実装したアプリケーションについて評価、考察を行なう。７章では本研究のまとめを述べる。

２．現状と問題点

本章では飲食店の現状と既存の飲食店の検索・評価サービスについて述べる。

２－１　飲食店の現状

総務省統計局による「平成 21 年経済センサス-基礎調査 事業所に関する集計」（図２－１－１）によると（参考文献【4】）、日本国内飲食店の店舗数は 673,458 軒であり、日本料理店・中華料理店などの専門料理店が最も多く割合を占めていることがわかる。最も店舗が集中している都道府県は東京都であり、単位面積あたりの飲食店数を比較すると東京都が 42 件と最も多く、件数が二桁の都道府県は東京都、大阪府、神奈川県の三つだけである（表２－１－２）。

飲食店といってもその種類は様々であり、人口が多い都心であるほど店舗数が多くなる傾向がある。そのため、店舗が密集している都心では自分の嗜好に合った店舗を見つけることが困難である。

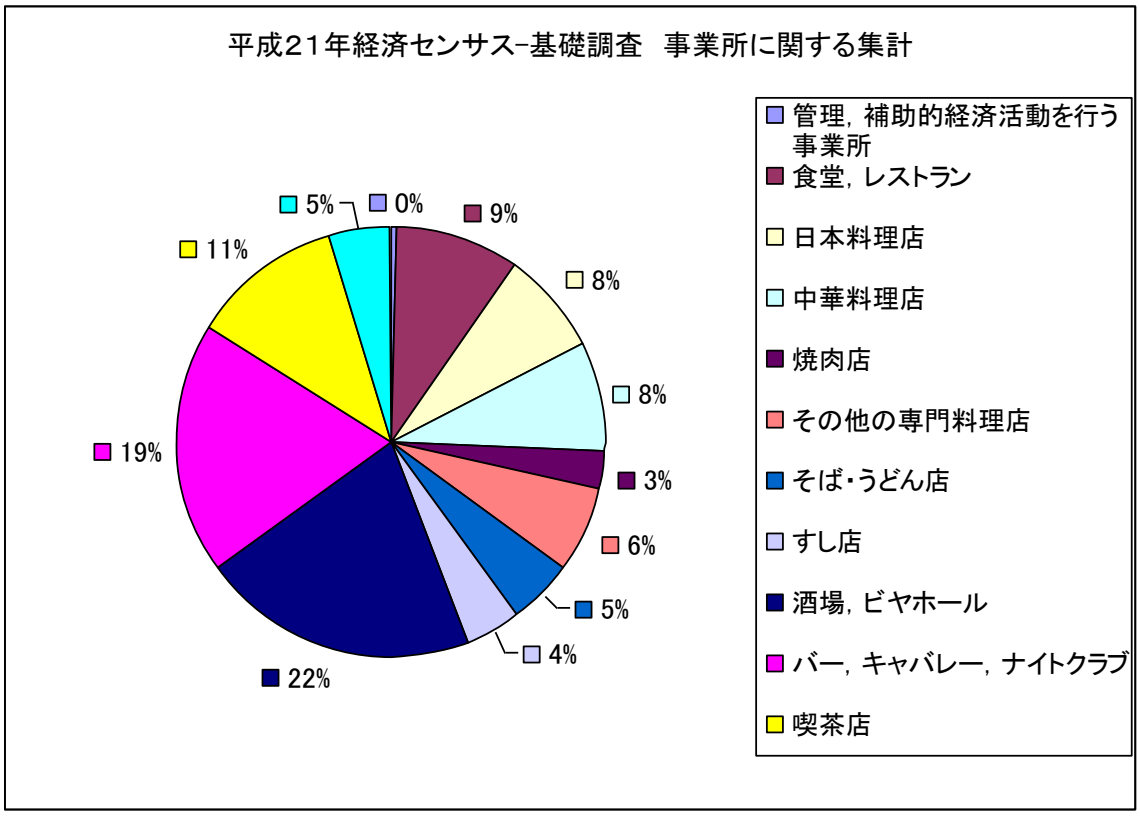


図２－１－１：平成 21 年経済センサス-基礎調査

事業所に関する集計・飲食店事業所数

表２－１－２：平成 21 年経済センサス-基礎調査

事業所に関する集計・都道府県別飲食店数

	飲食店数	１km²あたりの飲食店数		飲食店数	１km²あたりの飲食店数
東京都	89,243	42.44	青森県	8,244	0.85
大阪府	56,843	29.95	熊本県	8,009	1.13
愛知県	41,289	8.07	岡山県	7,945	1.13
神奈川県	37,035	15.33	愛媛県	7,351	1.29
北海道	31,620	0.38	長崎県	7,045	1.72
兵庫県	30,932	3.68	山口県	6,935	1.13
埼玉県	27,873	7.40	宮崎県	6,770	1.07
福岡県	26,568	5.48	岩手県	6,660	0.44
千葉県	23,995	4.72	石川県	6,532	1.56
静岡県	20,253	2.76	山形県	6,423	0.97
京都府	15,824	3.43	大分県	6,150	1.21
広島県	15,209	1.79	秋田県	5,661	0.49
茨城県	12,946	2.12	和歌山県	5,513	1.17
長野県	12,041	0.92	山梨県	5,415	1.29
新潟県	11,923	1.15	香川県	5,334	2.86
岐阜県	11,876	1.22	富山県	5,284	2.58
宮城県	10,876	1.58	高知県	5,169	0.73
沖縄県	10,825	4.76	滋賀県	4,953	1.31
栃木県	10,081	1.57	奈良県	4,680	1.27
群馬県	10,081	1.58	福井県	4,575	1.09
福島県	9,434	0.68	佐賀県	4,257	1.74
三重県	8,664	1.50	徳島県	4,229	1.02
鹿児島県	8,610	0.95	島根県	3,287	0.49
			鳥取県	2,996	0.85

２－２　飲食店の選び方

平成１９年に NTT レゾナント株式会社と株式会社三菱総合研究所が共同して実施した、インターネットアンケート・サービス「goo リサーチ」の第２回「外食の実態」に関する調査がある（参考文献【5】）。このアンケートは goo リサーチ登録モニターに向けて実施されたものであり、有効回答者数は 2201 名である。これによると、外食をする頻度は「週一回以上」が過半数を占めている。（図２－２－１）また、行きつけの店の有無では「行きつけの店がひとつ以上ある」が８割を超えている（図２－２－２）。そして、『特別な場面での外食』の際の店の選び方では「自分が過去にいったことのある店から選ぶ」が他の回答と大きく差をつけて一位になっている（図２－２－３）。これは、特別な場面では自分の経験から信頼のおける店舗を選びたいという心理があることがあるからである。特別な場面に限らず、実際に自身が経験したことはとても信頼度が高い情報であるといえる。行きつけの店というのも、一度店舗に行き経験したことが信頼できる情報となって「行きつけの店」になりうるのである。そもそも「行きつけの店」というものは、繰り返し行く店、何度も利用する店などを指しており、行きつけの店として利用している人にとっては評価がかなり高い店であるといえる。そういった店舗は他の店舗と差別化をした形式で記録できることが望ましい。

