TripAdvisor Crawler

Development of a focused crawler in Java
for reviews-extraction from TripAdvisor

final project

Web Information Management

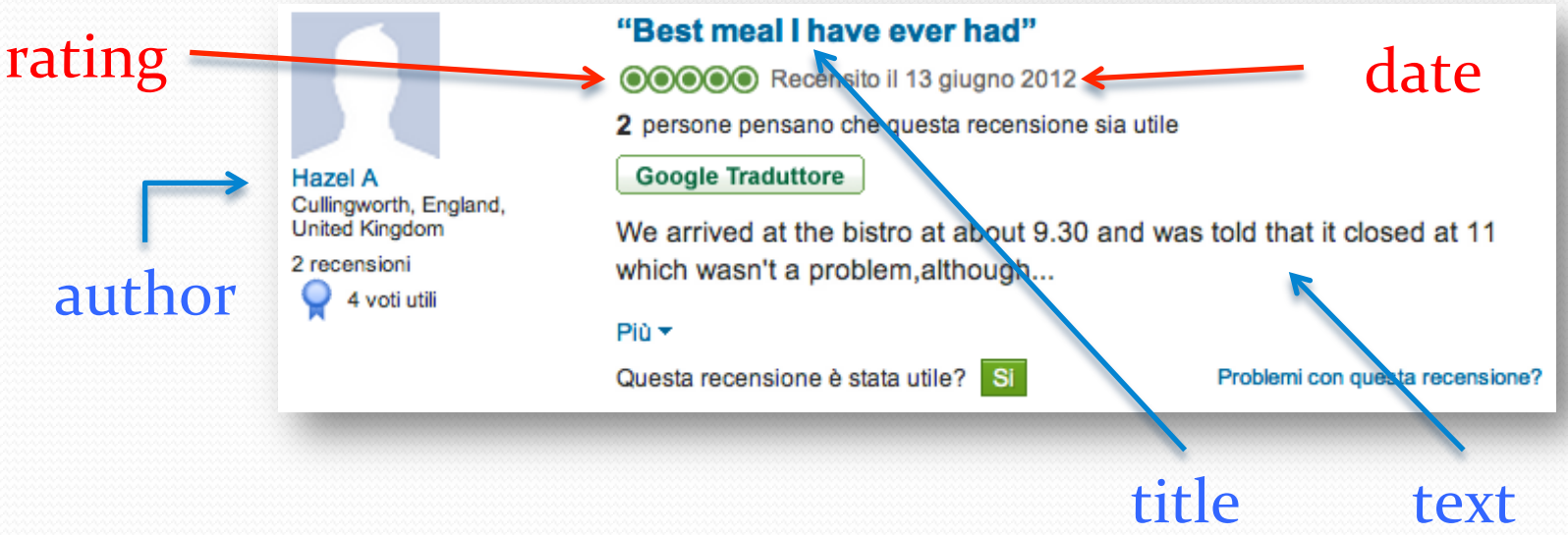
D. Grasso, M. Lombardi, A. Marani, P. Paris

Analysis of TripAdvisor

- To adapt the crawler to the website:
  1. Which ones are the data of interest?
  2. On which pages are these data displayed?
  3. How do we reach all these pages?

- Data of interest are opinions, expressed in the reviews of the users.
- Each *review* can belong to a hotel, a restaurant or attractions.
- In general terms we can define them as *items*, each one having many *review*.

Review



- We choose to focus only on rating and date
- Every review has also a unique identifier (reviewID) – like: "r133345545"

tripadvisor.it
Il portale di viaggi più grande del mondo

Item

- Is an abstraction representing either:
  - Hotel
  - Restaurant
  - Attraction
- Also for items we focus only on essential data:
  - Name
  - Number of reviews (useful for crawling…)
  - Identifier (itemID) - like: “d1132860”

tripadvisor.it
Il portale di viaggi più grande del mondo

Location

- Represents a geographic place f interest, where it is possible to find many items.
- The entities of type *location* are structured in a hierarchy, with a variable grain.

- Each Location has a name and an ID – examples:
  - Europe - "g4", France- "g187070", Paris- "g187147"
- Even if very useful to understand the structure of the website, these data are not going to be extracted.

