

バス停時刻表検索システム用データの更新方法の検討

島根大学 総合理工学部 数理・情報システム学科

計算機科学講座 田中研究室

S113091 宮地 朱音

目次

1. 序論
 - 1.1 研究背景
 - 1.2 研究概要
 - 1.3 先行研究

2. 時刻表 CSV ファイル生成
 - 2.1 CSV ファイル生成の流れ
 - 2.2 扱うデータ
 - 2.3 プログラム作成のための開発環境
 - 2.4 時刻表 CSV ファイル生成プログラム

3. Google App Engine へのアップロード
 - 3.1 Google App Engine とは
 - 3.2 アップロードに際した開発環境
 - 3.3 検索用データのアップロード
 - 3.3.1 エンティティの定義
 - 3.3.2 データの追加
 - 3.3.3 アップロード作業

4. 結論
 - 4.1 実行結果
 - 4.2 今後の課題

謝辞

参考文献

1. 序論

1.1 研究背景

近年、Google Maps[1]等を用いた、視覚的にも便利な各種ルート案内サービスが普及してきている。

先行研究[2]による地図によるバス停時刻表検索システムもそのひとつである。このシステムは Google App Engine[3]を用いて高いスケールアウト性を確保した、島根県松江市のバス停・時刻表及び路線沿いの施設情報を検索し表示するシステムである。

このシステムは検索のために時刻表等のデータをあらかじめ独自の形式でまとめたものを使用しており、内容の更新は手作業で行う必要があるため、ダイヤの改正に伴う作業量は膨大なものとなる。

そこで、このシステムの検索用データをダイヤ改正時に更新する際、その手間を減らすための方法を検討することにした。

1.2 研究概要

先行研究の検索システムに対応した独自形式の時刻表データを、バス会社が公表している PDF ファイルをもとにして CSV ファイルにまとめ、生成するプログラムを作成した。

また、生成した CSV ファイルを Google App Engine[3]上にアップロードするためのプログラムも作成した。

これらのプログラムの作成においては、Eclipse 4.3(Kepler)[4]を使用した。データアップロードの際には Google Plugin for Eclipse[5]も利用し、ウェブアプリケーションのプロジェクト作成、ローカルサーバでの実行等を行った。

1.3 先行研究

先行研究[2]の松江市バス路線時刻表検索システムは Web ページ上で実行される。出発地と目的地をそれぞれ地図上でクリックし、日時を指定すると、クリックしたそれぞれの地点から半径 500m 以内のバス停を探し、それら 2 つのバス停を通るバス路線とその時刻表を検索し、バスの通るルートと共に表示するものとなっている。さらに、沿線情報も検索できるようになっている。

松江市バス停・バス路線時刻表検索システム[試作評価版]

①出発場所をクリックしてください。
変更したい場合は「出発変更」を押してください。

②目的地をクリックしてください。
変更したい場合は「目的変更」を押してください。

③出発時刻を選択し、検索ボタンをクリックしてください。
出発時刻 [平日] ● 平日 ● 休日 [検索]

④検索手順
指定された出発地から半径100m以内の出発バス停を検索します。見つからなければ検索範囲を100mずつ増やしていき、半径500m以内まで検索します。

出発場所選択

目的地選択

路線表示

バス停位置表示

バス路線位置表示

沿線検索範囲表示

島根大学前→松江駅

時間 **経路**

17:25 - 17:38 北備環線(内回り)松江駅→松江駅(北北台経由)

《所要時間》18分 [沿線検索範囲中](#) >>次へ

バス停位置表示

バス路線位置表示

沿線検索範囲表示

島根大学前→朝日町

時間 **経路**

17:28 - 17:44 県合同庁舎～作楽・大橋～川津(旧路)

《所要時間》16分 [沿線検索範囲中](#) >>次へ

バス停位置表示

バス路線位置表示

沿線検索範囲表示

島根大学前→朝日町

時間 **経路**

17:04 - 17:44 北備環線(内回り)松江駅→松江駅(県民会館経由)

《所要時間》40分 [沿線検索範囲中](#) >>次へ

バス停位置表示

バス路線位置表示

沿線検索範囲表示

島根大学前→松江駅

時間 **経路**

17:04 - 17:48 北備環線(内回り)松江駅→松江駅(県民会館経由)