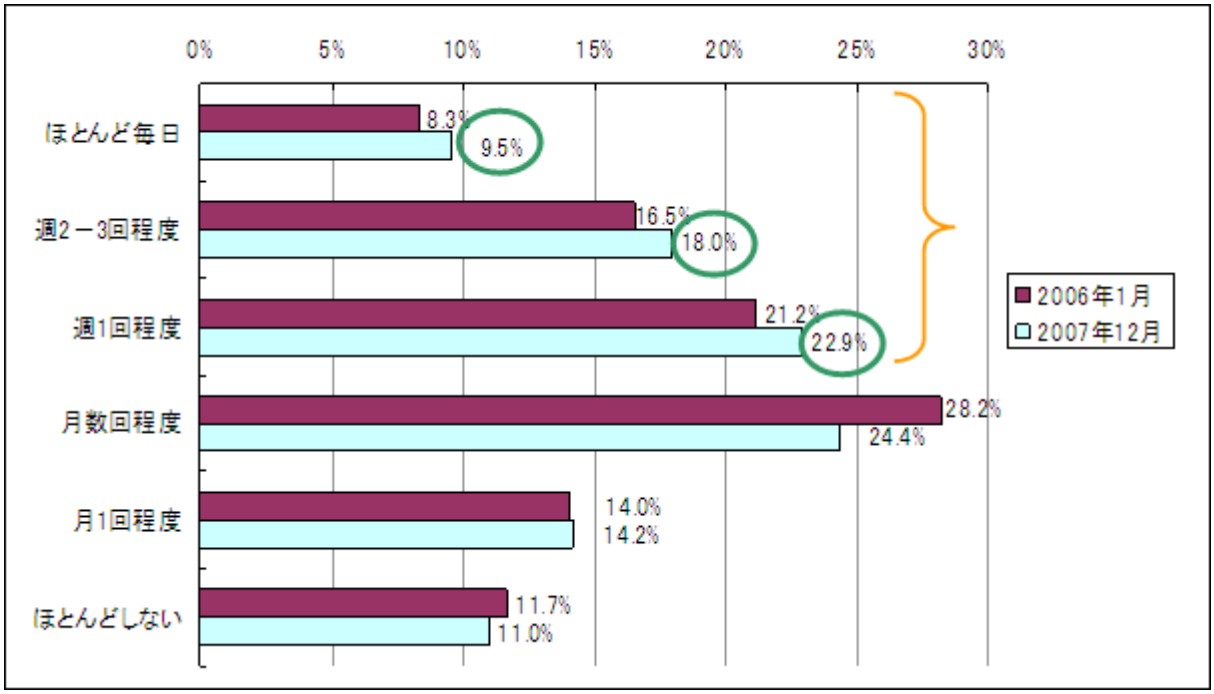


図２－２－１：第２回「外食の実態」に関する調査　外食の頻度

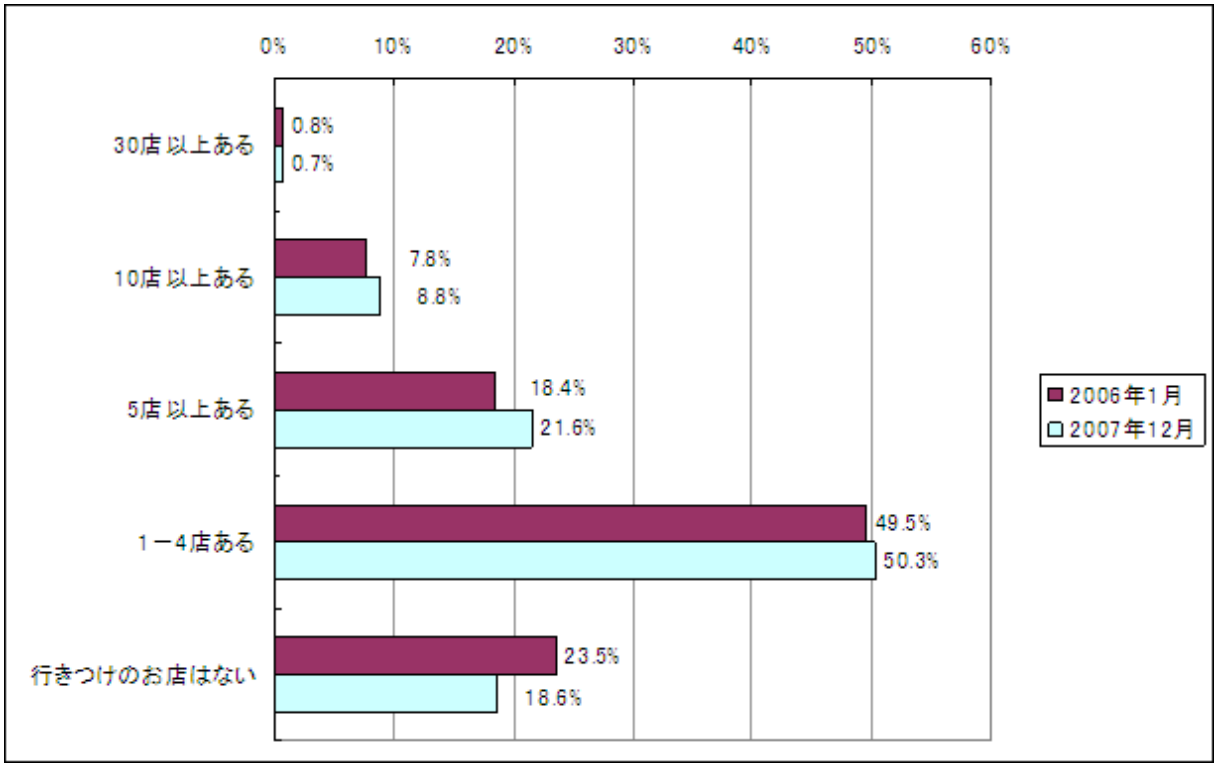


図２－２－２：第２回「外食の実態」に関する調査

行きつけの店の有無

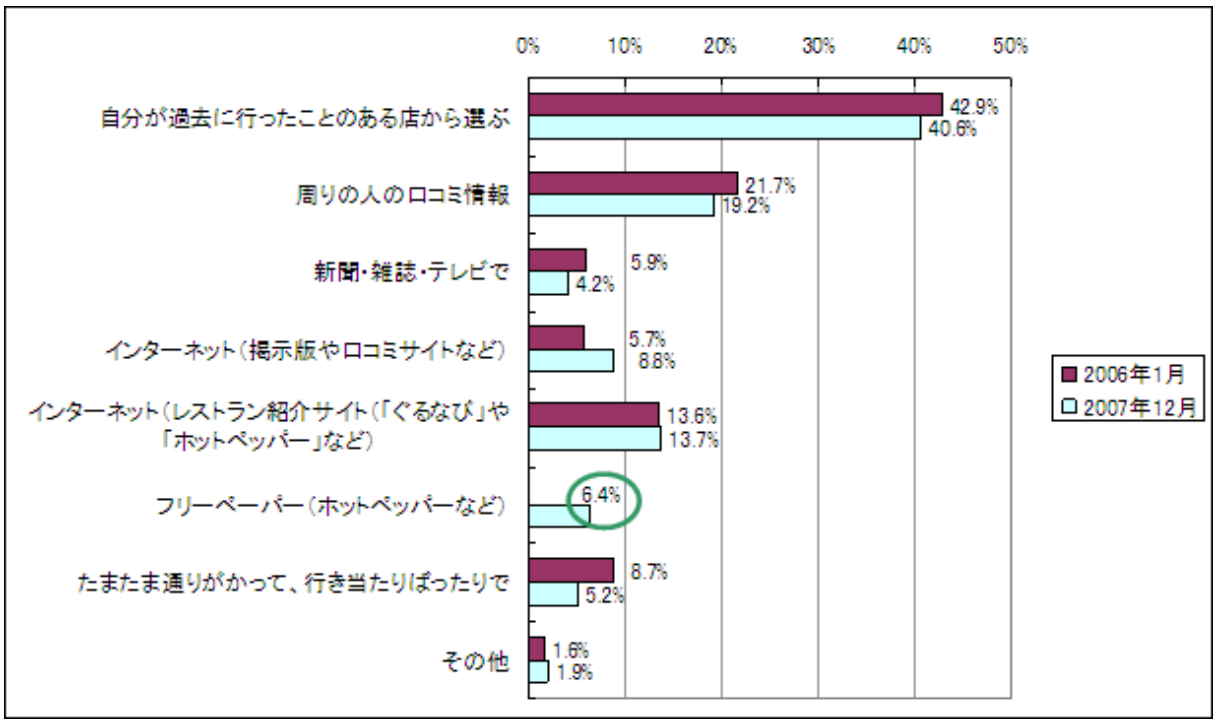


図２－２－３：第２回「外食の実態」に関する調査

「特別な場面での外食」の際の店の選び方

２－３　検索サービスおよび口コミサービスの現状

既存の飲食店検索アプリケーションと、その情報源となっている口コミサービスの分析を行なう。

２－３－１　食べログ（http://tabelog.com/）

食べログ（図２－３－１（１））は、口コミ情報を参考にして飲食店を検索できるサイトである（参考文献【6】）。サービスの特徴として、約 65 万件（2011 年 8 月現在）もの店舗情報を有している点と、口コミによる評価を点数化してランキングを作成している点があげられる。検索機能は、エリア・駅・住所などの情報と料理に関するキーワード（店名なども含む）、さらに予算も設定して検索することができる。最初から詳細な情報を入力しなくても、都道府県から徐々に範囲を狭めて検索することも出来る。また、口コミを投稿しているレビュアーがどんな店舗に行き口コミを投稿しているかを閲覧することも出来るので、自分と食の嗜好が似ているレビュアーを見つけることが出来れば自分に合っている店舗を見つけられる可能性もある。

食べログの評価システムは信頼性の高い情報を提供するために、点数化に独自のロジックを使用している。評価を行なうレビュアー自身が信頼できるかどうか、店舗自体が評価されている総数など、店舗の評価以外の情報も組み合わせて信頼できる情報を提供している。口コミ情報が参考になったら、そのロコミに対して投票をすることで情報の信頼性を客観的に見ることができる仕組みや、時間帯による評価、料理の味・サービス・雰囲気などといった項目別の点数、使用した金額の統計まで記録されているため、利用者はより詳細で信頼性のある情報を活用することが出来る。

