Social Computing Midterm Project Fall 2011

Kayak Alerts for www

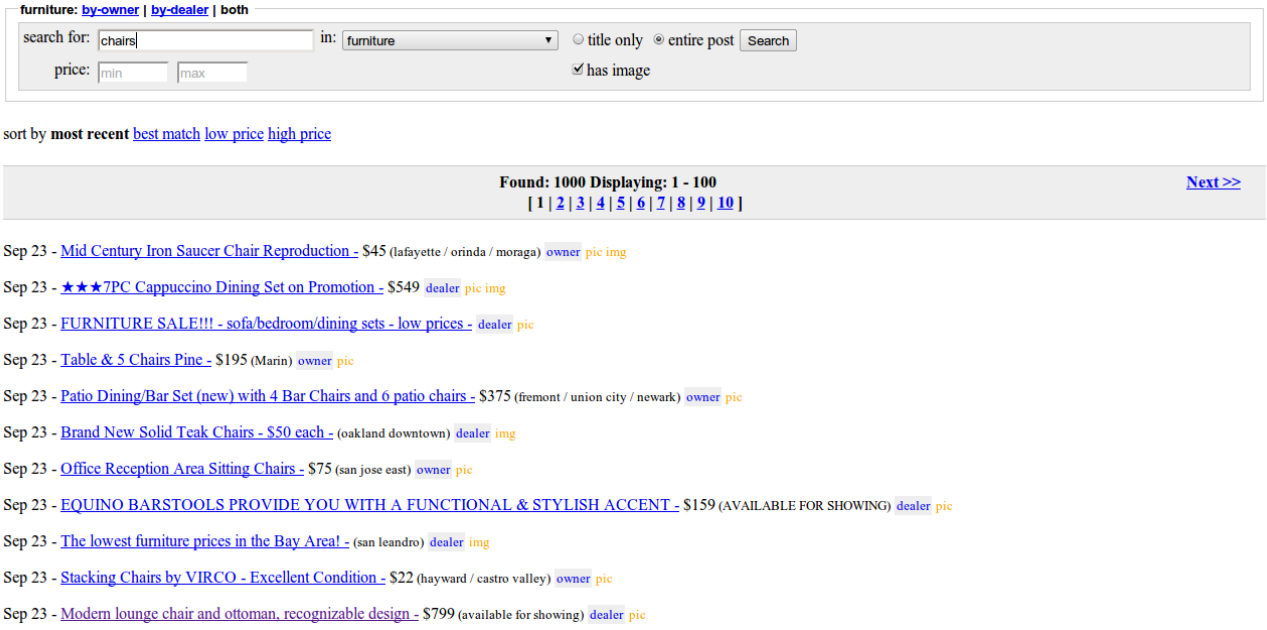
Philipp Gutheim - 11/01/2011

The notion of Cloud Computing has brought developers scalable, easy to use and cost effective services such as software as a service, storage as a service and computation as a service. A powerful application that is missing is human intelligence as a service, providing IT systems with disruptive capabilities of human intelligence in combination with computational power.

Crowdsourcing platforms, as marketplaces for tasks of just a few seconds that are completed by a global workforce of users who can earn small amounts of money, have the potential to enable such human intelligence service. However, major platforms such as Amazon's Mechanical Turk have significant drawbacks that limit their capabilities to become a service that is easy to access, reliable and delivers high quality responses. Instead, these platforms suffer from low quality responses by spammers, uncertain turn-around times and great integration efforts.