tripadvisor.it
Il portale di viaggi più grande del mondo

TripAdvisor - Fact sheet

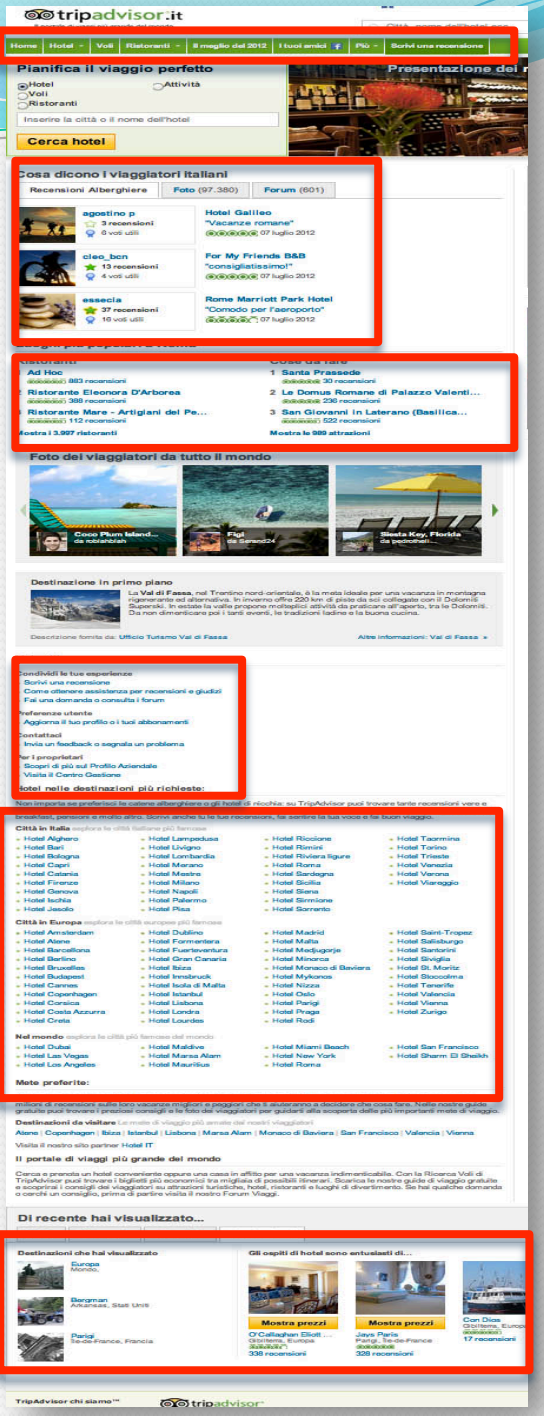
- How many are the items we need to extract?
  - More than 60 million travel reviews
  - 108,000+ destinations
  - 1,600,000+ businesses
    - 600,000+ hotels ,
    - 198,000+ attractions,
    - 858,000+ restaurants

from: *http://www.tripadvisor.com/PressCenter-c4-Fact_Sheet.html*

- So, for a complete crawling, we'll have:
  - 60,000,000+ reviews
  - 1,600,000+ items

Page Classes (1)

- Class: Home Page
  - A good starting point, containing many many links to pages of every kind, and specifically:
  - Most popular and newest Items
  - Main Locations
  - Other types of pages to ignore (good for testing)
    - like Flights, Forum, …



tripadvisor.it
Il portale di viaggi più grande del mondo

tripadvisor.it
Il portale di viaggi più grande del mondo

Page Classes (2)

- Class: List-of-locations

  Allows navigation in the locations tree, in which:

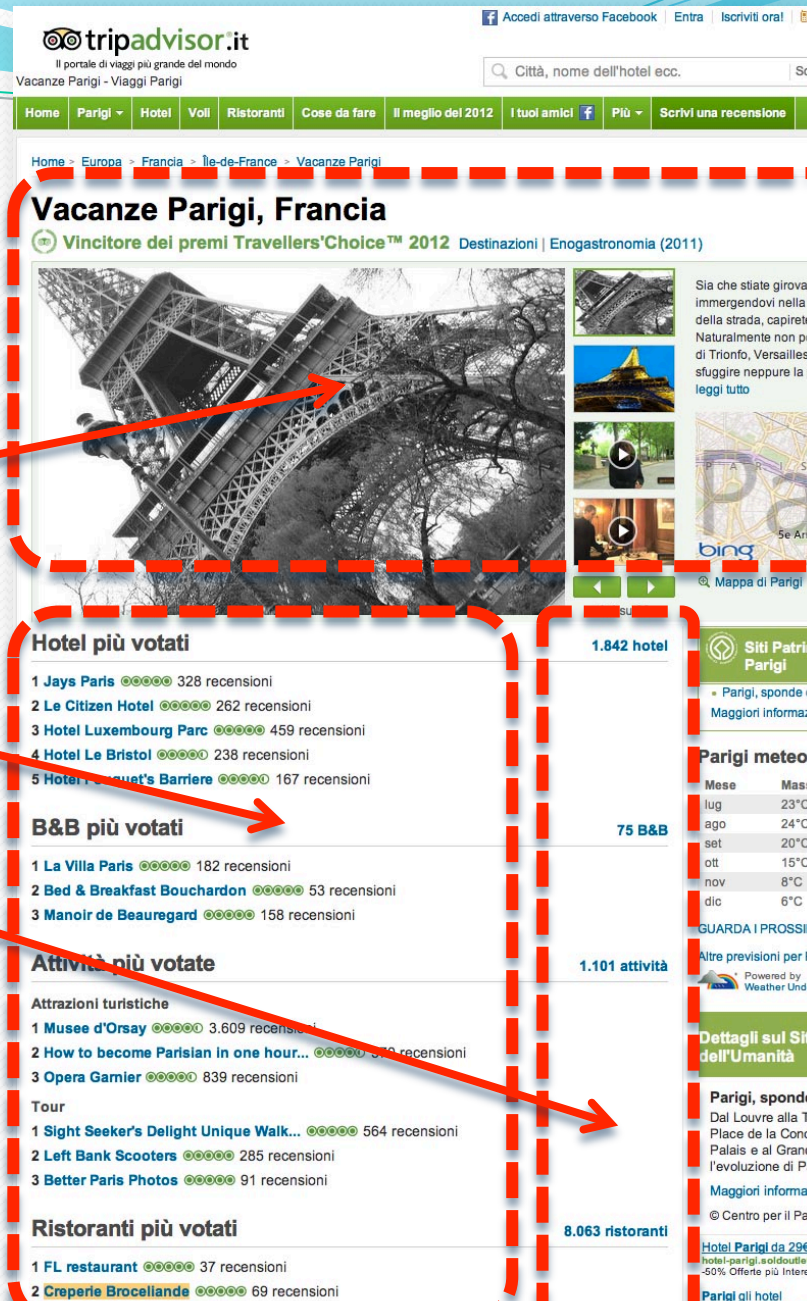
  - intermediate nodes are other list-of-locations
  - leaves are pages of the class location.