図２－３－１（１）：食べログ店舗詳細画面

２－３－２　ぐるなび（http://www.gnavi.co.jp/）

ぐるなび（図２－３－２（１））は、飲食店情報のポータルサイトである（参考文献【7】）。特徴として、ロコミとは違い飲食店自身が情報を発信している点と、検索する飲食店をジャンル別に分類している点があげられる。ジャンルの分類は「大人のレストランガイド」「Woman（女性向け）」「ベビなび（ママ・ベビー）」などがある。それぞれのコンテンツは利用者に合わせた情報提供を実現している。例えば、「Woman（女性向け）」ではヘルシーな食事や美容に効果があるとされている食材を使用している店舗

などが取り上げられており、検索機能もコンテンツごとに独自の条件検索ができるようになっている。

図２－３－２（１）：ぐるなび　Woman のトップページ

２－３－３　グル探

グル探（図２-３-３（１））は Android 端末で利用できるアプリケーションである（参考文献【9】）。主な機能は、地図上に飲食店の情報を写真で表示するというものである。検索機能は、キーワードから検索、現在地から検索、駅から検索、地図から検索の４種類用意されている。また、地図上に表示するだけでなく検索結果の一覧をリスト表

示できるため、店舗を比較することがしやすい。検索する範囲や表示する件数を設定することも出来るが、料理のジャンルなどの指定はできないため、一度に大量の店舗が表示されると画面が見づらくなってしまう。店舗を「お気に入り」に登録することができ、登録した店舗を一覧で表示することが出来る。現在位置から選択した店舗へのルート案内も実装されている。ルート案内は徒歩、電車、車といった移動手段別に表示することができる。

図２-３-３（１）：グル探利用画面

２－３－４　食べ地図

食べ地図（図２－３－４（１））は Android 端末で利用できるアプリケーションである（参考文献【11】）。主な機能は評価の高い飲食店を検索して地図上に表示するというものである。評価の高い飲食店というのは、情報源である「食べログ」での評価が高いものを指している。評価の高い店舗のみを表示することにより、利用者は満足度の高い飲食店を選べる可能性がある。検索機能は現在地周辺から検索、地図から検索、

地名から検索がある。グル探と同様にリスト表示も可能である。また、店舗の詳細情報から食べログに投稿されたコメントを閲覧することもできる。

図２－３－４（１）：食べ地図利用画面

２－３－５　ぐるなび（アプリケーション）

ぐるなび（図２－３－５（１））は同名サイトの情報をスマートフォン端末で閲覧、利用しやすくしたアプリケーションである（参考文献【12】）。Android 版と iphone 版が公開されている。検索位置をエリア選択か現在地から選ぶことができ、検索したい店舗の料理のジャンルも選択することが出来る。個室あり、禁煙席ありなどの設備の条件や予算も条件として指定することが可能である。検索結果はまずリストで表示される。リスト画面では店舗の名称、ジャンル、写真や、端末のコンパス機能によりその店舗がある方角と現在地からの距離が表示される。店舗の詳細な地図を閲覧するためには、店舗の詳細画面からルート案内を選択する必要がある。なお、その際は別の地図アプリケーションが起動し、そのアプリケーション上でのルート案内が行なわれる。

図２－３－５（１）：ぐるなび利用画面

２－３－６　Retty（http://retty.me/home.php）

Retty（図２－３－６（１））は、SNS サービスの twitter や facebook と連携したコメントの投稿により、行った飲食店や行きたい飲食店を共有するサイトである（参考文献【13】）。SNS サービスとの連携を強く意識したサービスの特性上、Retty 単体でのアカウントを作成することができず、必ず twitter または facebook のアカウントを使用しなければならない。利用者は自分の嗜好と似た嗜好をもつ他の利用者をフォローすることで、その人が行った飲食店や行きたい飲食店をチェックすることが出来る。飲食店の情報画面から他の飲食店検索サービス（食べログ、ホットペッパー、ぐるなび）の情報画面へのリンクもあり、利用者が簡単に多くの情報を得られるようにしてある。利用者自身も行った飲食店の評価や感想を投稿することで、自身をフォローしている人たちに対して情報を発信することが出来る。行きたい飲食店として店舗を登録しておくと、エリア別に分類され、外出先などで飲食店を選びやすくなる。さらに、スマートフォン用のサイトやアプリケーションを利用すると、現在地周辺の行きたい飲食店をリストアップする機能もあり、そこから地図を参照することも出来る。

図２－３－６（１）：Retty 利用画面

２－４　既存サービスの比較分析

飲食店検索サービスにおける検索システムは位置情報から検索するものがほとんどで、ジャンルの絞込みや予算の設定などのより詳細な検索を行えるものもある。検索結果の表示については、地図上に複数の店舗の位置を表示するものや店舗の一覧をリスト表示するものがある。地図を使ったルート案内は、現在位置と店舗の位置との距離関係が把握しやすく、検索をした直後に店舗へ向うことができる。

それぞれの分析結果は以下の表２－４－１の通りだ。項目「地図で表示」では複数の店舗がひとつの地図上に表示され位置関係を比較できれば○、選択した店舗の位置のみ地図で表示されれば△とした。項目「お気に入り機能」は店舗をお気に入り登録またはそれと同様の処理が可能ならば△とした。Retty 以外は登録した店舗を一覧で表示する機能だけであり、検索範囲内に登録した店舗が含まれていても特別に表示されることはなかった。項目「検索機能」では詳細な条件検索が可能ならば○、位置情報のみによる検索や、キーワードのみによる検索ならば△とした。項目「履歴機能」では、店舗情報を何らかの履歴として残す機能があれば○としたが、今回の分析では該当するサービスは Retty だけであった。

２－２飲食店の選び方で述べたように、自身の経験による情報は信頼性の高い情報

である。しかしながら、履歴機能を実装しているサービスは少ない。お気に入り機能を活用すれば特別に気に入った店舗を記録することはできるが、多くの店舗を履歴として記録することができれば、利用者が意識的にお気に入り機能を使うことなく利用者にとって有益な情報を提供することが出来る。これらの機能を充実させることができれば、利用者の持つ潜在的な需要に対応できるサービスを提供できるだろう。

表２－４－１：既存サービスが提供する機能の分析

		店舗の情報	地図で表示	ルート案内	お気に入り機能	検索機能	履歴機能
サイト	食べログ	○	△	×	○	○	×
	ぐるなび	○	△	×	○	○	×
	Retty	○	△	×	○	△	○
アプリケーション	グル探	○	○	○	○	△	×
	食べ地図	○	○	×	×	△	×
	ぐるなび	○	△	○	○	○	×

３．解決法の検討

本章では 2 章で調査し判明した問題点を整理し、解決へのアプローチとアプリケーションのモデルについて述べる。

３－１　問題点の整理

飲食店情報の現状

―店舗の種類は様々であり、利用者が選ぶ際に時間がかかってしまう。

―都心では店舗が集中しており、どの店舗が自分の嗜好に合っているかの判断が難しい。

飲食店の選び方

―週一回以上外食をする人の数は半数を超えている。

―行きつけの店舗がある人の数は８割を超えている。

―自分自身の経験が最も信頼できる情報である。

検索サービスおよびロコミサービスの現状

―お気に入り機能は登録した店舗を一覧で表示するだけで、有効に活用していないサービスが多い。

―履歴機能が実装されているサービスは少ない。

―位置情報以外の条件は利用者が指定しない限り絞られないので、利用者は意識的に情報を入力しなければならない。

３－２　解決へのアプローチ

以上の問題点から解決方法を考察する。週一回以上外食をする人の数は半数を超えているが、飲食店の数と種類は非常に多く、自分の嗜好に合った店舗を探すには多くの情報を整理して判断する必要がある。また、自分自身の経験が最も信頼できる情報であるため、飲食店を選ぶ際には経験して得た情報を利用すべきである。しかし、個人が多くの飲食店の情報集めて整理するのは困難なので、ロコミサイトや飲食店検索サービスが利用されることが多い。

これらのサービスでは検索を行なう際に必要な入力事項が多く、利用者の趣味嗜好

に合わせようとするほど検索条件の設定が増えてしまう。そこで、検索や評価の履歴を保存し、自動的に分析し検索条件に組み込むことで、利用者が潜在的に抱えている需要を満たすことができる。

