Contextual Suggestion

Abhishek Mallik
Department of Computer Science and Enginerring
IIT Bombay
Powai, Mumbai 400076, India

Mandar Mitra
Computer Vision and Pattern Recognition Unit
ISI Kolkata
Kolkata 700 108, India

Kripabandhu Ghosh
Computer Vision and Pattern Recognition Unit
ISI Kolkata
Kolkata 700 108, India

October 16, 2012

Abstract

The goal of this project is to build a Contextual Suggestion system that will recommend the user a ranked list of suggestions depending on user's context as well as her preferences.

In this context we present an algorithm based on the regular expression for extracting time and address information from different websites for the places. Then we have suggested a ranking function that can utilize these timing and address information of those places as well as the users' context and preferences to rank the places. Here the pages are unstructured but the address is structured i.e. template based as we have worked with only the places located at Toronto, Canada.

1 Introduction

Main motivation behind this task is to build future information retrieval systems that must anticipate user needs and respond with information appropriate to the current context without the user having to enter an explicit query. In a mobile context such a system might take the form of an application that recommends interesting places and activities based on the user's location and time and other factor such as her preference etc.

- **Step 3** Google Map also gives some status messages along with the result regarding whether the place is closed or not. So we check if there is an associated status message saying that "The place is permanently closed". If not, we add the address to the address list [2] of that place. So by utlizing this feature of Google Map we have made our algorithm more reliable.

- **Step 4** Repeat step 2 and 3 until end of the last result of the page is encountered (when there are multiple such results on the page).

- **Step 5** If any address is found in the above steps, then stop, else goto next step.

- **Step 6** Repeat Step 1 with Google search[3] to find the address and download first result page.

- **Step 7** Result page basically consists of a collection of URLs with associated titles and snippets. We apply address extraction step to search for potential address patterns in those snippets of the results, starting from the top of the result page.

- **Step 8** If the best matched address pattern returned by the address extraction step contains full adress then we stop. Otherwise if partial address is found in the snipper, then that URL associated with the snippet is crawled with the hope of finding the full address.

- **Step 9** So if full address found then stop, if again a partial address found after crawling the URL, go to next step, else go to step 10.

- **Step 10** Query *www.yelp.ca*[4] with the title name and partial address, return the full address if found, otherwise the partial address, that has been found earlier, is returned.

- **Step 11** Here we crawl the URL of the example itself (that is also given with each exmaple in the file `examples.txt`) upto depth 2[5] and then applied the following address extraction step on the crawled pages.

- **Address Extraction Step** For that step, we have surveyed quite a large number of addresses in Canada (as all the examples are taken from Toronto, Onatrio). We have found that most of them follow a particular template as follows: They start with street number (in digit) followed by street Name followed by one of these termns **Street, St., Road, Rd., Boulevard, Blvd., Avenue, Ave., Parkway, Pky, Highway, Hwy, Broadway, Way, Wy, Drive, Dr, Turnpike** followed by one of these

[2]There may be more than one outlets of a shop or restaurant in a city

[3]http://www.google.com

[4]`yelp.ca` is a well established aggregator site in Canda that contains information about restaurants in Canada

[5]Usually for all of such websites addresses are found at depth ≤ 2

terms **North, South, East, West** followed by term **Toronto** then **Ontario/ON** and then postal code which is in alpha-numeric format followed by Country Name.

But sometimes they deviate from this by certain amount, for example `Country Name`, the direction `North`, `South`, `East` and `West` are not present in all of them. So we have made a regular expression using the above heuristic but not made the above requirements stringent.

As a result with the valid address expressions a few meaningless expressions also have got returned which have nothing to do with an address. So those matched expressions was again scored, based on how much they are close to the above template. Expression having maximum score is returned, which is nothing but the valid postal address we are looking for.

2.2 Temporal Interval Extraction Algorithm

Temporal Intervals of geographic entities are expressed in various forms. Some of the motivating examples are as follows: To extract timing, we have used several heuristics in our algorithm as follows:

- **Step 1** Query Google with the title of the example appended with "Toronto[6] hours of operation", for example if the title is "CN Tower", we have formulated the query as "CN Tower Toronto hours of operation" and download the first result page. We have chosen the phrase "hours of operation" because mostly in the websites of places like restaurant, hotel, cafe, art gallery etc. timing information is specified as "hours of operation".

- **Step 2** Parse the downloaded result page to collect URLs, titles and snippets of all results, store them in separate files.