Page Classes (3)

- Class: Location

  Given a locationID, shows:

  - descriptive information
  - a set of most popular items of any type.
  - links to complete lists of item by type.

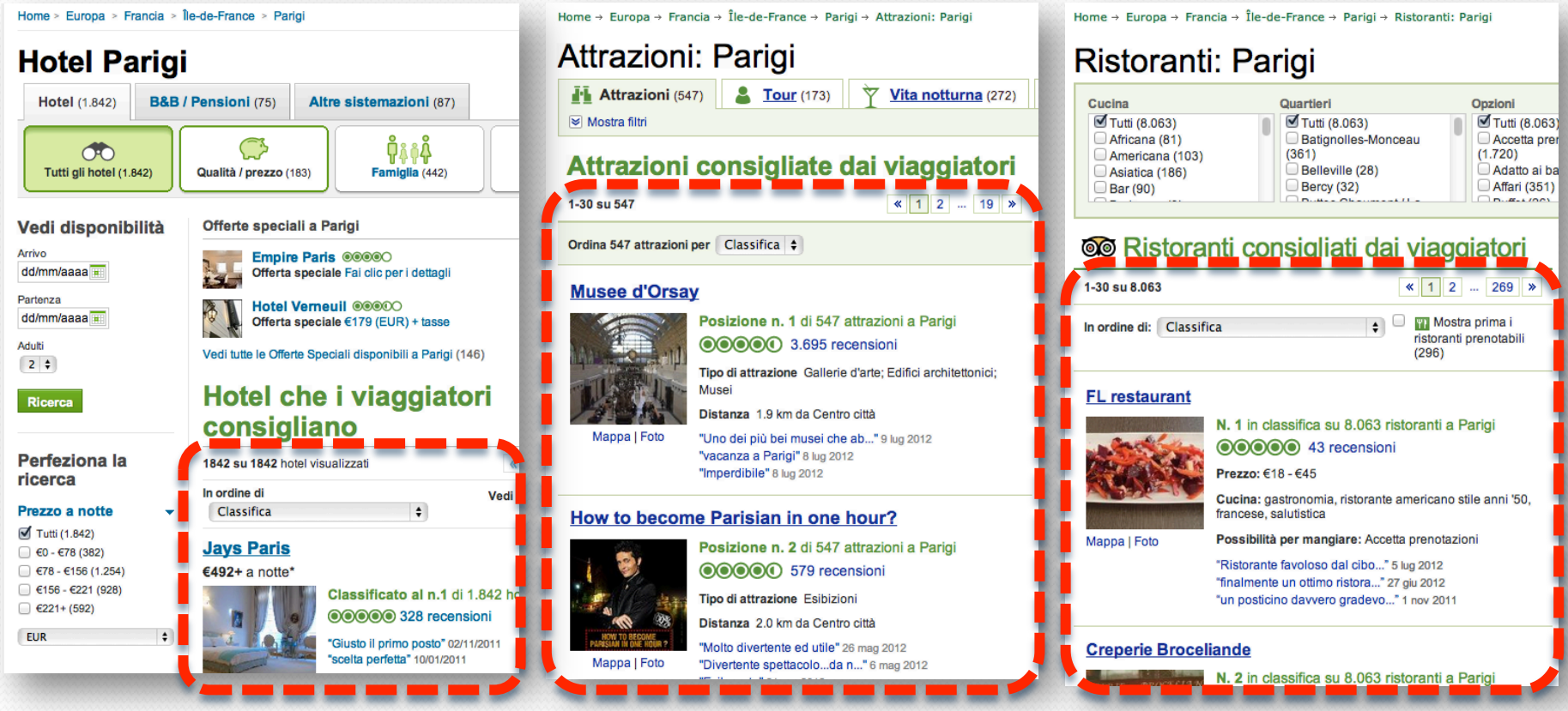


tripadvisor.it
Il portale di viaggi più grande del mondo

Page Classes (4)

- Class: List-of-Items

Given a location (locationID) and an Item type, shows a list of al the local Items of that type and some search ui.