３－３　アプリケーションモデル

解決へのアプローチにて考察した結果をもとに、アプリケーションモデルについて述べる。

飲食店を検索する際の条件
―現在地情報
―料理のジャンル
―予算
―お気に入り店舗
―履歴により分析された嗜好

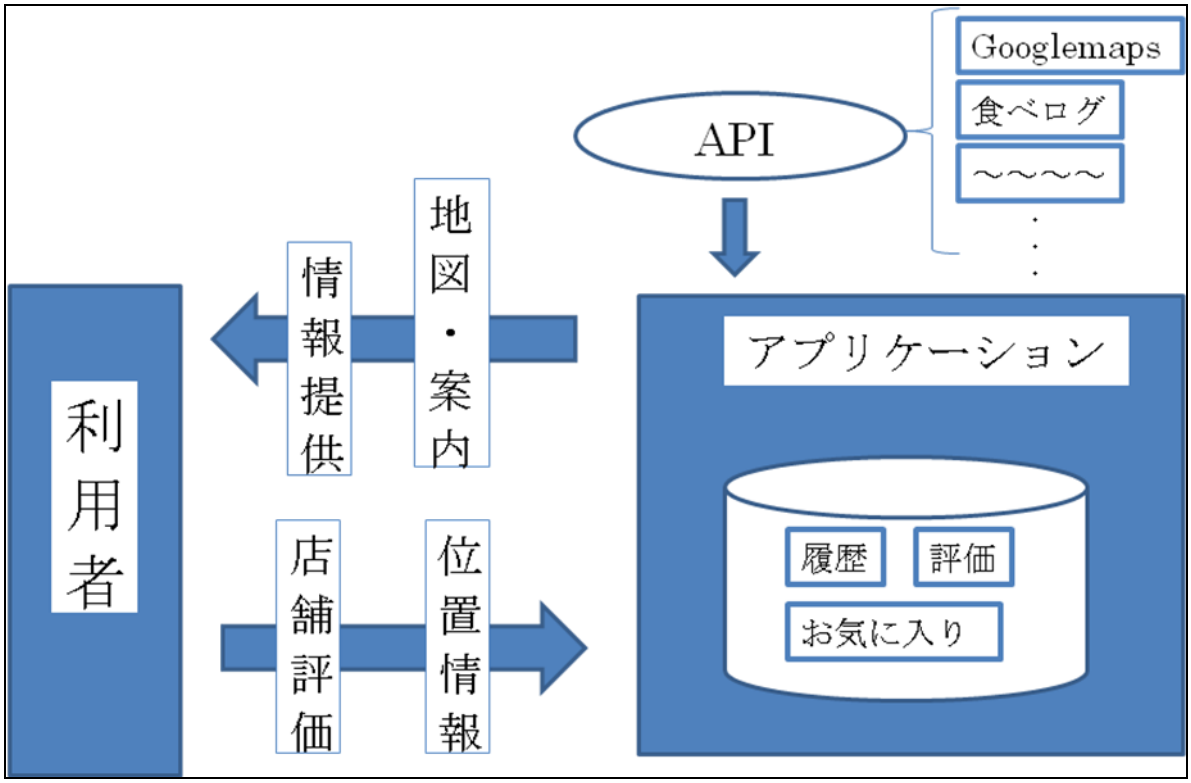
検索結果として提供される要素
―店舗情報
―ルート案内

記録される要素
―行った店舗の情報（ジャンル、回数、評価など）
―お気に入りの登録

「飲食店を検索する際の条件」は、現在地情報や履歴により分析された嗜好などの要素について、利用者が入力する項目をできるだけ少なくして、自動的に処理されることが望ましい。「検索結果として提供される要素」は、地図上に複数の店舗をアイコンで表示し、各アイコンをクリックすることで店舗の詳細情報を表示する。また、現在地から店舗までのルート案内を行う。「記録される要素」は、ルート案内を行うことでその店舗を利用したとみなし、利用した店舗として履歴に保存する。利用者が利用した店舗を評価し、記録する。特別に気に入った店舗であればお気に入りの登録を行

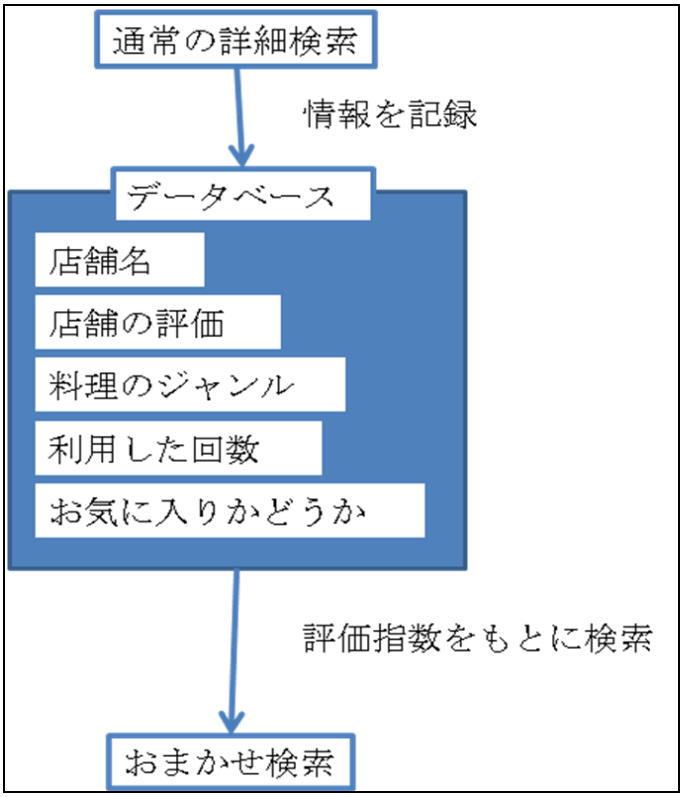
う。

アプリケーションモデルの利用者は、自身の履歴や評価により分析された情報をもとにした飲食店の検索結果を得ることができる。また、検索を行う際には入力する項目が少なく、詳細な検索であっても簡単に行うことができる。店舗の履歴や評価が増えていくにつれ、経験がより信頼できる情報源となり、利用者が潜在的に抱えている需要に対応できる（図３－３－１）。



（図 3－3－1：アプリケーションモデル）

データベースに記録された情報をもとに、利用者の嗜好を分析し、検索に利用する（図３－３－２）。アプリケーションを利用する際に、データベースへ入力される情報は“店舗名”“料理のジャンル”“店舗を利用した回数”“店舗の評価”“お気に入りかどうか”となる。利用者が何度か通常の検索を行うことにより、データを蓄積する。“店舗を利用した回数”と“店舗の評価”からその店舗の評価指数を導き出し、検索条件に評価指数の高いジャンルを組み込むことで、潜在的に抱えている需要を満たす形で検索を行う。また、検索範囲内にお気に入りに登録している店舗がある場合は、他の店舗とは違う形式で表示し、利用者が選びやすいようにする。