- **Step 3** Examine the URLs collected in the above step one by one from top to check if any of them conains the url of the example itself or contains the term **"faq"**, as index page of a site and an url containing the term **"faq"** are more likely to contain the timing information. If any such URL is found, download the page at this URL, else, goto step 5.

- **Step 4** Apply timing information extraction step (this will be described later in this report) on the downloaded page.

- **Step 5** If any timing expression is found in the above step then stop, else goto next step.

- **Step 6** Check rest of the results one by one from top whether its title contains title of concerned example or its associated URL contains the URL of the concerned example. If any of these two satisfies, goto next step else goto step 10.

[6]As here we are concerned with city of Toronto and there may be outlets of a brand (for example, Pizza Hut) all over the world, otherwise we may get erroneous result

- **Step 7** Associated snippet checked to see if it contains certain phrases like **hours of operation/operating hours/timing/hour**, **am/pm** and **mon/tue/wed/thu/ .../daily** etc. If yes, then the associated page from which this snippet has been generated is more likely to contain the timing information. So page at the associated URL is downloaded and then we goto next step else, goto step 10.
- **Step 8** Apply timing information extraction step on the page which we have just downloaded in the above step.
- **Step 9** If any timing expression is found in step 8 then stop, else goto next step.
- **Step 10** This is again a fallback step. Here we crawl the URL of the concerned example itself upto depth 2[7].
- **Step 11** Parse all these web pages that we have crawled at step 10 and applied timing information extraction step on the crawled pages.
- **Timing Extraction Step** Similar to address extraction step, we have tried to come up with a regular expression for extracting timings also. As we have seen earlier timing information i.e. temporal interval may appear in several forms, that is why we have to turn and twist the regular expression several times in order to capture the interval accurately. So it is clear from this discussion so far that this temporal interval extraction requires a considerable amount of manual intervention.

  Then we have scored the expressions that have matched with our final regular expression to determine which expression is most fit to appear as temporal interval of a geographic entity.

2.3 Computation of Geo-Temporal Match

To compute the geo-temporal match between the example place and user's context we shall first discuss about how we have utlized the addresses of the example places to determine their proximity with respect to the geographic contexts of the users, then we shall discuss about matching between the temporal intervals of the geographic entities and the temporal contexts of the users. So to utlize the users' geographic context, we have followed the approach as follows:

2.3.1 Finding geographical match

We cannot directly process the address directly, so to utlize them in ranking we have to map them to the geographic distance of that place. In order to do this, each of the postal address found in the above steps is passed as parameter to Yahoo! geocoding API[8] to find corresponding latitude and longitude and

[7]The same logic in case of address extraction is applicable here i.e. usually all such websites contain timing information at depth ≤ 2

[8]http://api.maps.yahoo.com/ajax/geocode?appid=onestep&qt=1 &id=m&qs="address"

then the geographic distance is calculated from user's location to that place by applying Vincent's formula, that takes latitudes and longitudes of two places and computes the geographic distance.

There are places having more than one outlets in a city and each of the outlet has a postal address. So for each of them we calculate the distance to the user's current geographic context and select the outlet with the minimum distance, which is only taken as the distance of the entity in the user's current context.

Further we normalize the distances as follows: for each context of the user, after getting its distance to each of the example place in the above step, we find the maximum distance `max` among them and divide each distance that I have computed earlier by `max`. As a result the resulting distances fall in the range between 0 and 1. So from the discussion so far, it is clear that we are successfully able to utilize user's geographic contxet.

2.3.2 Finding temporal match

Here like address we also postprocess the temporal interval that we have extracted in the earlier steps, as follows: Suppose for one place hours of operation is as follows: "$Mon - Fri$: $10am$ – 10pm Sat and Sun : $11am - 8pm$". As we cannot incorporate timing information in this format in our ranking function, so it is converted to the following expression:

weekday, 1, morning, 1, 4, afternoon, 0, 5, evening, 0, 2, weekend, 1, morning, 2, 3, afternoon, 0, 5, evening, 0, 2

Here we have followed the format as follows: weekday, term1, morning, term2, term3, afternoon, term4, term3, evening, term5, term3, weekend, term6, morning, term2, term3, afternoon, term4, term3, evening, term5, term3

The meaning of each of the terms are given as follows:

$$\begin{aligned}
term1 &= \frac{\text{number of weekdays the location is opened}}{5} \\
term2 &= \text{(opening time of the location - 9 am [9])} \\
term3 &= \text{(duration the location is opened during this 5 hours)} \\
term4 &= \text{(opening time of the location - 1 pm)} \\
term5 &= \text{(opening time of the location - 6 pm)} \\
term6 &= \frac{\text{number of weekends the location is opened}}{2}
\end{aligned}$$

While converting the timing information that we have extracted from the pages to the format (that we have shown above) we have faced a challenge. The timing information we have extracted for places are basically daywise of a week. But here in this task we are concerned with only "weekday" and "weekend".

