Information extraction for classified advertisements

A project in the course
EDA171/DAT171 Language Processing and Computational Linguistics

Johan Eriksson

Abstract

This paper presents work done in project form in the course Language Processing and Computational Linguistics given at Lund School of Technology during the fall of 2005. The goal of the project has been to implement a simple information extraction tool from the domain of classified advertisments about apartments.

Introduction

The desire to find things on the web has lead to the development of search engines like Altavista, yahoo! and many more. The basic idea of these is to allow the user to find pages that contain certain words and do not attempt to understand or make sense of any web pages. Whenever they do take steps towards understanding web pages they risk losing their generality. Google, for instance, did not support stemming in the beginning but does so now and it is described here: http://www.google.com/help/basics.html#stemming . Searching for ”google stemming” currently, 2006-01-11, gives 716,000 results and browsing through the results one can see that there are a lot of discussions surrounding the peculiarities of Google stemming.

Another category of search engines is the shopping agents who index web shops. These are specialized to index the semistructured data of web shops where product data and prices usually are displayed in table-like structures. A technology that can be used for such applications is described in ”A Scalable Comparison-Shopping Agent for the World-Wide Web”[3]
Examples of sites using this technology include Froogle(http://froogle.google.com) and Pricerunner (http://pricerunner.com).

A third category, which index text that is unstructured, is what this paper is about. An example of this is http://eniro.jobsafari.se which indexes job ads. I have chosen to work with apartment ads from the site www.blocket.se which is a Swedish categorized ad site. I have implemented a simple information extraction engine using hand crafted patterns.

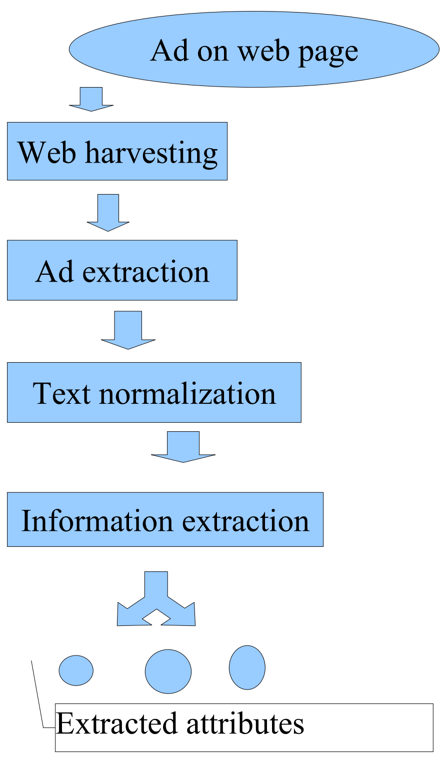
Implementation

I chose to split my system into three different logic parts: the robot that gathers ads from the site blocket.se, the web interface where a user can make queries and the information extraction engine.

From ad to information

Figure 1 shows how the information from an ad in a web page becomes extracted. The differrent steps are then described in detail.

Figure 1: From ad to information



Web harvesting
An ad starts out being a web page that is gathered by the robot.

Ad extraction
The actual ad is extracted from the markup of the web page.

Text normalization
Ad text is split into chunks that are well fitted for running patterns against. I choose to split the text at the sentence level.

Information extraction
Regular expressions are run against the sentences in an ad to retreive the desired information.

Nature of ads

Sometimes the data of interest have what I would like to call labels while other times context is used to tell the reader about the data. To exemplify we can look at apartment rental ads. My translations of the examples appear in paranthesis after each sample and might not be exact, but they suffice for the point I am trying to make.

For "Hyran ligger på 3207" (The rent is 3207) or "pris 4136 Kr/mån" (Price 4136 crowns/month), the labels are "Hyra" and "pris" respectively.
In other cases the data is described by context like
"2100:-/månad (förstahandskontrakt) inflytt 1/10" (2100 crowns/month (first hand lease) available 1/10) where the fact that it is a price/month and the mentioning of type of lease tells us that it is about rent. Another example is the very short: "900kr/mån" (900 crowns/month) which really only consists of a measure and a unit. The available information, out of context, only tells us that it is a price, but in the context of an ad for an apartment an unlabeled price is likely to be the price of the apartment.