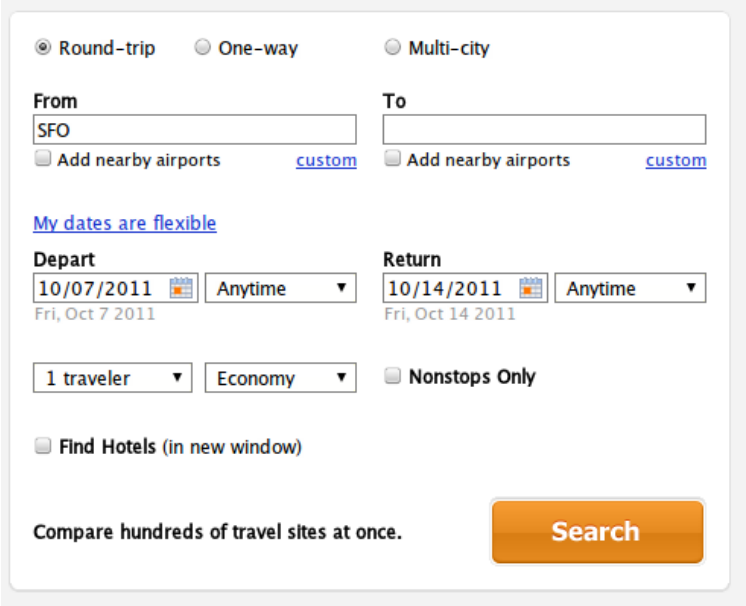
Kayak Alerts for www is a project that is designed to show case an additional management layer on top of a crowdsourcing marketplace that aims to alleviate developers from most of the drawbacks current crowdsourcing marketplaces have and enables a simple though powerful automation of tasks that computers currently cannot (or not reliably) solve. The show case illustrates how an application such as intelligent web research (think of it as a Google powered by humans) can be made available through a (1) simple user interface as well as (2) easy to use python library to enable developers with the power to make an API call to a human.

Intelligent Web Research is a field that still requires significant amount of human intervention. For instance, if somebody is searching on Craigslist for a set of living room chairs (hence at least 4 or 6 chairs), the user will need to search through more than 100 listings, and spending large amount of time and effort to find suitable listings.



Craigslist is just one example out of a large problem sets. Developers have challenges to automate data retrieval on the web that is ambiguous though essential to their software.

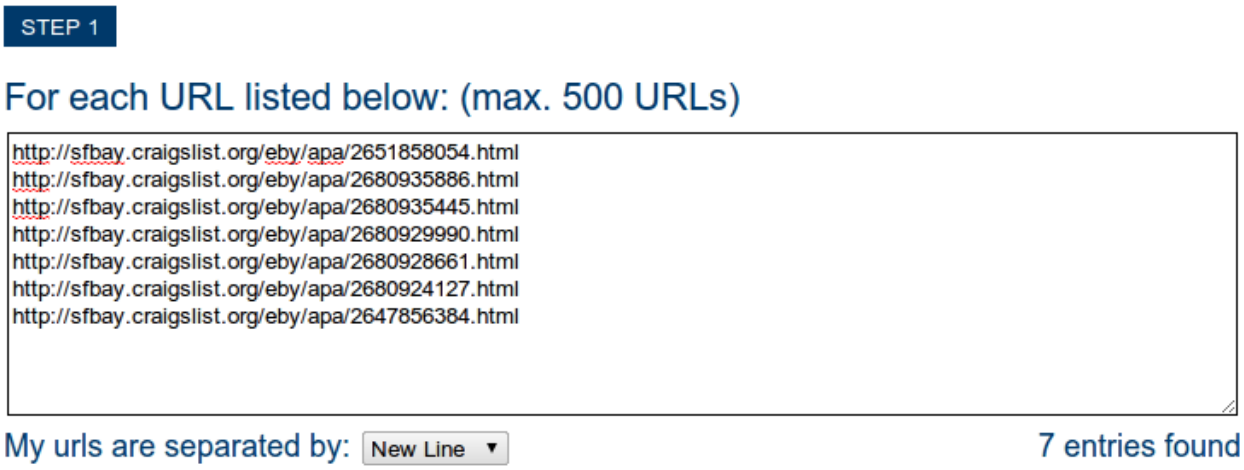
The approach taken in this show case is threefold. First, a first prototype for a user interface for the intelligent web research service (e.g. for businesses) has been designed. This would allow anybody who would need to extract certain information from a large number of websites to only copy & paste his/her urls and select the required information and then sit and wait until they will receive the results in a couple of hours. The design is inspired by the simple search interface through which Kayak.com enables users to query are amounts of (structured) flight schedules.