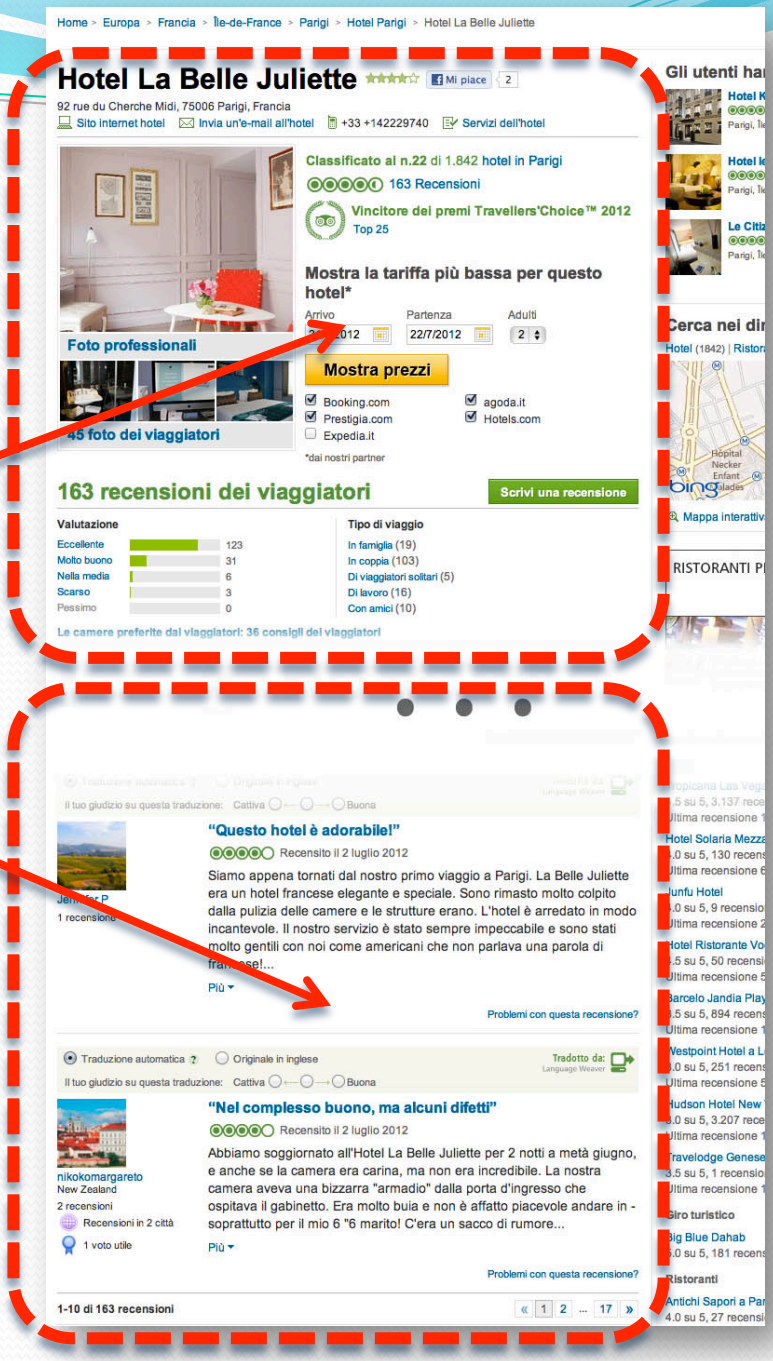


Page Classes (5)

- Item-Reviews

  Given an itemID, shows:

  - descriptive information and overall stats on the reviews.
  - The complete list of all the reviews, divided into pages. Reviews are shortened!



tripadvisor.it
Il portale di viaggi più grande del mondo

Page Classes (6)

- ReviewDetails

  Given a reviewID, shows:

  - Again descriptive information on the item to which the review belongs to.
  - The complete text of the review, with many additional data.
  - The other reviews of the item, in a detailed fashion

tripadvisor.it
Il portale di viaggi più grande del mondo

URL and page classes

- To distinguish different page classes it has been enough to look at the url:

http://www.tripadvisor.it/Tourism-g187147-Paris_Ile_de_France-Vacations.html
http://www.tripadvisor.it/Restaurants-g187147-Paris_Ile_de_France.html
http://www.tripadvisor.it/Restaurant_Review-g187147-d1991708-Reviews…

- This section identifies the kind of page
  - Accordingly, the next sections can refer to the location and the item identifiers.

tripadvisor.it
Il portale di viaggi più grande del mondo

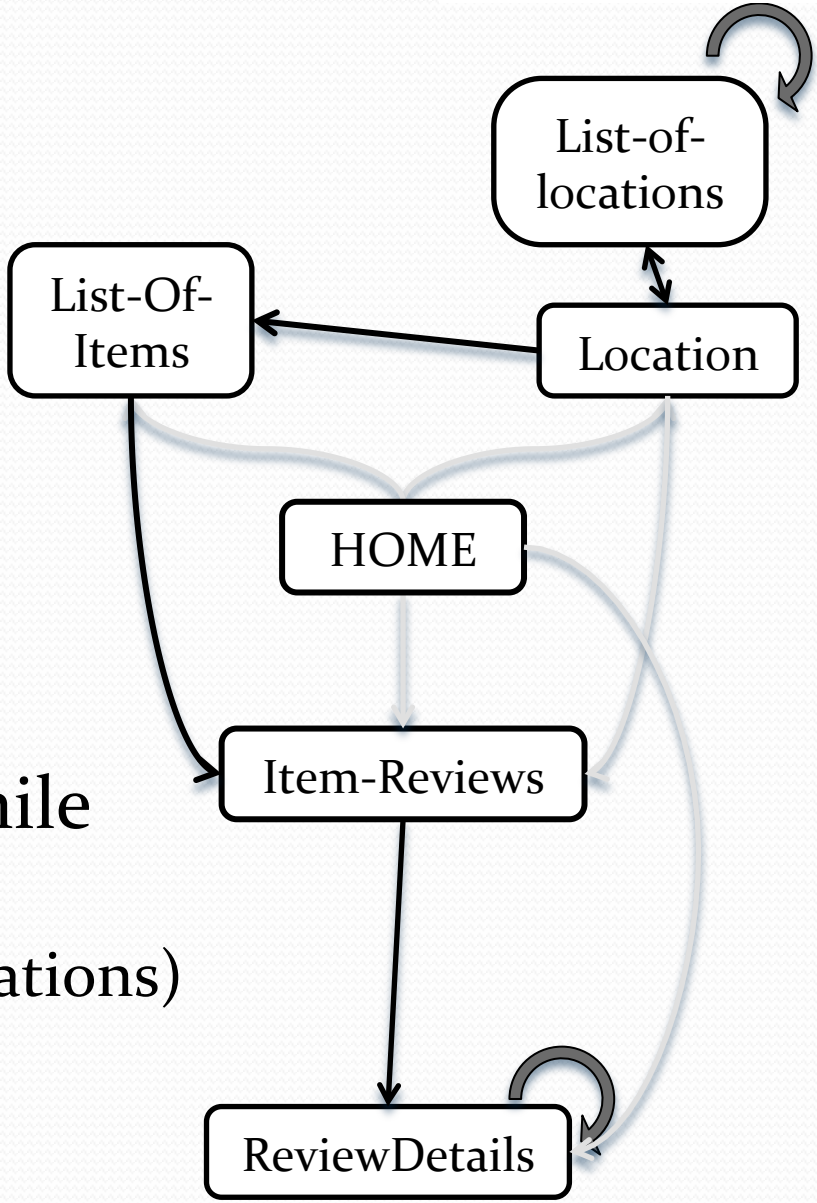
Mapping URL - Classes

URL MAPPING	Class	#estimated pages
/AllLocations	List-of-Locations	< 100,000
/Tourism	Location	~108,000
/Hotels /Restaurants /Attractions	List-of-Items	< 1,000,000
/Hotel_Review /Restaurant_Review /Attraction_Review	Item-Reviews	6,000,000
/ShowUserReview	ReviewDetails	60,000,000
…	…	?

tripadvisor.it
Il portale di viaggi più grande del mondo

SiteMap

Linking page-classes according to the contained links, we can obtain a SiteMap.