In my observations it seams like if a price refers to something other than what one would expect for a typical item of this kind it usually carries a label: "Jag kommer börja arbeta i Köpenhamn och kommer ha en månadslön på ca 25 000 danska" (I will start working in Copenhagen and I´ll have a monthly salary of about 25000 danish[crowns]), which carries the label “månadslön” or “Hyra 4828:- Deposition 7500:-” (rent 4828 crowns deposit 7500 crowns) which uses the labels ”hyra” ”deposition”.

It seems like for different kinds of ads, or maybe texts to be more general, there exist some kind of defaults which tell us what unlabled data is about. If you look at ads for bikes, it seems like the only time you would find any mentioning of the number of wheels is when it is different from the default 2-wheel. It would be interresting to examine to what extent an information extraction system needs to have knowledge about such defaults and what influence cultural differences have on this matter, but there was no time to dwell deeper into this area.

regular expressions sub language

In an attempt to maintain some order in the chaos that emerged from testing a lot of regular expressions and having cleanup or transformation code that should be run after a successful match I experimented with creating a new regular expression definition language. The main features are that such a regular expression can

- inherit from another regular expression
- define cleanup/transformation
- add things that must match

Example. Price pattern:

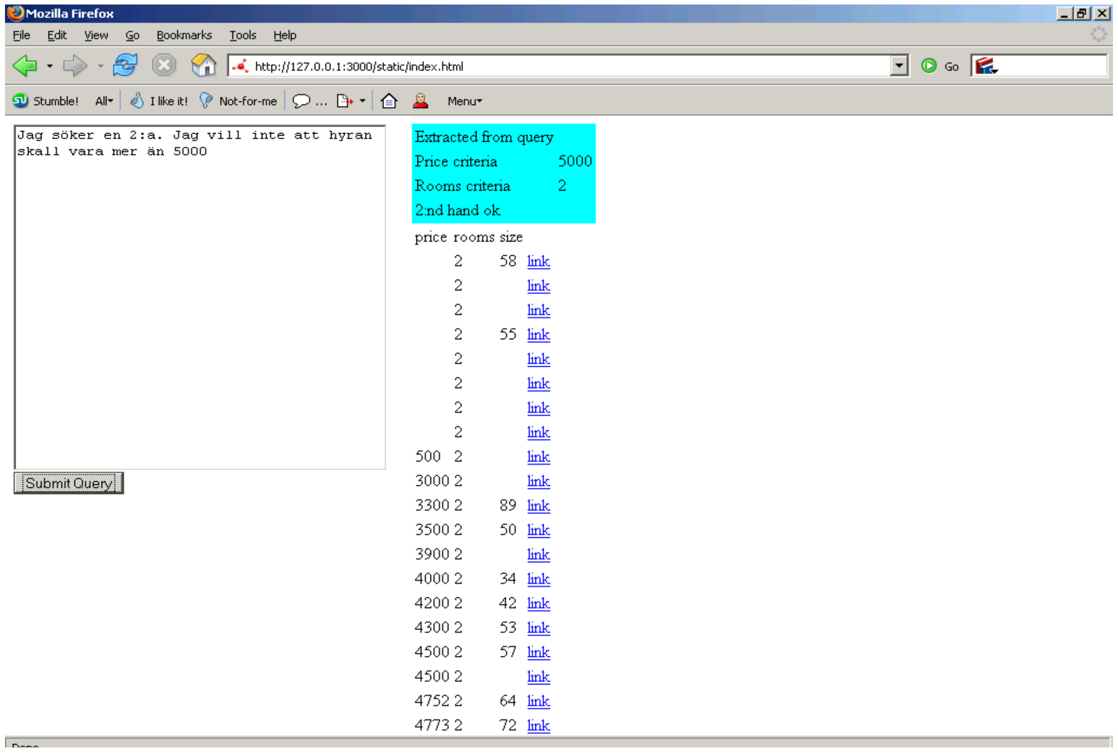
```
# base pattern 'number_free'
'number_free' => {
  'pattern' => q{\b(\d+[\d., ]*)},
  postprocessing' => [q{=~s/(\s|\.)+//g}]
}



# pattern 'rent' which inherits from
'number_free'
'rent' => {
  'ISA' => 'number_free',
  'preceeded_by'=>q{hyra.*?}
}

# pattern 'price' which also inherits
from 'number_free'
'price' => {
 'ISA' => 'number_free',
 'followed_by'=>q{ ?(kr\b|:-|\/[md])}
}
```

First I define the pattern 'number_free' which matches a wide range of numbers including “2 000” and “2.000”. I also define a some post processing which removes space or dot characters turning both “2 000” and “2.000” into ”2000”.