《所要時間》44分 [沿線検索範囲中](#) >>次へ

バス停位置表示

バス路線位置表示

沿線検索範囲表示

島根大学前→松江駅

時間 **経路**

17:28 - 17:49 県合同庁舎～作楽・大橋～川津(旧路)

《所要時間》21分 [沿線検索範囲中](#) >>次へ

図 1.1 先行研究での実行例

2. 時刻表 CSV ファイル生成

2.1 CSV ファイル生成の流れ

まず対象とする PDF ファイルを開いて読み込み、全体をテキストとして抽出する。次に抽出したテキストから時刻表にあたる部分のみを切り抜き、その部分を一行ずつ整理して CSV に書き込んでいく。すべての行について出力できたらプログラムを終了する。

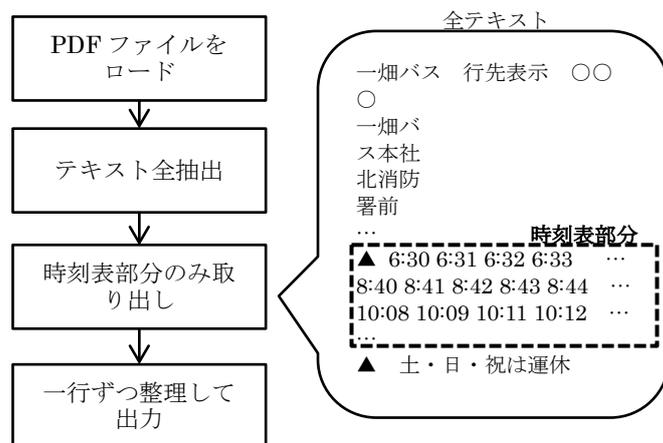


図 2.1 CSV ファイルを生成する流れ

2.2 扱うデータ

今回は、検索システムに必要なデータのひとつであるバス編成テーブルについて生成する。バス編成テーブルは各バス停の時刻表を格納する表である。出発バス停、目的バス停を通るバス路線を検索した後、このテーブルで時刻表を検索する。このテーブルは路線 ID、時刻、バス ID、備考の列で構成される。路線 ID は路線ごとにユニークに割り振った ID である。また、バス ID とは、同じ路線を通るバスを区別するために、その日のその路線の何本目のバスかというのを決定するためのものである。時刻については通過するバス停順に格納されている。このデータは、一畑バス[6]が提供する松江管内のバス時刻表 PDF ファイルを対象に取得した。

表 2.1 バス編成テーブル

路線 ID (STRING)	平日 (STRING)	休日 (STRING)	備考 (STRING)	バス ID (INT)	時刻 1 (INT)	時刻 2 (INT)	...	時刻 64 (INT)	時刻 (STRING)
10020101	1	0	休日運休	1	730	732	...	0	7:30/7:32/.../8:10
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	⋮
10460101	1	0	休日運休	1	1020	1021	...	0	10:20/10:21/.../11:01
10460101	1	0	休日運休	2	1120	1121	...	0	11:20/11:21/.../12:01
10460101	1	1		3	1220	1221	...	0	12:20/12:21/.../13:01
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	⋮

2.3 プログラム作成のための開発環境

プログラム作成にあたり開発環境を整えた。1.2 研究概要でも述べた通り Eclipse 4.3(Kepler)[4]を用い、開発言語は Java とした。PDF ファイルを扱うにあたり Java ライブラリである Apache PDFBox[7]を使用した。

2.4 時刻表 CSV ファイル生成プログラム

まず対象となる PDF ファイル名を指定して開く。続いて、開いたファイルの全文をテキスト抽出し、その中から時刻表部分のみを取り出す。以下にプログラムを示す。

```
35 > > > /* PDFドキュメントをロード */  
36 > > > PDDocument doc = PDDocument.load("time_" + rosen + ".pdf");  
37 > > > PrintWriter outFile = new PrintWriter(  
38 > > >     (new BufferedWriter(new OutputStreamWriter(new FileOutputStream("time_" + rosen + ".csv"), "MS932"))));  
39 > > > PDFTextStripper stripper = new PDFTextStripper();  
40 > > > String pdfData = stripper.getText(doc).trim();  
41  
42 > > > /* PDFデータから読み込んだテキストの時刻表テーブル部分のみを抜き出す */  
43 > > > int j = pdfData.indexOf(":"); //左側からの検索  
44 > > > String tmp = pdfData.substring(0, j); //全文の最初から":"出現直前まで  
45 > > > int t_start = tmp.lastIndexOf("#n"); //最後に出現する改行文字の位置  
46 > > > String pdfDataTmp = pdfData.substring(t_start);  
47  
48 > > > int k = pdfDataTmp.lastIndexOf(":"); //右側からの検索  
49 > > > tmp = pdfDataTmp.substring(k+1);  
50  
51 > > > int t_end = tmp.indexOf("#n");  
52 > > > String timeTable = pdfDataTmp.substring(0, k+t_end); //timeTable:時刻表部分だけ取り出した文字列  
53 > > > timeTable = timeTable.trim(); //最初と最後の改行文字を削除
```

