MetaBrain: Web Information Extraction and Visualization

João Teixeira **Gabriel Barata** **Daniel Gonçalves**
Department of Computer Science and Engineering, IST
Av. Rovisco Pais, 1000 Lisbon
{joao.teixeira,gabriel.barata}@ist.utl.pt, daniel.goncalves@inesc-id.pt

ABSTRACT
Nowadays, the web is a huge source of information on different branches of knowledge. This knowledge, however, is dispersed across many sites, making it difficult to interrelate and understand. In the past few years some approaches have been developed to ease the extraction of this information, from Open Information Extraction to simpler data mining. Usually these solutions work as standalone applications and are developed from scratch and are brittle, very sensitive to changes in the data sources. This makes it difficult for the final user to fully explore the potential of using different algorithms together to better extract and analyze information. In this paper we propose a new approach where users can create their own personalized information extractors and visualizations, without needing to type a single line of code, in an easy and highly flexible manner using a special-purpose interface. Since raw data is most times difficult to understand, we also study how the user can create customized visualizations of this extracted data with low effort. A prototype of this concept, MetaBrain, has been implemented and tested. Preliminary heuristics evaluation, demonstrate favorable results for the concept.

Author Keywords
Information Extraction, visualization, user interaction.

ACM Classification Keywords
H.5.2 User Interfaces - Graphical user interfaces (GUI), H.5.m Miscellaneous.

INTRODUCTION
The versatility of the web is also its biggest problem. Since anyone is free to create their website in any way they want, there is no unifying structure for all this information. More than a huge repository of knowledge, the web contains a whole set of hidden implicit information. The way people express their thoughts reflect an unconscious collective of trends and patterns which are not obvious at first sight. What color does the Internet relate to the term apple? Surprisingly, white is the color that more frequently co-occurs with apple in web pages, next to red and green. Apple Inc. and Snow White may be to blame for this.

Traditionally, Information Extraction (IE) focuses on extracting information from specific pre-defined domains. Changing domains implies that new extraction rules need to be manually created, making it hard to scale. Manually querying search engines in order to extract large quantities of information is also not the right approach, since it is tedious and error-prone as pointed out by Etzioni [6]. A possible solution to this problem is the use of Open Information Extraction [2], which states that a high amount of relashionships are expressed through a compact set of relation-independent lexico-syntactic patterns. This is only one of several techniques [3,5,7] which allow the extraction of information from the Web using only statistics and probabilities.

Although many new tools for web IE have recently appeared, these tools are usually designed to use a single type of IE technique with no possibility of interaction with others. It may be in the best interest of the user to use different IE techniques simultaneously, thus discovering hidden and unexpected patterns in apparently unrelated data. For example, the possibility to automatically extracting a list of Operating Systems and see how popular each one is on different search engines or social networks, for different kind of users. Another problem found in these tools is that most are developed from scratch. Currently, there is no unified framework with different IE modules available for programmers or other users to use as a basis for their IE tools. Also, state of the art tools like TextRunner [1] lack advanced search options, like the selection of search engine to use, or the possibility to extract the retrieved data. These options may be important for advanced users.

Our research aims at finding ways for normal web users to access the collective unconscious that is the Internet. Given the giant number of possible extraction scenarios this can be a very complex and difficult task. Our efforts were directed at creating the best interface to make this task as easy as possible. Since raw data from these techniques, at times, is difficult to understand, we also analyzed several information visualization techniques, from simple bar charts to hierarchical tree-maps, with the objective of creating a good and easy way for the user create and export their customized visualizations.

In the next sections, we detail how we extract information from the web. Then we explain our design and interaction decisions for our solution prototype. This is followed by the result analysis of the prototype's heuristics evaluation and finally. We conclude with our final remarks and talk about future work.

CREATING CUSTOMIZED IE SOLUTIONS

There are different approaches to extract information from the web without the use of complex natural language parsers. Different algorithms use different features to extract the information. Generally, we find three different classes of approach that use: number of results found for a given query [9]; lexico-syntactic patterns [5,6]; and word co-occurrence [8]. Next we'll see how we can use these different classes together to create customized IE tools.

Selected Information Extraction approaches

The *number of results* can be used as a way to identify the popularity of one or more concepts on the Internet, and also to measure the validity of extracted data. For example, if "fishing water" has more results than "fishing wall" then fishing is probably more related to water than to a wall.

By using *lexico-syntactic patterns* like `C{,} "such as " IList`, where C is a concept and IList is a list of instances from that concept, it is possible to generate special queries to use in search engines that will be able to map concepts to instances or instances to concepts.

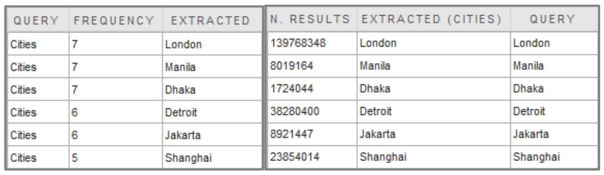
Recent works have been created to prove the validity of using *term co-occurrence* to do opinion mining [7,8]. With the rise of micro-blogging usage, it is now possible to more easily extract the general Internet opinion of a given concept by looking at what words co-occur with that concept.

Putting It All Together

Each one of these approaches is a way to extract a different type of information, so it would be good if we could use them together or alone, depending on what we want to extract. We can think of each one of these as a different search module. If we would like to extract a list of cities and then check their popularity online, instead of manually executing two different searches it would be good to create a single search query for the whole extraction.