通常の詳細検索
情報を記録
データベース
店舗名
店舗の評価
料理のジャンル
利用した回数
お気に入りかどうか
評価指数をもとに検索
おまかせ検索

（図 3－3－2：検索モデル）

４．設計

本章では３章のアプリケーションモデルに基づき、構成機能と構成要素について考察する。

４－１　構成機能

アプリケーションモデルに基づき、アプリケーションの構成機能の設計を行った。以下に述べる（図 4－4－1)。

① 地図、ナビゲーション機能

Google が提供する WebAPI を利用し、表示する。

② 検索機能

食べログが提供する WebAPI を利用し、店舗を検索する。現在地周辺からの検索を行う場合、GPS 機能により現在地を割り出す。

③ 店舗情報

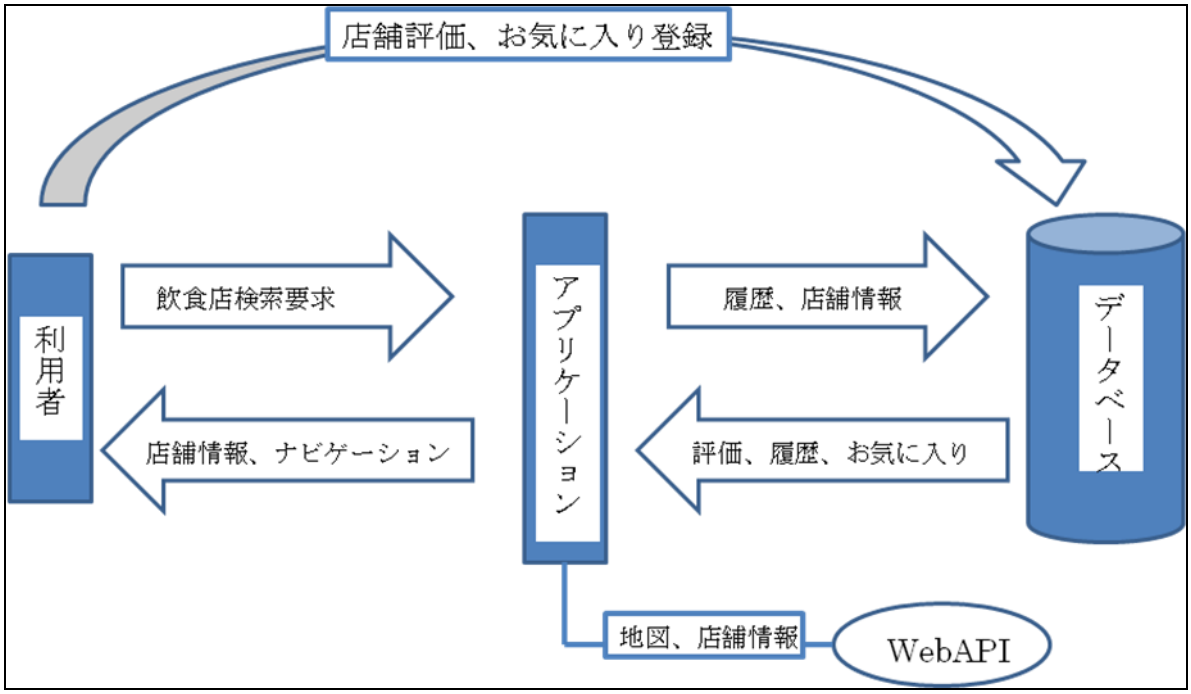
②と同じく、食べログが提供する WebAPI を利用する。

④ 履歴、お気に入り、評価の記録

SQLite を利用し、アプリケーション内にデータベースを作成する。店舗の情報や評価などを記録する。

⑤ 提案機能

データベースの情報を分析し、利用者の嗜好に合わせた提案を行う。



（図 4－1－1：構成機能）

４－２　構成要素

アプリケーションの構成要素を述べる。

① 利用者

利用者はアプリケーションを利用し飲食店を検索する。検索後、目的の店舗へのナビゲーションを利用し店舗へたどり着く。店舗を利用後、店舗の評価やお気に入り登録を行う。

② サービス提供

検索結果を地図上に表示し、店舗の詳細情報も提供する。店舗へのナビゲーションも行う。利用者が行った店舗の評価やお気に入りの登録などを記録する。蓄積されたデータから、利用者の嗜好に合わせた条件検索を行う。

③ アプリケーションの構成要素

―店舗の情報を検索、取得する WebAPI の受信機能

―地図と店舗情報を組み合わせて表示する機能

―データベースに評価や履歴を記録する機能

―記録から利用者の嗜好を分析する機能

４－３　アプリケーションの動作

アプリケーションを利用する際、各機能がどのように動作するのかを述べる。

①詳細検索

利用者が飲食店を検索する条件を自ら指定し、検索を行う機能。検索条件は以下の通りである。

・検索範囲

・ジャンル名

・点数（総合評価）

位置情報は GPS により現在地の緯度経度を指定する。飲食店の検索には「食べログ API」を利用するため、リクエストパラメータにジャンル名などを指定することができない。なので、ジャンル名営業時間か否かなどの絞込みはリターンパラメータのデータをもとに行う。

検索結果からレストラン ID、レストラン名、レストラン詳細ページ（PC）の URL、点数（総合評価)、ジャンル名、住所、電話番号、営業時間、休日、緯度、経度を取得し、緯度と経度をもとに地図上にアイコンを表示する。各アイコンを選択すると、詳細画面としてレストラン名、URL,ジャンル名、住所、電話番号、営業時間、休日が表示される。詳細画面から店舗へのナビゲーションを選択すると、現在地と選択された店舗の緯度経度が受け渡され、ナビゲーション画面へと移る。その際、データベースにレストラン ID、レストラン名、ジャンル名をデータベースに登録する。

②データベース

データベースのテーブルおよびフィールドは以下の通りである（表４－３－１）。

表４－３－１：店舗記録テーブル

INTEGER 型	TEXT 型	TEXT 型	INTEGER 型	INTEGER 型
レストラン ID	レストラン名	ジャンル名	利用者が つける点数	お気に入り

レストラン ID、レストラン名、ジャンル名はナビゲーション画面へ移った際に登録される。利用者がつける点数、お気に入りは店舗評価機能にて利用者が登録する。利用者がつける点数は五段階評価とする。登録されたデータからジャンル名と利用者がつける点数を抜き出し、各ジャンルごとに平均値を出すことでジャンルの評価指数とする。評価指数はおまかせ検索の際に利用する。

③おまかせ検索

ジャンルの評価指数を計算し、最も数値が高いジャンルを検索条件として飲食店検索を行う。リクエストパラメータは緯度経度以外のものはデフォルト値で送る。店舗の地図上への表示などは詳細検索と同様であるが、評価指数が最も高いジャンルの店舗のみが表示される。

④ナビゲーション機能

GoogleMaps の機能を利用し、ナビゲーションを行う。店舗の詳細画面からナビゲーションを利用する際、現在地と店舗の位置情報が受け渡され、二点間のルートをナビゲーションする。

⑤店舗評価機能

データベースに登録された情報をリスト表示し、利用者が店舗の評価を行う。評価は星１つから５つの五段階評価とし、テーブルには１～５の値が登録される。お気に入りか否かは二択で選択し、テーブルには true または false が登録される。

⑥お気に入り一覧機能

店舗評価機能により登録されたお気に入り店舗の一覧を表示する。その画面から直接お気に入りか否かを編集し、情報を更新する。

５．実装

４章で述べた設計に基づき行なった実装について述べる。

５－１　実装環境

・開発環境

PC　NEC LaVie LL750/E

OS　Microsoft Windows XP Home Edition version 2002 Service Pack 3

CPU　Intel(R) Celeron(R) M processor 1.50GHz

メモリ　750MB

・実装環境

Android 端末　NEC MEDIAS N-04C

OS　Android2.3.3

CPU　MSM7230 800MHz

メモリ　ROM 1024MB　RAM 512MB

・ソフトウェア環境

Eclipse IDE for Java Developers Version: Indigo Service Release 1

java version “1.6.0_29”

Java(TM) SE Runtime Environment (build 1.6.0_29-b11)

５－２　各機能の実装

実装したアプリケーションの各機能について述べる。

- Project 名　GourmetSearch
- パッケージ名　org.yy.test.gourmetsearch

５－２－１　メニュー画面

アプリケーションを起動したとき、初めに表示される画面。それぞれのボタンを押すことで各機能の動作画面へと移る（図５－２－１（１））。“おまかせ検索”をタップすると Search.java が実行され、“履歴確認・修正”をタップすると History.java が実行される。

・ プログラム　GourmetSearchActivity.java

```
package org.yy.test.gourmetsearch;


import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;



public class GourmetSearchActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ((Button)
findViewById(R.id.button1)).setOnClickListener(btn1Listener);
   //詳細検索用ボタンをbutton2に設定予定
        ((Button)
findViewById(R.id.button3)).setOnClickListener(btn3Listener);


    }


    OnClickListener btn1Listener=new View.OnClickListener() {


                @Override
                public void onClick(View v) {
                        Intent i =new
Intent(getApplicationContext(),Search.class);
```

```
                startActivity(i);
                }
        };

        OnClickListener btn3Listener=new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                        Intent i =new
Intent(getApplicationContext(),History.class);
                startActivity(i);
                }
        };
}
```

・ レイアウト　main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
   >
     <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/button2"
        android:layout_centerHorizontal="true"
        android:text="おまかせ検索" />
<Button
        android:id="@+id/button2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="詳細検索"
        />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button2"
        android:layout_centerHorizontal="true"
        android:text="履歴確認・修正" />
</RelativeLayout>
```

（図５－２－１（１）：メニュー画面）

５－２－２　検索機能

地図が表示されると同時に GPS により現在地の取得を開始する（図５－２－２（１））。現在地を取得し終えると、Overlay クラスを継承した UserIconsOverlay クラスを利用し現在地にアイコンを表示する。そして、取得した緯度・経度をもとに周囲の飲食店の検索を行なう。飲食店の検索には食べログ API を利用している。検索が終了すると同時に、各店舗の情報を ArrayList 配列に格納し、それらの情報をもとに地図上の飲食店がある地点にアイコンを表示する。店舗のアイコンの表示には ItemzedOverlay クラスを継承した TenpoIconOverlay クラスを利用する。アイコンをタップすると店舗名、料理のジャンルが表示される。そのとき、“登録”ボタンをタップすることでデータベースに情報を登録する。データベースの操作には SQLiteOpenHelper クラスを継承した MyDatebaseHelper クラスを利用する。

・ プログラム　Search.java