図 2.2 PDF ファイルから時刻表部分をテキスト抽出するための記述

時刻表部分は、時刻を記述する際「6:30」のように”:(コロン)を用いているため、コロンが含まれる最初の行から最後の行までを取り出すように指定している。

次に取り出した時刻表部分を1行ずつ整理し、バス路線IDやバスID、平日運行・休日運行といった情報も付加してCSVファイルに書き込んでいく。このとき、「▲」（土日祝運休）の印が行頭にあるかないかで処理を分岐するようにしている。以下にプログラムを示す。

```

73 > > > int loop = countStringInString(timeTable, "\n") + 1; //時刻表部分の行数カウント↓
74 ↓
75 > > > for(int i = 1; i < loop; i++){↓
76 ↓
77 > > > > int tab = timeTable.indexOf("\n"); //時刻表で初めて改行が登場する位置(=一行目終端)↓
78 ↓
79 > > > > if((timeTable.substring(0, tab)).indexOf("▲") == -1){ //一行目に▲(土日祝運休)がなければ↓
80 > > > > > ttline = timeTable.substring(0, tab+1); //一行目↓
81 > > > > > ttlineStr = (ttline.replaceAll(" ", "/"), trim());↓
82 > > > > > ttline = (ttline.replaceAll(":", ";"), trim());↓
83 > > > > > outFile.write(bus_id + ",1,1," + "一畑バス," + i + ",");↓
84 ↓
85 > > > > > ttSplit = ttline.split("[\s]+"); //半角スペースごとに区切る↓
86 ↓
87 > > > > > int a;↓
88 > > > > > for (a = 0; a < ttSplit.length; a++){↓
89 > > > > > > outFile.write(ttSplit[a] + ",");↓
90 > > > > > }↓
91 ↓
92 > > > > > for(int b = 63-a; b>0; b--){↓
93 > > > > > > outFile.write(",");↓
94 > > > > > }↓
95 > > > > > outFile.write(ttlineStr + "\n");↓
96 ↓
97 > > > > } else{ //一行目に▲があれば↓
98 > > > > > ttline = timeTable.substring(2, tab+1); //一行目から▲を抜いたもの↓
99 > > > > > ttlineStr = (ttline.replaceAll(" ", "/"), trim());↓
100 > > > > > ttline = (ttline.replaceAll(":", ";"), trim());↓
101 > > > > > outFile.write(bus_id + ",1,0,休日運休," + "一畑バス," + i + ",");↓
102 > > > > > ttSplit = ttline.split("[\s]+");↓
103 ↓
104 > > > > > int a;↓
105 > > > > > for (a = 0; a < ttSplit.length; a++){↓
106 > > > > > > outFile.write(ttSplit[a] + ",");↓
107 > > > > > }↓
108 ↓
109 > > > > > for(int b = 63-a; b>0; b--){↓
110 > > > > > > outFile.write(",");↓
111 > > > > > }↓
112 > > > > > outFile.write(ttlineStr + "\n");↓
113 > > > > > }↓
114 > > > > > timeTable = timeTable.substring(tab + 1); //CSVに書き込み済みの一行を削除する↓
115 > > > }↓

```

図 2.3 時刻表データを CSV ファイルにまとめ出力するための記述

3. Google App Engine へのアップロード

3.1 Google App Engine とは

Google App Engine[3]とは、Google のインフラ上で、開発した Web アプリケーションを実行できる Web アプリホスティングサービスである。データストアは Key-Value Store である Bigtable で、負荷分散を自動で行うため、アクセスが集中してもパフォーマンスが悪化しないという特長を持つ。