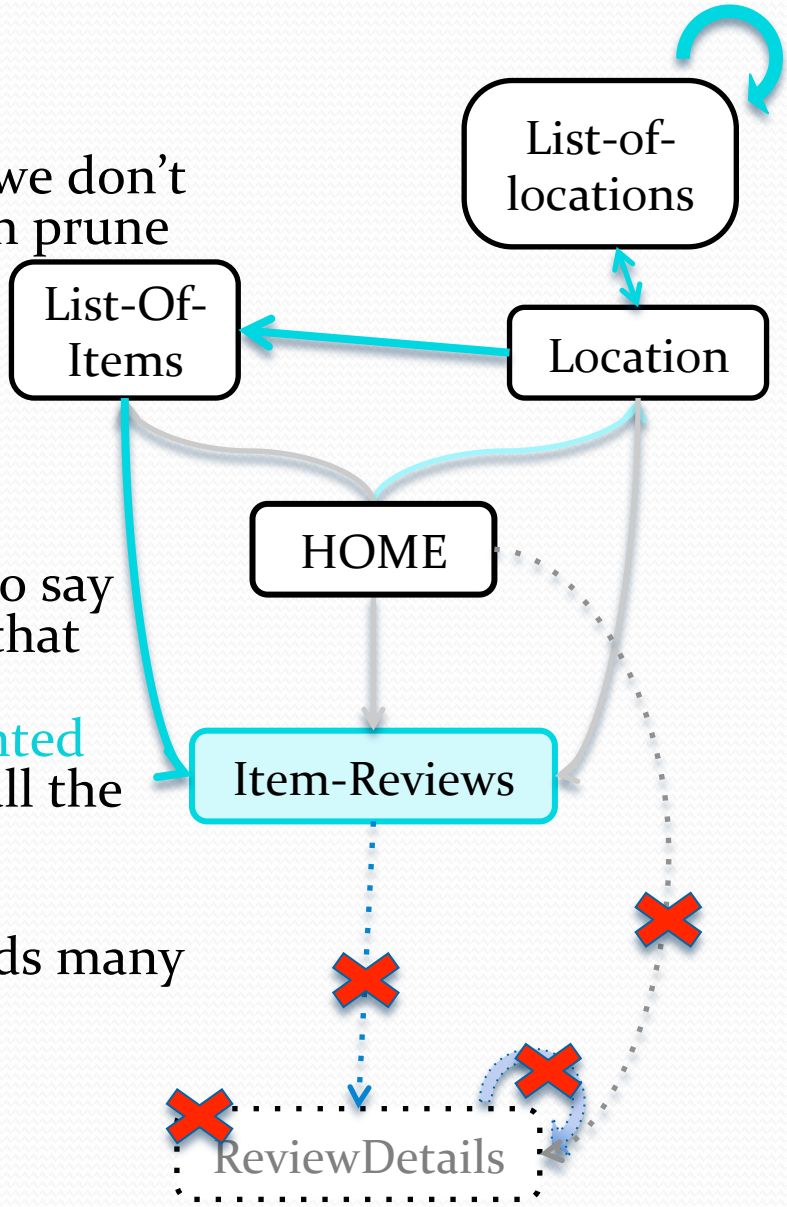
black links are exhaustive, while **gray** ones are partials.
(e.g.: home contains only some locations)

tripadvisor.it
Il portale di viaggi più grande del mondo

Analysis

- If we only have to extract date and rating, we don't need to reach ReviewDetails. Whe can then prune part of the graph.
- All the data that we need to extract are contained in Item-Reviews pages
- The structure of list-of-location allows us to say that it's enough to visit one single page of that class to b able to reach every location.
  From each Location, following the highlighted path, it is possible to reach the reviews of all the items.

This approach, unlike *casual crawling*, discards many pages improving Efficiency, at the same time ensuring Completeness.