```
package org.yy.test.gourmetsearch;


import java.io.IOException;
import java.io.InputStream;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.ArrayList;
import java.util.List;


import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
```

```
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.Drawable;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Bundle;
import android.util.Log;
import android.util.Xml;

public class Search extends MapActivity implements LocationListener {
    private LocationManager mLocationManager;
    private MapController mMapController;
        private MapView mMapView;




        TenpoIconOverlay users_overlay;
    ArrayList<Integer> Rcd =new ArrayList<Integer>();
        ArrayList<String> RestaurantName=new ArrayList<String>();
        ArrayList<Float> tabeLatitude=new ArrayList<Float>();
```

```
        ArrayList<Float> tabeLongitude=new ArrayList<Float>();
        ArrayList<String> tabelogUrl=new ArrayList<String>();
        ArrayList<String> Category=new ArrayList<String>();

        Geocoder mGeocoder;



    @Override
    public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.search);

         mLocationManager = (LocationManager)
getSystemService(LOCATION_SERVICE);
         initMapSet();

    }
    private void initMapSet(){
         mMapView = (MapView)findViewById(R.id.map_view);
         mMapView.setBuiltInZoomControls(true);
         mMapView.setClickable(true);

         mMapController = mMapView.getController();
         mMapController.setZoom(16);
        }

    @Override
    protected void onResume() {
         if (mLocationManager != null) {
```

```
            mLocationManager.requestLocationUpdates(
                LocationManager.GPS_PROVIDER,
                0,
                0,
                this);
        }
        super.onResume();
    }

    @Override
    protected void onPause() {
        if (mLocationManager != null) {
            mLocationManager.removeUpdates(this);
        }
        super.onPause();
    }



//位置情報を取得したら実行するメソッド
    @Override
    public void onLocationChanged(Location location) {
       //現在地を中心に表示
       int gplat =(int)(location.getLatitude()*1e6);
       int gplon =(int)(location.getLongitude()*1e6);
       GeoPoint gp=new GeoPoint(gplat,gplon);
       mMapController.animateTo(gp);

    //一度だけで終わらせるために位置情報の更新を終了
       mLocationManager.removeUpdates(this);
```

```
        //アイコン表示
        Bitmap bmp = BitmapFactory.decodeResource(getResources(),
android.R.drawable.ic_menu_myplaces);
        UserIconsOverlay overlay = new UserIconsOverlay(bmp, gp);
        List<Overlay> list = mMapView.getOverlays();
        list.add(overlay);

        Drawable drawable_icon =
this.getResources().getDrawable(R.drawable.ic_launcher);
        TenpoIconOverlay users_overlay = new TenpoIconOverlay(drawable_icon,
this);
        list.add(users_overlay);



        //現在地から検索するメソッド呼び出し
        kensaku(gp);




        for(int i=0;i<Rcd.size();i++){
                int tabelat=(int)(tabeLatitude.get(i).floatValue()*1E6);
                int tabelon=(int)(tabeLongitude.get(i).floatValue()*1E6);
                GeoPoint tabegeo=new GeoPoint(tabelat,tabelon);
                String tabename=RestaurantName.get(i);
                String tabecategory=Category.get(i);

                users_overlay.drawicons(tabegeo,tabename,tabecategory);

        }
```

```
}



public void kensaku(GeoPoint gp){
    float latitude_str=(float) (gp.getLatitudeE6()/1e6);
    float longitude_str=(float) (gp.getLongitudeE6()/1e6);

    String uri = "http://api.tabelog.com/Ver2.1/RestaurantSearch/?"
                    +"Latitude="+latitude_str+"&"
                    +"Longitude="+longitude_str+"&"
                    +"SearchRange=large"+"&"
                    +"ResultSet=large"+"&"
                    +"Key=6a04509766533d28661dd05442a4f28fbe1b4aeb";

     //httpクライアントの設定
     HttpClient client = new DefaultHttpClient();
     HttpGet get = new HttpGet();

     try{
         get.setURI(new URI(uri));
         HttpResponse res = client.execute(get);
         InputStream in = res.getEntity().getContent();

         //パーサーの設定
         XmlPullParser parser = Xml.newPullParser();
         parser.setInput(in, "UTF-8");

         for(int eventType = parser.getEventType();eventType !=
```

```
                        XmlPullParser.END_DOCUMENT; eventType =
parser.next()){

                        String tagName;
                        String tagText;

                        if(eventType == XmlPullParser.START_TAG){

                        tagName = parser.getName();
                        Log.d("tag",tagName);
                        //各要素をArrayListに格納
                        if(tagName=="Rcd"){
                                Rcd.add(Integer.valueOf(parser.nextText()));

                        } else if(tagName=="RestaurantName"){

                                RestaurantName.add(parser.nextText());

                        }else if(tagName=="Latitude"){

tabeLatitude.add(Float.valueOf(parser.nextText()));
                        }else if(tagName=="Longitude"){

tabeLongitude.add(Float.valueOf(parser.nextText()));
                        }else if(tagName=="TabelogUrl"){
                                tabelogUrl.add(parser.nextText());
                        }else if(tagName=="Category"){
                                Category.add(parser.nextText());
                        }
```

```
                    parser.next();

                    if(parser.getEventType() == XmlPullParser.TEXT){
                    tagText = parser.getText();
                    Log.d("tag text",tagText);
                }
            }

        }

    }
    catch (URISyntaxException e) {
        e.printStackTrace();
    }
    catch (ClientProtocolException e) {
        e.printStackTrace();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    catch (XmlPullParserException e) {
        e.printStackTrace();
    }
 }

@Override
public void onProviderDisabled(String provider) {

}
```

```
    @Override
    public void onProviderEnabled(String provider) {


    }


    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        switch (status) {
        case LocationProvider.AVAILABLE:
            Log.v("Status", "AVAILABLE");
            break;
        case LocationProvider.OUT_OF_SERVICE:
            Log.v("Status", "OUT_OF_SERVICE");
            break;
        case LocationProvider.TEMPORARILY_UNAVAILABLE:
            Log.v("Status", "TEMPORARILY_UNAVAILABLE");
            break;
        }
    }


       @Override
       protected boolean isRouteDisplayed() {
              return false;
       }
}
```

・ プログラム　UserIconsOverlay.java

```
package org.yy.test.gourmetsearch;
```

```
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Point;
import com.google.android.maps.*;

public class UserIconsOverlay extends Overlay {
 private final Bitmap bmp;
 private final GeoPoint gpoint;

 public UserIconsOverlay(Bitmap bmp, GeoPoint gp) {
   this.bmp = bmp;
   this.gpoint = gp;
 }

 public void draw(Canvas canvas, MapView mapView, boolean shadow) {
   Projection pro = mapView.getProjection();
   Point p = pro.toPixels(gpoint, null);
   canvas.drawBitmap(bmp, p.x, p.y, null);
 }
}
```

・ プログラム　TenpoIconOverlay.java

```
package org.yy.test.gourmetsearch;

import java.util.ArrayList;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.drawable.Drawable;
```

```
import com.google.android.maps.GeoPoint;
import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.OverlayItem;

public class TenpoIconOverlay extends ItemizedOverlay<OverlayItem> {

  private ArrayList<OverlayItem> overlay_items = new ArrayList<OverlayItem>();
  private Context context;



  public TenpoIconOverlay(Drawable defaultMarker, Context context) {
      super(boundCenterBottom(defaultMarker));
      this.context = context;
      populate();
  }

  @Override
  protected OverlayItem createItem(int i) {
    return overlay_items.get(i);
  }

  @Override
  public int size() {
    return overlay_items.size();
  }

  // アイテムの追加
  public void addOverlayItem(OverlayItem overlayItem) {
    overlay_items.add(overlayItem);
    populate();
```

```
}

public void drawicons( GeoPoint g ,String name,String category)
{
  int lat = g.getLatitudeE6();
  int lng = g.getLongitudeE6();

  addOverlayItem( new OverlayItem(
    new GeoPoint(lat , lng ),
    name,
    category
  ) );
}



// アイコンタップ時
@Override
protected boolean onTap(int index) {

  final OverlayItem item = overlay_items.get(index);

  AlertDialog.Builder dialog = new AlertDialog.Builder(context);
  dialog.setTitle(item.getTitle());
  dialog.setMessage(item.getSnippet());
  dialog.setPositiveButton("登録", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {

        MyDatebaseHelper helper = new MyDatebaseHelper(context);
          SQLiteDatabase db = helper.getWritableDatabase();
          String tenponame="";
```