また、ある程度までの使用は無料でできる点や、サーバ構築の必要がない点から、アプリケーションの公開が容易に行えるという利便性もある。

3.2 アップロードに際した開発環境

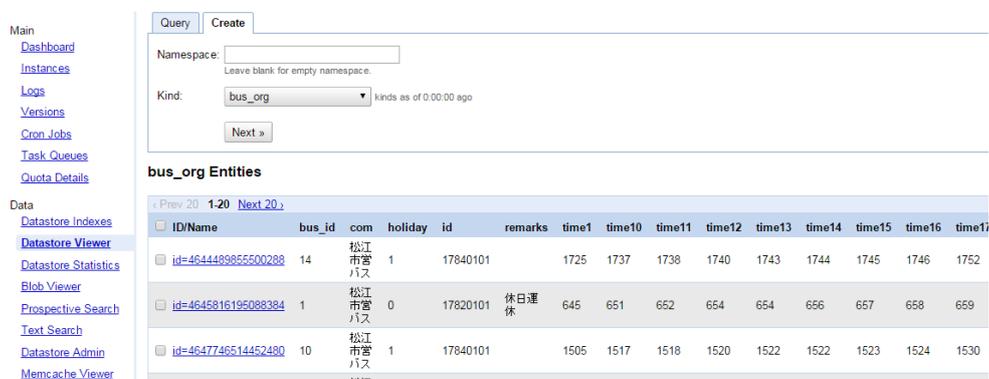
Google App Engine 上へのデータアップロードに際し、Google App Engine と Eclipse の連携を取る必要があったため、Google App Engine SDK for Java[8] のパッケージの「appengine-java-sdk-1.9.5.zip」をダウンロードし、インストールした。そして Eclipse の機能を用いて Google Plugin for Eclipse[5]をインストールした。さらに、CSV ファイルアップロードのために

「commons-fileupload-1.3.1.jar」を追加した。

また、データの格納には Google App Engine でデータストアの API 標準としてサポートされている Java Data Objects を利用した。

3.3 検索用データのアップロード

Google App Engine の管理画面には Datastore Viewer があり、データストアにあるデータの確認やアップロード等はそこから行えるが、用意したデータをひとつひとつアップロードするには膨大な手間がかかる。そこで、CSV ファイルにまとめたデータをアップロードするプログラムを、先行研究に従って作成した。



The screenshot shows the Google App Engine Datastore Viewer interface. On the left is a navigation menu with links for Main (Dashboard, Instances, Logs, Versions, Cron Jobs, Task Queues, Quota Details) and Data (Datastore Indexes, Datastore Viewer, Datastore Statistics, Blob Viewer, Prospective Search, Text Search, Datastore Admin, Memcache Viewer). The main area has tabs for 'Query' and 'Create'. Below the 'Create' tab, there is a 'Namespace' field (empty) and a 'Kind' dropdown menu set to 'bus_org'. A 'Next >' button is visible. Below this is a table titled 'bus_org Entities' with columns: ID/Name, bus_id, com, holiday, id, remarks, time1, time10, time11, time12, time13, time14, time15, time16, time17. The table contains three rows of data for bus entities.

ID/Name	bus_id	com	holiday	id	remarks	time1	time10	time11	time12	time13	time14	time15	time16	time17
<input type="checkbox"/> id=4644489855500288	14	松江市営バス	1	17840101		1725	1737	1738	1740	1743	1744	1745	1746	1752
<input type="checkbox"/> id=4645816195088384	1	松江市営バス	0	17820101	休日運 休	645	651	652	654	654	656	657	658	659
<input type="checkbox"/> id=4647746514452480	10	松江市営バス	1	17840101		1505	1517	1518	1520	1522	1522	1523	1524	1530

図 3.1 Google App Engine の管理画面の一部

3.3.1 エンティティの定義

Google App Engine では Java Data Objects をサポートしているため、これを用いてオブジェクトの永続化を行う。

```
11 @PersistenceCapable(identityType = IdentityType.APPLICATION) ↓
12 public class bus_org { ↓
13     ↓
14     > @PrimaryKey ↓
15     > @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY) ↓
16     > private Key key; ↓
17     ↓
18     > @Persistent ↓
19     > private String id; //バス路線ID ↓
20     ↓
21     > @Persistent ↓
22     > private String weekday; //平日運行かどうか(0/1) ↓
23     ↓
24     > @Persistent ↓
25     > private String holiday; //休日運行かどうか(0/1) ↓
26     ↓
27     > @Persistent ↓
28     > private String remarks; //備考 ↓
29     ↓
30     > @Persistent ↓
31     > private String com; //バス会社名 ↓
32     ↓
33     > @Persistent ↓
34     > private int bus_id; //バスID ↓
35     ↓
36     > @Persistent ↓
37     > private int time1; //時刻1 ↓
38     ↓
39     > @Persistent ↓
40     > private int time2; //時刻2 ↓
41     ↓
42     > @Persistent ↓
43     > private int time3; ↓
44     ↓
45     > @Persistent ↓
46     > private int time4; ↓
```