Then I create two sub patterns of 'number_free'. The first sub pattern is able to match the kind of prices that are preceeded by a label, as mentioned in 'Nature of ads'. It is called 'rent' and is a 'number_free' preceeded by the label 'hyra'. The second pattern can match a price. It is called 'price' and also inherits from 'number_free' and narrows the matching by saying that 'price' is a 'number_free' followed by a (Swedish) money unit.

Web interface

The web interface consists of a text box in


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://127.0.0.1:3000/static/index.html
Go
Stumble! All I like it! Not-for-me ... Menu
Submit Query

Jag söker en 2:a. Jag vill inte att hyran skall vara mer än 5000

Extracted from query	
Price criteria	5000
Rooms criteria	2
2:nd hand ok	

price	rooms	size	
	2	58	link
	2		link
	2		link
	2	55	link
	2		link
	2		link
	2		link
	2		link
500	2		link
3000	2		link
3300	2	89	link
3500	2	50	link
3900	2		link
4000	2	34	link
4200	2	42	link
4300	2	53	link
4500	2	57	link
4500	2		link
4752	2	64	link
4773	2	72	link

Figure 2 The web interface

the left side where the user can enter a query and then press 'Submit Query' after which the results will be displayed in the right part of the browser. It is shown in figure 2.
The results page starts with displaying the values that have been extracted from the query and continues with displaying information from the 'For rent' ads that match the query. The web application was created using the Catalyst[4] web framework.

Evaluation

Scoring

I have used a simple way of scoring, namely for each attribute to be extracted:
+1 for correct value
0 for incorrect value
The learning set has not been included in these test runs.

For rent ads

The following table show the results for the 'for rent' ads.

attribute	**correct/ total**	**percentage correct**
price	114/133	85%
number of rooms	91/133	68%
size	113/133	85%

Wanted ads

The following table show the results for the 'wanted' ads

attribute	correct/ total	percentage correct
price	109/115	95%
wanted rooms range	71/115	62%

"Price" here is maximum price, "wanted rooms range" is the number of rooms the advertiser wants expressed as a range like "2 to 4".

Comment

The closer you desire to come to being able to extract all information correctly using this method the more domain and language knowledge you have to add to the system.
This fact makes it not feasible to use this method on a larger scale(across many different domains). When building this system I came up with the idea that it would be nice to have a system which discovers the patterns itself and that maybe something like n-grams and manual tagging could be used to find the kind of labels I mentioned before.
This might be a bit naive and would not solve the problem for data that does not carry labels, but it could be a starting point for diving into this problem.
A simple example would be "The rent is $500" and "The rent, including electricity, is $1500". Having a corpus of such sentences and the price tagged, you would probably get "the" and "rent" as candidate labels. The system could then try "the", which would probably get too many false hits since it is such a common word and "rent" which would prove to be a good candidate.

Something similar had already been done and one example I found was the AutoSlog[1] system which "automatically builds dictionaries of extraction patterns" and "uses an annotated corpus and simple linguistic rules". Another would be the TIMES[2] system.

Conclusion

For a simple information extraction task within one or few domains, handcrafting patterns might be the right way to go. But as the task grows larger and the domains increase the need for more sophisticated tools emerges like the aforementioned AutoSlog and TIMES.

References

[1] Ellen Riloff:Automatically Constructing a Dictionary for Information Extraction Tasks(http://citeseer.ist.psu.edu/riloff93automatically.html)
[2] AMIT BAGGA, JOYCE CHAI and ALAN BIERMANN: Extracting Information from Text (http://citeseer.ist.psu.edu/588088.html)
[3] Robert B. Doorenbos, Oren Etzioni, Daniel S. Weld: A Scalable Comparison-Shopping Agent for the World-Wide Web (http://citeseer.ist.psu.edu/doorenbos97scalable.html)
[4]Catalyst web framework http://catalyst.perl.org/