Screenshot of www.Kayak.com search field

This prototype has been designed as minimalistic as possible, asking the user for a

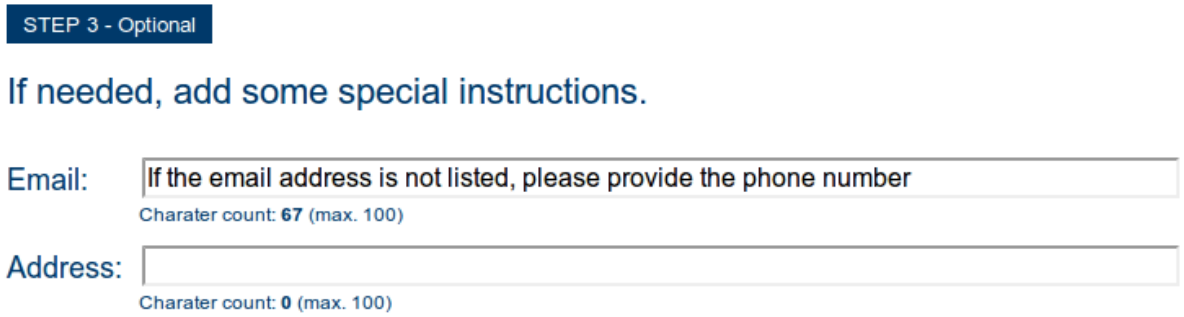
(1) resource



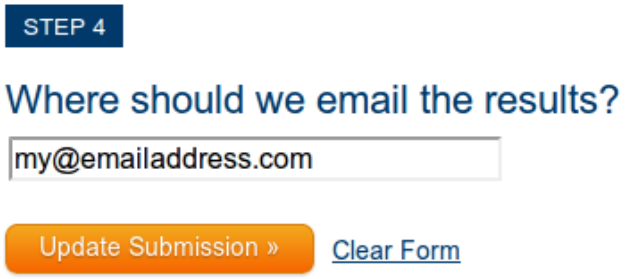
(2) sample information the user would like to extract

(3) optional instructions in addition to a set of predefined (optimal)

instructions



(4) the contact email address of the user



Although this prototype already presents a simple interface, it has certain limitations. In its current form, it does not support open ended questions but requires the user to provide a set of urls. Between the midterm and the final submission of this report, the interface will be adjusted to allow open-ended queries such as “I need a list of email addresses and hourly rates of 100 Yoga trainers in San Francisco”.

In the second step, the system to store a user’s submission (urls, required information and instructions), to generate tasks based on them and to integrate the back-end with a crowdsourcing platform has been completed. In addition, first means to increase the quality through answer format standardization has been implemented as well. Based on that, simple double entry (at least two people have to agree on the a task for an answer to be valid) and gold standard system will be put in place until the final report to increase the answer quality significantly.

The third step is to enable an API call to a human. This means to abstract from the crowdsourcing platform API, which allows calls to a marketplace, and to build an API on top of the show case layer. This is currently done by means of the tastypie library. Similarly to the simple user interface, it allows developers to post a web research question to a human and receive an answer shortly. The current call looks similar to this python command to the RESTful POST request with a json encrypted dictionary:

```
import httplib2,json
url = "http://www.HumanIntelligenceInTheCloud.com/submit"
connection = httplib2.Http()
connection.add_credentials(yourusername,yourpassword)
headers = {"Content-Type": "application/json"}
query = dict({
       "question": "your instructions here" ,
       "resource":"http://anylinkhere.com",})
response, content = connection.request(url, method = "POST", body =
json.dumps(query), headers=headers)
```

The future goal at the end of this course is to provide a simple python wrapper that allows you to do the following:

```
import human
paintings = human.ask("What are the 5 most beautiful paintings of Van
Gogh?")
```

or

```
import human

emails = human.search("Emailaddress", 100, "Yoga Teachers in San
Francisco")
```

The challenge of this project is predominantly in (1) guaranteeing accuracy of the results, while enabling a broad array of applications, and (2) buliding a system that is robust (can cope with insufficient or incomplete answers) and designed for the asynchronous nature of information flow.

Since part of the quality controls rely on formatting of the input, it is a challenge to design a system that would theoretically provide these error controls for a variety of information, e.g. physical addresses are more challenging. The key here will be experimenting with the system and putting quality features in place that are as agnostic to the form of information as possible. Similarly, insufficient, inadequate or incomplete answers may cause high complexity and low accuracy levels. There exists a variety of examples: For instance, users might have misunderstood the instructions because the instructions were ambiguous, an answer is correct though incomplete, the information do not exist or cannot be obtained, etc.

The goal of this project is to investigate whether we can make a significant step towards the vision of a human intelligence as a service that is as easy as “human.ask(‘Questions’)” and to show case a potential human intelligence web research application.