[9] According to assumption in this track, morning session starts at 9 am, afternoon session starts at 1 pm, evening session starts at 6 pm and duration of each session is 5 hours

There are places where the business hours vary daywise in each week. So for handling that, we have simply taken the maximum of the opening times and minimum of the closing times for each place for weekdays and weekends separately and consider them as starting times and closing times of those places. We illustrate this with a simple example as follows:

Suppose there is a place A for which business hour is as follows: **Mon-Wed: 10 AM-11PM, Thu and Fri: 11AM-9PM, Sat: 10AM-9PM, Sun: 10AM-5PM**. In that case for "weekday", according to our logic we take opening time of A as 11 AM and closing time as 9 PM and for "weekend" we take opening time as 10AM and closing time as 5PM.

3 Ranking Function

To rank the example places first we have to determine how the spatial information and temporal information, that we have gathered already, can influence ranking function individually and then finally we shall combine their influences. So, we shall present first the ranking function itself and then explain influence of each of them separately.

$$rf() = \begin{Bmatrix} t_1 \times (-t_2 + t_3 + (1-d)), & t_1 \geq 0 \\ (1-d), & \text{Timing information is not found} \end{Bmatrix} \quad (1)$$

3.1 Incorporating Geographic Information In Ranking Function

While considering the spatial information only, we can easily conclude that farther a place is from an user's geographiic context, lower is its rank and vice versa. We have incorporated the contribution of the distance in ranking function $rf()$ as follows:

If the normalized distance (that we have talked about earlier) from a place to an user's geographic context is computed as d, then the contribution of the distance to ranking function $rf()$ is $(1-d)$ and that appears in the ranking function $rf()$ as an $+$ term in the equation 1.

3.2 Incorporating Temporal Information In Ranking Function

Now we shall discuss the influence of timing information on the ranking function $rf()$. Like spatial information, timing information also plays a crucial role in the ranking function $rf()$. Timing information will come in three different forms in the ranking function:

First and most important thing is the probability that a place is opened on a "weekday" or a "weekend" (as these are the only two days of a week we are interested in the current task). We have calculated this probability t_1 as follows: For weekdays t_1 is the number of days between Monday to Friday, the place is opened divided by 5 (which is the number of weekdays) and for weekend t_1 is number of days between Saturday and Sunday, it is opened divided by 2 (which is the number of weekends in a week). This term is the most important in the ranking function for the following simple reason i.e. if the place is not available in that day i.e. weekday or weekend, then whatever the distance between this place and current context is, it should get a score of 0. So we can incorporate the contribution of timing information in this probability form in ranking function $rf()$ as follows: If the probability is t_1, then we can write its contribution to the ranking function $rf()$ as t_1 itself and it appears as an AND i.e a product term as shown in the above equation 1.

Second important thing is the time gap between the time a session (i.e. `morning`, `afternoon` or `evening` whichever is applicable with respect to user's context) starts and time the place is available in that session, which is basically captured by one among *term*2, *term*3 and *term*4 depending upon the user's temporal context. Larger this time gap is, less is the availability of the associated place in the current user's context. So later the place opens in a session, which is an user's current context, lower its rank will be in the ranked list. So it is quite clear that, ranking function $rf()$ should be decreasing function of this time gap. So we have incorporated the contribution of this this time gap in ranking function $rf()$ in equation 1 as follows: If the time gap is t, which is nothing but, one among *term*2, *term*4 or *term*5, depending on the user's current context, then the contribution of timing information in normalized form to the ranking function $rf()$ is $-t_1$ (where $t_1 = t/5$) and it appears as a + term in equation 1.

The last form, by which the timing information can influence the ranking function $rf()$, can be explained as follows: in a session that is appropriate to user's current context, more duration a place is opened, more will it gets score, so higher will be its rank in the ranked list of places. It is quite clear that ranking function $rf()$ should be increasing function of this time duration. So we have incorporated the contribution of temporal interval in the form of time duration in our ranking function $rf()$ in equation 1 as follows:

Here we have taken the normalized time duration, which is nothing but *term*3/5 (*term*3 is actual time duration with which the place is opened in that session and according to ssumption of this task duration of each session is 5 hours). So if it is t_3, then the contribution of timing information in this form to the ranking function $rf()$ is t_3 i.e. the time duration in the normalized form itself and that also appears as a + term as shown in equation 1.