図 3.2 エンティティの定義

@PersistenceCapable で永続化対象のクラスであることを宣言し、@Persistent で永続化対象のフィールドであることを宣言、@PrimaryKey でエンティティの主キーを定義している。

3.3.2 データの追加

データの追加を行うプログラムを以下に示す。

```
36     FileItemIterator itemIterator = fileUpload.getItemIterator(req);  
37     while (itemIterator.hasNext()) {  
38         FileItemStream itemStream = itemIterator.next();  
39         InputStream inputStream = itemStream.openStream();  
40         String contentType = itemStream.getContentType();  
41         if (contentType == null) {  
42             contentType = "";  
43         }  
44         if (contentType.contains("text") || itemStream.getName().endsWith(".csv")) {  
45             resp.setContentType("text/html; charset=UTF-8");  
46             BufferedReader buffer = new BufferedReader(new InputStreamReader(inputStream, "UTF-8"));  
47             String line = null;  
48             int i = 0;  
49             while ((line = buffer.readLine()) != null) {  
50                 String[] split = line.split(",",-1);  
51                   
52                 // csvからデータを取り出す  
53                 String id = split[0].trim();  
54                 String weekday = split[1].trim();  
55                 String holiday = split[2].trim();  
56                 String remarks = split[3].trim();  
57                 String com = split[4].trim();  
58                 int bus_id = Integer.parseInt(split[5].trim());  
59                 int time1 = Integer.parseInt(split[6].trim());  
60                 int time2 = Integer.parseInt(split[7].trim());  
61                 int time3 = Integer.parseInt(split[8].trim());  
62                 int time4 = Integer.parseInt(split[9].trim());  
63                   
64                 :  
65                   
66                 int time63 = Integer.parseInt(split[68].trim());  
67                 int time64 = Integer.parseInt(split[69].trim());  
68                 String time_table = split[70].trim();  
69                   
70                 bus_org per = new bus_org(id, weekday, holiday, remarks, com, bus_id,  
71                 > > time1, time2, time3, time4, time5, time6, time7, time8, time9, time10,  
72                 > > time11, time12, time13, time14, time15, time16, time17, time18, time19, time20,  
73                 > > time21, time22, time23, time24, time25, time26, time27, time28, time29, time30,  
74                 > > time31, time32, time33, time34, time35, time36, time37, time38, time39, time40,  
75                 > > time41, time42, time43, time44, time45, time46, time47, time48, time49, time50,  
76                 > > time51, time52, time53, time54, time55, time56, time57, time58, time59, time60,  
77                 > > time61, time62, time63, time64, time_table);  
78                   
79                 PersistenceManager pm=pmf.getPersistenceManager();  
80                   
81                 if (pm != null) {  
82                     try{  
83                         > pm.makePersistent(per);  
84                     }finally{  
85                         > pm.close();  
86                     }  
87                 }  
88             }  
89         }  
90     }  
91 }
```

図 3.3 データの追加を行うプログラム

String 型配列「split」に CSV ファイルのデータを一行ずつ取得し、trim メソッドでその各パラメータを取り出し、インスタンス化してデータストアに格納している。

3.3.3 アップロード作業

3.3.2 節のプログラムをサーブレットとし、簡単な HTML ファイルを作成した。アップロードはこの HTML のページ上で実行する。

```
1 <!DOCTYPE html>↵
2 ↵
3 <html>↵
4 <head>↵
5 <meta http-equiv="content-type" content="text/html; charset=UTF-8">↵
6 <title>File Upload</title>↵
7 </head>↵
8 <body>↵
9 <h1>File Upload(バス編成)</h1>↵
10 <form action="/fileuploadbusorg" method="post" enctype="multipart/form-data">↵
11 <input type="file" name="file" /><br />↵
12 <input type="submit" value="アップロード" />↵
13 </form>↵
14 </body>↵
15 </html>↵
```

図 3.4 アップロード作業を行うための HTML ページ

File Upload(バス編成)