tripadvisor.it
Il portale di viaggi più grande del mondo

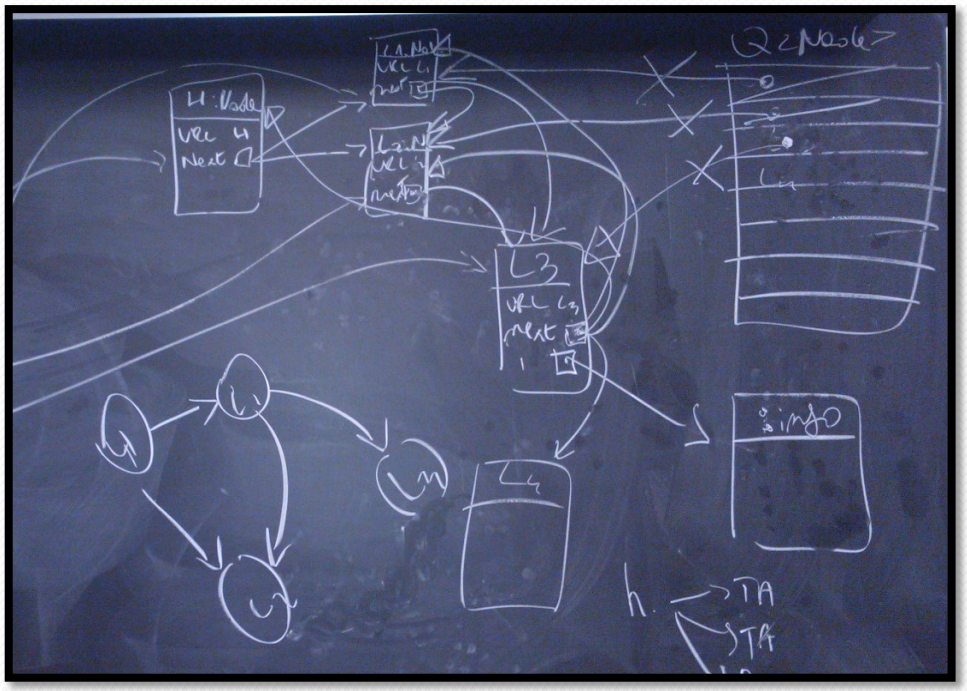
Estimates

- The total number of pages we want to visit lies around 6,500,000, of which:
- 6,000,000 are the pages to contain all the reviews (10 per page) to be extracted.
- 500,000 are pages to be “traversed” extracting only links to the items

tripadvisor.it
Il portale di viaggi più grande del mondo

Development of the System

- First steps…
  - Wanting to create a crawler from scratch, we built a first working version.
  - It allowed us to study the structure of a crawling engine and to face the typical involved problems and tradeoffs.



Problems

- Slow! (multithreading?)
- Manage many server response types
- Manage the delay between 2 requests
- Do not re-invent the wheel!

Development of the System(2)

- Crawler4j
  - A very useful library, clean and with all the tools
  - It allowed us to configure the crawling over the whole website with n crawlers at the same times, the extraction of some basic data and controlled minimum inter-request delay.
    (to configure everything: 10 minutes ☺)
  - We found in the structure of crawler4j the main ideas of our first crawler
  - We avoided to rewrite tons of code

Development of the System(3)

- Optimization based on the structure of the website
  - discarded useless links
  - parser created to extract data of interest from the pages of class: *item-reviews*
  - Extension of some of the classes of crawler4j
  - Fix of some navigation problems between the reviews
- Development of the persistency layer

tripadvisor.it
Il portale di viaggi più grande del mondo

Crawling: management & execution

CrawlHandler

1. CrawlHandler prepares the environment for the execution and management of crawling and the domain model, and starts crawler4j through CrawlController. In addition instantiates and interacts with the parser ItemReviewsPageParser.

CrawlController

2. CrawlController start threads of TACrawler instances, equal in number as indicated in the setup. Each TACrawler is created with a reference to CrawlHandler.

TACrawler

3. Each thread of TACrawler takes care of crawling TripAdvisor. It calls CrawlHandler to invoke the parser in case it needs to parse the reviews of a page of type Item.

CrawlHandler

- Requires three setup parameters
  - number of crawlers (threads)
  - Path to temporary folder for the frontier data
  - Minimum delay between two requests (in ms)
- Instanciates the specific parser object for the pages of TripAdvisor
  - Low coupling between crawler and parser
- Initializes the “seed” links
- Creates a Map<String,Item>, with the Item and the related reviews as soon as they are collected by the many crawlers.
  - Each crawler, knowing the CrawlHandler, can access that map to insert and update items.
- Starts the crawling using CrawlController, with the setup parameters and a reference to itself, for the creation of the crawlers.
- At the end of the crawling the domain model, contained in the map, will be saved in the database using the persistence layer.

CrawlController

- Crawler4j has a own CrawlController
  - Launch and management of the crawling algorithm of crawler4j
  - MultiThreading management
    - Launch of the requested number of crawlers, one per thread.
- It has been necessary to extend CrawlController from crawler4j so that threads are istantiated with a reference to the CrawlHandler (not existing in crawler4j)
- Our CrawlControllerOwn extends CrawlController allowing to pass the Handler to every new thread of generic type (T)
  - So this is not dependent on the specific crawler, given that in the constructor a reference to CrawlHandler is requested.

TACrawler

- This is the specific Crawler, of which many instances are launched, each one knowing CrawlHandler
- Extends the class WebCrawler of Crawler4j, overriding its main methods:
  - boolean **shouldVisit**(WebUrl url)
    - Given a url, extracted from the outgoing links, determines if this has to be loaded by the fetcher in a page and put in the frontier.
  - void **visit**(Page page)
    - Is invoked on a page extracted from the frontier.
    - if the page belongs to the class: Item-Reviews it calls ItemReviewParser for data-extraction. The data are then saved by CrawlHandler.