But there are few places, where we could not extract the timing information due to the reason of unavailability of this information or it was embedded in such a format we could not extract it.

4 Results

In this section we describe the results. There are 3 **evaluation measures**. The measures are $P@5$ for each of the following combinations of results:

WGT, GT and W. Due to space constraint, we are reporting here the result for only the dimension P@5_W (As for other two dimensions we have got worst result). Here following table shows per-context statistics based on mean scores over profiles.

Table 1: Table for the metric P@5_W

P(5)@W				
Context	Best	Median	Worst	Our value
3	0.8000	0.4000	0.0000	0.8000
4	0.7000	0.3000	0.0000	0.7000
9	0.7000	0.3000	0.0000	0.3000
11	0.7000	0.3000	0.0000	0.4000
12	0.6000	0.0000	0.0000	0.0000
13	1.0000	0.6000	0.0000	0.6000
14	1.0000	0.4000	0.0000	1.0000
15	0.8000	0.2000	0.0000	0.8000
16	0.8000	0.3000	0.0000	0.8000
18	1.0000	0.4000	0.0000	1.0000
19	0.4000	0.0000	0.0000	0.2000
20	0.4000	0.0000	0.0000	0.0000
22	0.8000	0.4000	0.0000	0.2000
23	0.7000	0.4000	0.0000	0.6000
28	0.8000	0.4000	0.0000	0.6000
29	0.6000	0.2000	0.0000	0.0000
30	0.8000	0.0000	0.0000	0.6000
31	0.8000	0.2000	0.0000	0.2000
32	0.8000	0.4000	0.1000	0.7000
33	0.8000	0.4000	0.0000	0.6000
34	1.0000	0.4000	0.0000	0.8000
35	0.8000	0.4000	0.2000	0.7000
36	1.0000	0.6000	0.0000	1.0000
37	0.8000	0.4000	0.0000	0.6000
38	0.8000	0.4000	0.0000	0.8000
39	1.0000	0.4000	0.0000	0.8000
40	1.0000	0.6000	0.2000	1.0000
41	0.8000	0.4000	0.0000	0.4000
42	0.9000	0.5000	0.1000	0.8000
43	0.8000	0.2000	0.0000	0.4000
44	1.0000	0.4000	0.0000	1.0000
45	0.8000	0.4000	0.0000	0.6000
46	0.0000	0.0000	0.0000	0.0000
47	1.0000	0.6000	0.0000	1.0000
50	0.8000	0.5000	0.1000	0.5000

Above table shows a best, median and worst values over all runs submitted to this track, as well as the same value for our run for the measure P@5_W per context. It is clear from the table that, almost in all the cases P@5_W for our run is close to the best result.

Now in the following tables we have reported per-profile statistics based on mean scores over contexts.

Table 2: Table for the metric P@5_W

P(5)@W				
Context	Best	Median	Worst	Our value
2	0.6667	0.4000	0.0000	0.6667
3	0.7333	0.2667	0.0000	0.7333
6	0.8000	0.4000	0.0667	0.8000
8	0.8667	0.4667	0.2000	0.7333
11	0.7333	0.2667	0.0000	0.6667
14	0.6667	0.4000	0.0000	0.6000
16	1.0000	0.4000	0.0000	1.0000
17	0.6000	0.4000	0.0000	0.6000
18	0.8000	0.4000	0.0000	0.8000
19	0.3000	0.1000	0.0000	0.1000
21	0.0000	0.0000	0.0000	0.0000
22	0.7333	0.2000	0.0000	0.7333
23	0.4000	0.0000	0.0000	0.2000
25	0.8667	0.6000	0.2667	0.8667
26	0.8000	0.3000	0.0000	0.7000
27	0.7333	0.5333	0.0667	0.4000
28	0.6000	0.2000	0.0000	0.0000
31	1.0000	0.4000	0.0000	1.0000
33	0.6000	0.3333	0.0000	0.2667

Above table shows a best, median and worst values over all runs submitted to this track, as well as the same value for our run for the measure P@5_W per context. It is clear from the table that, almost in all the cases P@5_W for our run is close to the best result.

5 Conclusions

In this report we have seen how to extract temporal and address information from web pages having heterogeneous strcuture, that is having no template common to each other and utlizing these extracted information suggest a ranked list of places to user depending on her context.