ファイルを選択 time_daitou_0.csv
アップロード

図 3.5 アップロード作業画面

フォームで CSV ファイル名を指定し、アップロードボタンを押すとアップロードを開始する。

4.結論

4.1 実行結果

一畑バスの各時刻表 PDF ファイルについて、それぞれ CSV ファイル生成が出力できた。

20000000	1	0	休日運休	一畑バス	1	630	631	632
20000000	1	0	休日運休	一畑バス	2	640	641	642
20000000	1	0	休日運休	一畑バス	3	650	651	652
20000000	1	0	休日運休	一畑バス	4	710	711	712
20000000	1	1		一畑バス	5	740	741	742
20000000	1	0	休日運休	一畑バス	6	800	801	802
20000000	1	1		一畑バス	7	840	841	842
20000000	1	1		一畑バス	8	940	941	942
20000000	1	1		一畑バス	9	1040	1041	1042
20000000	1	1		一畑バス	10	1140	1141	1142
20000000	1	1		一畑バス	11	1240	1241	1242
20000000	1	1		一畑バス	12	1340	1341	1342
20000000	1	1		一畑バス	13	1440	1441	1442
20000000	1	1		一畑バス	14	1540	1541	1542
20000000	1	0	休日運休	一畑バス	15	1605	1606	1607

図 4.1 出力された CSV データの一部

また、Google App Engine へのアップロードが成功した。

実行に用いた HTML ページのアドレスは

<http://1-dot-s113091-su.appspot.com/fileuploadbusorg.html>

である。

4.2 今後の課題

PDF ファイルの情報をテキストとして抜き出す際、下図のように空白テーブルがあるとその部分が抜け落ちるようになってしまう。このような抜け落ちのないプログラムに改良するのが今後の課題である。

また、今回は一畑バスのデータを対象に時刻表データ生成プログラムを作成したため、他形式の PDF ファイルには対応していない。よって、他社の時刻表 PDF ファイルの形式にも対応するようなシステムの検討も課題として挙げられる。

↑時刻表 PDF ファイル
出力された CSV ファイル↓

▲	15:40	15:41	15:42	15:43	15:44	15:50	15:52	15:53	15:54	15:55	15:56
▲	16:05	16:06	16:07	16:08	16:09	16:15	16:17	16:18	16:19	16:20	16:21
	16:40	16:41	16:42	16:43	16:44	16:50	16:52	16:53	16:54	16:55	16:56
					▲	16:55	—	—	—	—	—
▲	17:05	17:06	17:07	17:08	17:09	17:15	17:18	17:19	17:20	17:21	17:23
	17:20	17:21	17:22	17:23	17:24	17:30	17:33	17:34	17:35	17:36	17:38
	17:40	17:41	17:42	17:43	17:44	17:50	17:53	17:54	17:55	17:56	17:58

1540	1541	1542	1543	1544	1550	1552	1553	1554	1556
1605	1606	1607	1608	1609	1615	1617	1618	1619	1621
1640	1641	1642	1643	1644	1650	1652	1653	1654	1656
1655	—	—	—	—	—	—	—	—	—
1705	1706	1707	1708	1709	1715	1718	1719	1720	1723
1720	1721	1722	1723	1724	1730	1733	1734	1735	1738
1740	1741	1742	1743	1744	1750	1753	1754	1755	1758

図 4.2 テーブルの空白部分の欠落

謝辞

本研究を進めるにあたり、最後まで熱心なご指導をして頂きました田中章司郎教授には心より御礼申し上げます。

また、数々のご協力とご助言をして頂きました同研究室の皆様にも、厚く御礼申し上げます。

なお、本論文、本研究で作成したプログラム及びデータ、並びに関連する発表資料等の全ての知的財産権を本研究の指導教官である田中章司郎教授に譲渡致します。

参考文献

- [1] <http://maps.google.co.jp/>
- [2] 三島健太, 「JTS を用いた沿線検索システムの Google App Engine 上での実装」, 島根大学 総合理工学研究科 数理・情報システム学専攻 修士論文, 2012
- [3] <https://cloud.google.com/appengine/docs>
- [4] <http://www.eclipse.org/>
- [5] <https://cloud.google.com/appengine/docs/java/tools/eclipse>
- [6] <http://www.ichibata.co.jp/bus/>
- [7] <http://pdfbox.apache.org/>
- [8] <https://cloud.google.com/appengine/downloads>