```
            String tenpocategory="";
            tenponame="'"+item.getTitle().replaceAll("  ", "")+"'";
            tenpocategory="'"+item.getSnippet()+"'";
            String sql="insert into tenpo(tenponame,category,rate) values (" +
                        tenponame+","+
                        tenpocategory+","+
                        "0);";

            db.execSQL(sql);
      }
    });
    dialog.setNegativeButton("キャンセル", new
DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
      }
    });
    dialog.show();
    return true;
  }
}
```

・ プログラム　MyDatebaseHelper.java

```
package org.yy.test.gourmetsearch;

import android.content.Context;

import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
```

```
public class MyDatebaseHelper extends SQLiteOpenHelper {
    public MyDatebaseHelper(Context c) {
        super(c, "tenpo", null, 1);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // table create
        db.execSQL(
            "create table tenpo("+
            " No integer primary key autoincrement,"+
            "   tenponame text,"+
            "   category text,"+
            "  rate text"+
            ");"
        );
    }
}
```

・ レイアウト　search.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
```

```
    <com.google.android.maps.MapView
        android:id="@+id/map_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:apiKey="01NJfB6tTCXt14n_fmqXoZop-LnuOzj4gCKtCOg"
        />
</LinearLayout>
```

（図５－２－２（１）：検索結果画面）

５－２－３　履歴確認機能

データベースに登録された飲食店情報をリスト表示する。登録される情報は、登録された順番に振られるインデックス、店舗名、店舗のジャンル、評価である。今回の実装では店舗の評価機能まで至らなかったので、データを格納するフィールドの作成のみとなっている。データベースの操作は MyDatebaseHelper を利用している。

・ プログラム　History.java

```
package org.yy.test.gourmetsearch;

import org.yy.test.gourmetsearch.R.id;

import android.app.Activity;
import android.os.Bundle;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class History extends Activity {



    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.history);

        MyDatebaseHelper helper = new MyDatebaseHelper(this);
        SQLiteDatabase db = helper.getReadableDatabase();

       Cursor c = db.query("tenpo", new String[] { "No","tenponame",
"category" ,"rate"},
                null, null, null, null, null);

        boolean isEof = c.moveToFirst();

        ArrayAdapter tenpoarray=new ArrayAdapter(this,
android.R.layout.simple_list_item_1);
```

```
        while (isEof) {
            tenpoarray.add(c.getInt(0)+":"+c.getString(1)+"¥n"
+c.getString(2)+"¥n"+c.getString(3)+"¥n");
            isEof = c.moveToNext();

        }
        ListView listview =(ListView)findViewById(id.listview);
        listview.setAdapter(tenpoarray);
        c.close();
        db.close();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }


}
```

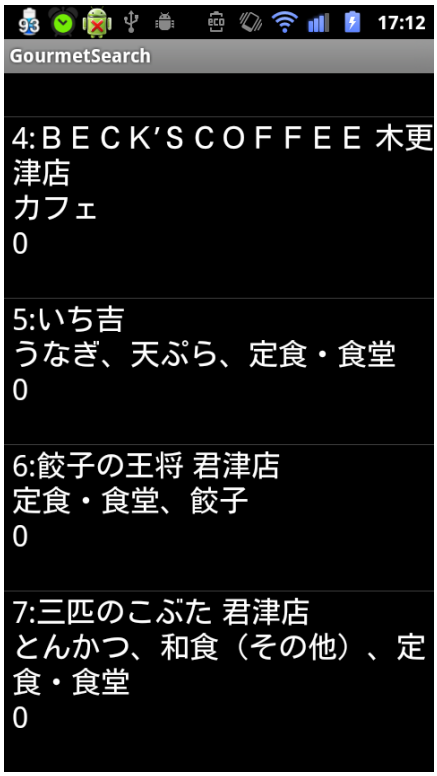
・ レイアウト　history.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

  <ListView android:id="@+id/listview"
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content" />