Parser (ItemReviewPageParser)

- While link extraction from the HTML is performed by crawler4j using Apache Tika, this parser works on the already loaded HTML.
- Has the goal of extracting data with a higher level of abstraction, only on some of the pages.
- Keeps a good level of low coupling from the crawler, while being the part of the system that is more specifically bounded to the internal structure of the pages of TripAdvisor.

Parser (ItemReviewPageParser)

- Based on the library Jericho HTML parser
  - Documentation defines it as:
    "… neither an event nor tree based parser…"
  - *Compared to a tree based parser such as DOM, the memory and resource requirements can be far better if only small sections of the document need to be parsed or modified. Incorrect or badly formatted HTML can easily be ignored, unlike tree based parsers which must identify every node in the document from top to bottom.*
  - *Compared to an event based parser such as SAX, the interface is on a much higher level and more intuitive, and a tree representation of the document element hierarchy is easily created if required.*

The collected data

- Collected data were saved into a relational DB
  - Easy to query
  - Few updates, indexes are “for free”
  - Allows Stored Procedures
- Persistence provided by the Hibernate framework
  - Independency from the specific DBMS
  - Only needs a few lines in the config file
  - adapts to the variations in the models

DB Schema

- The schemaof the database looks like a tiny datamart
  - Item *(itemid, crawlDate,description,type)*
  - Review *(reviewid,date,review, value)*
  - ItemReviews *(itemid,reviewid)*
    - With obvious foreign keys and constraints
- This schema allows, if needed, to assign more informations to the correlation between item and review

...extra tips for the DB

- Typical queries will be mainly based upon the date of the reviews for a given item:
  - Indexes are created to speed up the queries
    - they are convenient also since updates are not frequent

- Use of Stored Procedures for the queries
  - `showAverageReviewPerDate ( 'item_name' , 'start_date' , 'end_date' )`
  - `showReviewsPerDate ( 'item_name' , 'start_date' , 'end_date' )`

tripadvisor.it
Il portale di viaggi più grande del mondo

Execution times

- Execution times were a little too long for our laptops:
  - 16.000 items/hour -average
  - 100 hours, given by a total of 1.600.000 item (Restaurants, Hotel, Attractions)
- Too much, so we decided to execute our crawler on Amazon EC2:
  - 45.000 items/hour (about 3x faster)
  - About 18 hours for all the items of TripAdvisor

Test: effectiveness

- Useful to understand if:
  - only pages of type Item are parsed.
  - all reviews are extracted for each Item
- The test has ben conducted limiting the frontier expansion to a subset of the items, extracting outgoing links only from their pages and completing the extraction of all their reviews.
- As the seed page we set the page of a single Item
  - Different for every test.
- An aim was to find corner cases and possible intervention measures.

tripadvisor.it
Il portale di viaggi più grande del mondo

Test: effectiveness - results

- Setup parameters:
  - 500 crawlers
  - 2 ms delay, so a maximum of 500 requests/second
- Only Items have been parsed.
  - No false positive
- Test results were divided by Item type
- Note: Every test had a total duraton of about 10 minutes

tripadvisor.it
Il portale di viaggi più grande del mondo

Results - Hotels

- Hotel
  - I Test: missed 2 reviews out of 42.625, for a total of 155 different hotels. Accuracy: 99,99%
  - II Test: missed 3 reviews out of 33.125, for a total of 79 different hotels. Accuracy: 99,99%
  - III Test: perse 6 recensioni su 40.847, for a total of 188 different hotels. Accuracy: 99,98%
- Average: 99,99% of extracted reviews, for a total of 422 different hotels.

tripadvisor.it
Il portale di viaggi più grande del mondo

Results - Restaurants

- Restaurant
  - I Test: missed 20 reviews out of 15.339, for a total of 118 different restaurants. Accuracy: 99,87%
  - II Test: missed 0 reviews out of 9.012, for a total of 98 different restaurants. Accuracy: 100%
  - III Test: missed 20 reviews out of 15.313, for a total of 121 different restaurants. Accuracy: 99,87%
- Average: 99,91% of extracted reviews, for a total of 337 restaurants

Results - Attractions

- Attraction
  - I Test: missed 20 reviews out of 14.309, for a total of 87 attractions. Accuracy: 99,86%
  - II Test: missed 140 reviews out of 33.212, for a total of 103 attractions. Accuracy: 99,58%
  - III Test: missed 100 reviews out of 35.193, for a total of 130 attractions. Accuracy: 99,72%
- Average: 99,72% of extracted reviews, for a total of 320 attractions

Effectiveness - Conclusions

- Hotels were the type of item which was more in line with our original study
- From these tests some bugs were found in the extraction logic, due to inconsistencies in the pages of TripAdvisor.
  - A problem was due to the presence of some reviews without any rating (which is currently not allowed by TripAdvisor, but maybe was possible before)