</LinearLayout>
```



（図５－２－３（１）：履歴確認画面）

５－３　実装結果

今回の実装では、4 章で行なった設計で述べている機能の一部を実装することが出来た。４－１節で述べた構成機能をもとに、実装した機能と未実装の機能を述べる。

―地図、ナビゲーション機能

地図上に現在地、周辺の飲食店を表示する機能は実装したが、ナビゲーション機能は未実装である。

―検索機能

詳細な条件での検索は未実装だが、現在地から周辺の飲食店を検索する機能は実装した。

―店舗情報

食べログ API を利用し店舗の情報を得る機能を実装した。

―履歴、お気に入り、評価の記録

履歴をデータベースに保存する機能は実装したが、お気に入り、評価などの情報の記録は未実装である。

―提案機能

未実装である。

６．評価と考察

この章では、実装したアプリケーションの評価について述べる。

６－１　動作確認

実装したアプリケーションの動作を述べる（図６－１－１）。実際に利用者が店舗を検索する際に本アプリケーションを利用する場合を想定して動作の説明を述べる。

① メニュー画面の表示

Android 端末にインストールされた本アプリケーションのアイコンをタップすると、アプリケーションのメニュー画面が最初に表示される（図６－１－１）。この画面から目的に応じ、“おまかせ検索”“履歴確認・修正”のボタンをタップするとそれぞれの画面が表示される。ここでは店舗の検索を行うため“おまかせ検索”のボタンをタップする。

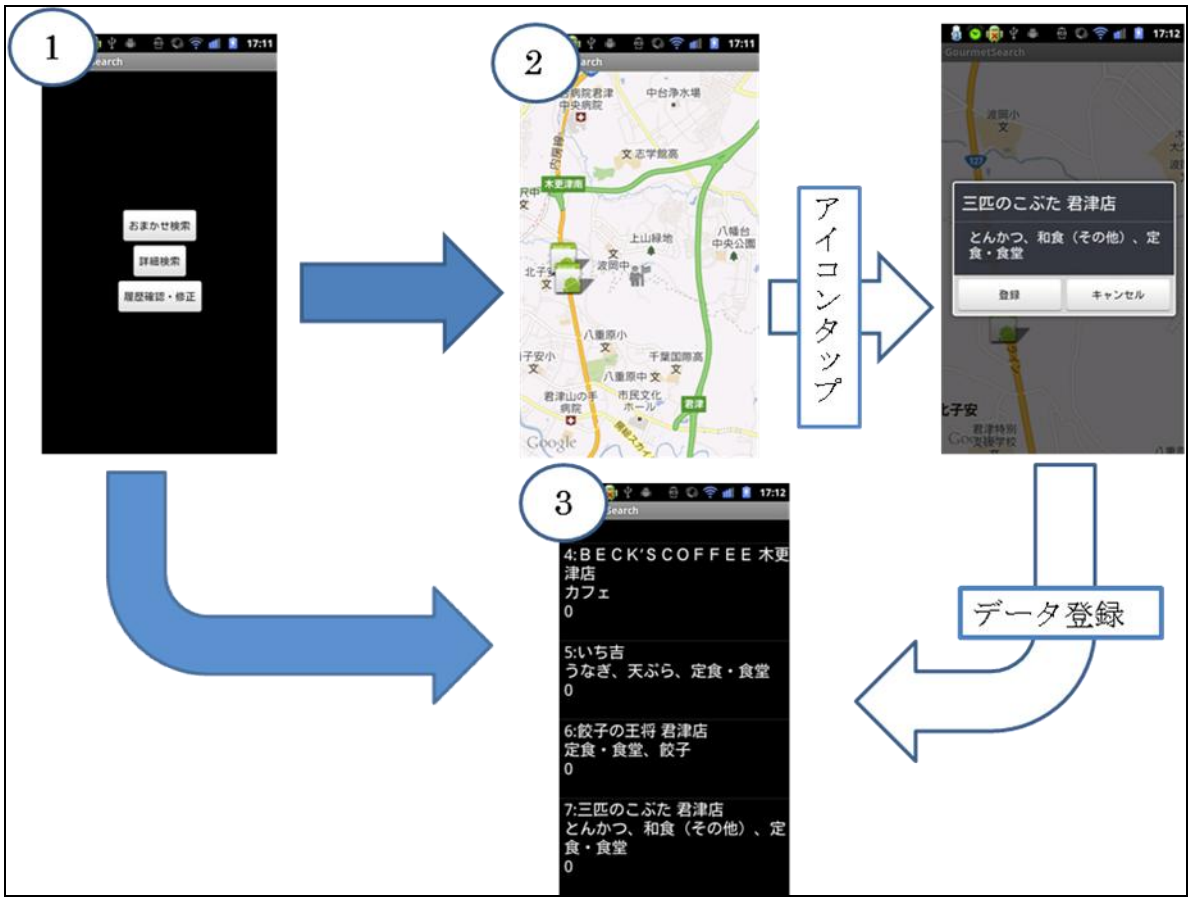
② 飲食店検索、地図表示

“おまかせ検索”のボタンをタップすると、地図が表示された画面へと移る（図６－１－２）。それと同時に GPS 機能が動作し、現在地を割り出す。現在地の割り出しが終了すると同時に現在地情報を食べログ API に送信し、リターンパラメータとして周囲の飲食店情報を得る。タップすると各店舗の情報を閲覧できるアイコンが表示されるので、利用者はそれぞれのアイコンをタップし、店舗の情報を確認する（図６－１－３）。閲覧できる店舗の情報は“店舗名”“料理のジャンル”である。その店舗を履歴として残したいのであれば“登録”ボタンを、そうでないのなら“キャンセル”ボタンをタップすることで、履歴として残したい店舗はデータベースに登録される。いずれかをタップすると、店舗のアイコンをタップする前の状態に戻る。利用者は複数表示される店舗の位置や料理のジャンルを確認することができ、履歴に残したい店舗を保存することができる。

③ 履歴の確認

履歴の確認を行うためにはメニュー画面で“履歴確認・修正”ボタンをタップをし、登録された店舗情報の一覧を表示する（図６－１－４）。各店舗ごとに“店舗名”“料

理のジャンル”“店舗の評価”が表示される。“店舗の評価”は編集する機能を実装していないため、“０”となっている。利用者は登録した店舗の情報を確認することができる。



（図６－１－１：動作の流れ）

６－２　機能評価

実装された機能の評価を述べる。設計で述べた機能と実装された機能を比較した（表６－２－１）。“地図、ナビゲーション機能”については、地図の表示と現在地と店舗のアイコンを同時表示できるので△とした。“検索機能”については、現在地周辺の飲食店を検索する機能を実装しているが、詳細な検索機能は未実装なので△とした。“店舗情報”については、情報の取得する機能は実装したが、その全てを表示する機能を実装していないので△とした。“履歴、お気に入り、評価の記録”については、店舗の名前と料理のジャンルのみ記録できるので△とした。“提案機能”については、未実装なので×とした。今回はおおまかな情報のやり取りを実現することで、設計通りの動

作が可能かどうかの指標を示すことができた。

表６－２－１：実装の評価

	地図、 ナビゲーション機能	検索機能	店舗情報	履歴、 お気に入り、 評価の記録	提案機能
実装	△	△	△	△	×

６－３　既存サービスとの比較評価

２－３節で述べた既存の飲食店検索サービスとの比較評価を行なった（表６－３－１）。評価の項目と基準は２－４節での評価と同じである。また、評価の対象は設計で述べた機能である。店舗の情報は、食べログ API から店舗名、位置情報、開店時間、定休日、料理のジャンルなどの複数の情報を得ることができるため○とした。地図での表示は、GoogleMap 上に利用する端末の現在位置と検索条件に合致した店舗のアイコンを複数表示するので○とした。ルート案内は、選択した店舗と現在位置の情報を利用し、GoogleMap のナビゲート機能を利用することで実現するので○とした。お気に入り機能は、端末内にデータベースを作成し、そこに記録させることで実現するので○とした。検索機能は、食べログ API から得られたリターンパラメータをもとに店舗を絞り込み、詳細検索を行うので○とした。履歴機能は、端末内データベースに行った店舗を登録していき、利用者が直接店舗の評価を行い、その評価をもとに利用者が潜在的に抱えている需要を満たす検索を行うので○とした。既存のサービスで最も実現されていなかった履歴機能を充実させることにより、信頼のある経験の情報を利用した検索を行うことができ、利用者が潜在的に抱えている需要を満たすサービスになったといえる。

表６－１－１：既存サービスと本研究の比較

		店舗の情報	地図で表示	ルート案内	お気に入り機能	検索機能	履歴機能
サイト	食べログ	○	△	×	○	○	×
	ぐるなび	○	△	×	○	○	×
	Retty	○	△	×	○	△	○
アプリケーション	グル探	○	○	○	○	△	×
	食べ地図	○	○	×	×	△	×
	ぐるなび	○	△	○	○	○	×
本研究	設計	○	○	○	○	○	○

６－４　考察

本研究の目的は、利用者が潜在的に抱えている需要を満たす飲食店の検索機能を持つアプリケーションによって、利用者がより快適に飲食店を選択、利用できるようにすることである。実装されたアプリケーションから、設計通りのアプリケーションの作成は十分可能であると判断できる。店舗を評価する機能と、その評価を検索機能に反映する機能を実装することで、利用者は検索を行うたびに詳細な条件を入力することなく嗜好に合った店舗を検索することが出来る。６－３節での評価の通り、設計したアプリケーションは利用者が潜在的に抱えている需要を満たし、快適に飲食店を検索することができるといえる。

７．まとめと課題

７－１　まとめ

１章では、数多くの飲食店の中から自身に合った店舗を見つけ出すことが容易ではないという背景のもと、利用者が快適に飲食店を検索できるアプリケーションの作成を目的とし、アプリケーションの設計と実装を目指した。2 章では、飲食店と飲食店検索サービスの現状と問題について述べた。飲食店の数と種類の多さから利用者は店舗を選びづらいという問題点や、検索サービスの比較分析を行った。3 章では、2 章での問題点をまとめて解決法の検討を行い、アプリケーションモデルを作成した。4 章では、アプリケーションの設計を行った。5 章では設計した機能の一部分を実装した。6 章では実装したアプリケーションを評価し、設計を評価した。また、本研究の考察を述べた。

７－２　今後の課題

本研究の課題として挙げられる項目を以下に箇条書きする。

① 設計に基づいた実装の完了
② 実装されたアプリケーションの評価
③ 評価に基づいた実装、設計の見直し

①については、4 章での設計をもとに 5 章の実装を完了させる。java や Android のプログラミングの知識を深め、設計で述べた機能を全て実装したアプリケーションを作成する。②については、実装が完了した後、改めてアプリケーションの機能や動作の確認、設計どおりにできたか確認、既存アプリケーションとの比較などを行い評価をする。③については、評価した内容をもとに不足している機能や不十分な機能を見直し、必要があれば設計の段階から修正を行う。これらの課題を通して、利用者がより快適に店舗を選択できるアプリケーションの作成を目指す。

参考文献

【1】赤坂玲音　『読本 Java』　㈱毎日コミュニケーションズ、2003 年

【2】安生真、柴田文彦、藤枝崇文
『初歩からわかる Android 最新プログラミング』
株式会社インプレスジャパン、2010 年

【3】郷田まり子、宅間俊志、近藤昭雄
『ジオモバイルプログラミング　-iPhone&Android で位置情報アプリを作ろう！-』
株式会社ワークスコーポレーション　2011 年

【4】統計局ホームページ 平成２１年経済センサス-基礎調査
http://www.stat.go.jp/data/e-census/2009/index.htm

【5】goo リサーチ 第二回「外食の実態」に関する調査
http://research.goo.ne.jp/database/data/000812/

【6】食べログ
http://tabelog.com/

【7】ぐるなび
http://www.gnavi.co.jp/

【8】週刊アスキープラス　料理は見た目も重要です！「グル探」
http://weekly.ascii.jp/elem/000/000/035/35646/

【9】Android マーケット グル探
https://market.android.com/details?id=jp.aquabox.goumet

【10】アンドロイダー　食べ地図
http://androider.jp/a/00a971720ec771c4/

【11】Android マーケット　食べ地図 Free
https://market.android.com/details?id=org.cuspy.android.tabelog2

【12】Android マーケット　ぐるなび
https://market.android.com/details?id=jp.co.gnavi.activity

【13】Retty
http://retty.me/home.php

【14】Android wiki*　Google Map に Drawable を配置する
http://wikiwiki.jp/android/?Google%20Map%A4%CBDrawable%A4%F2%C7%DB%C3%D6%A4%B9%A4%EB

【15】HatenaDiary　主に言語とシステム開発に関して
http://d.hatena.ne.jp/language_and_engineering/20110907/p1

【16】mucchin の Android 戦記　Android アプリで別のアクティビティを起動させる方法
http://android.roof-balcony.com/intent/intent/

【17】電脳羊（Android Dream）　Android WebAPI からデータを XML で取得して解析
http://xiangcai.at.webry.info/200907/article_41.html

【18】iPentac　[Java]Android で SQLite を利用する
http://www.ipentec.com/document/document.aspx?page=android-use-sqlite-simple-app

謝辞

本研究にあたり、渡辺恭人准教授に多くの手助けをしていただきました。本当にありがとうございました。文章の校正や論文全体の助言、プログラミングにあたっての参考書を貸していただくなど、非常に助かりました。実装段階で行き詰ったとき、励ましの言葉をいただいたので、諦めることなく本研究を終わらせることができました。この経験をもとに、日々精進していきたいと思います。最後にもう一度感謝を述べさせていただきます。ありがとうございました。