Because these modules are domain independent it's a matter of defining a way to direct a module's output to another's input. In order to do this we can standardize all the three modules' main input as a single query parameter and their output (result set) as a table (Figure 1), were the rows represent the different extracted information and the columns represent the extracted information (primary column) and some auxiliary attributes of the extraction.

Looking at only the primary column of a result set we get a list of results which can be iterated by another search module as its input parameter. This way it is possible to easily create multi-level search queries. Figure 1 also shows a result of a multi-level search.



QUERY	FREQUENCY	EXTRACTED
Cities	7	London
Cities	7	Manila
Cities	7	Dhaka
Cities	6	Detroit
Cities	6	Jakarta
Cities	5	Shanghai

N. RESULTS	EXTRACTED (CITIES)	QUERY
139768348	London	London
8019164	Manila	Manila
1724044	Dhaka	Dhaka
38280400	Detroit	Detroit
8921447	Jakarta	Jakarta
23854014	Shanghai	Shanghai

Figure 1. Left: result set for an extraction of city instances. Each row represents an extracted city, which is presented on the *Extracted* column, the table's primary column; Right: result set for the number of results found for the different cities extracted on the left table.

A prototype library was implemented with these capabilities and also the possibility to customize each search parameters (thresholds, search engine, etc.). Several search engines can be used, including social networks. A modular approach was used to create this library in order for it to be easily expansible with new search engines, IE algorithms, or simple web service APIs. Also, since some IE modules need to sometimes perform thousands of search queries, a cache system was developed to make the searches faster when possible. The direct use of this library still requires programming skills. Hence, we developed a special-purpose interface, Metabrain, which allows even non-programmers to perform IE and visualization tasks in a more natural way.

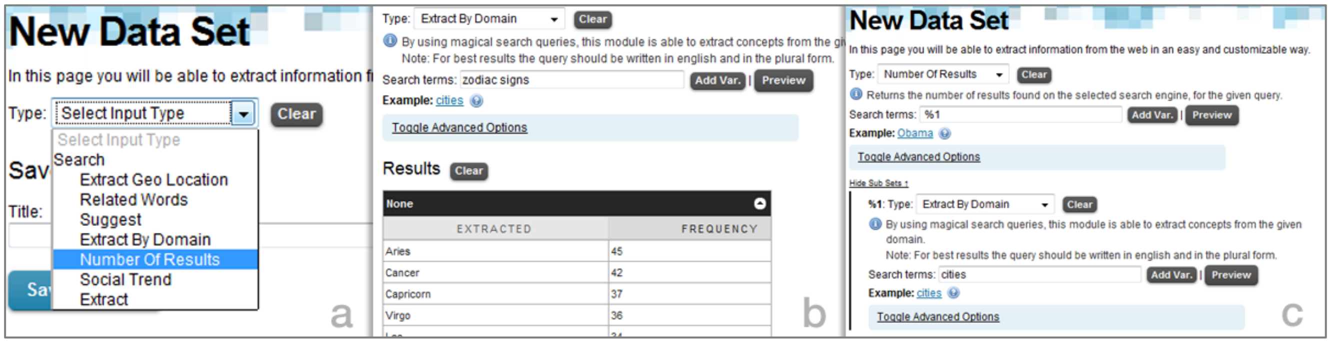
METABRAIN PROTOTYPE

With the library complete, we started looking into how we could create a GUI simple enough to allow regular Internet users to interact with it, without neglecting all the advanced options required by expert users. With this in mind, we decided to use HTML and Javascript, in order to create a very dynamic interface with standards-compliant technology. Also, it is easy to connect with our Python library. We want not only the users to extract information but also for them to create meaningful visualizations of the raw data. All these visualizations were implemented using the Protovis framework [4].

Data Set Creation

Since the use of IE tools may not be common to most users, an effort was made to simplify every possible step of the extraction process, without disregarding the needs of advanced users. By default all customization options are hidden, although easy to access, and preset to a default value. This way the only thing needed is for the users to select what they want to extract. They can choose, and at any time change, between the different available extraction modules. These modules allow for the same type of IE previously discussed plus easy access to public API services, such as location to geographic coordinates and search engine suggestions. Each module is accompanied by a quick description of its purpose and a series of possible input examples with explanations.

The design philosophy we follow is to only show relevant information in the interface so, by default, there is only one input section visible to the user. This reduces the visual noise needed to complete his task. For a simple one level IE



In this page you will be able to extract information f

Type: Select Input Type

Type: Extract By Domain

By using magical search queries, this module is able to extract concepts from the gi
Note: For best results the query should be written in english and in the plural form.

Search terms: zodiac signs

Example: cities

EXTRACTED	FREQUENCY
Aries	45
Cancer	42
Capricorn	37
Virgo	36

In this page you will be able to extract information from the web in an easy and customizable way.

Type: Number Of Results

Returns the number of results found on the selected search engine, for the given query.

Search terms: %1

Example: Obama

%1: Type: Extract By Domain

By using magical search queries, this module is able to extract concepts from the given domain.
Note: For best results the query should be written in english and in the plural form.

Search terms: cities

Example: cities

Figure 2. a) List of available extraction modules for the first input. b) Example of an extraction of the zodiac signs. c) Example of a multi-level search query. The final result will be the popularity, on the selected search engine, of every extracted city.

the process is very straightforward: select the IE module to use, input the query parameter and search. For example, if the user wishes to extract from the Internet a list of zodiac signs, he just needs to select the *Extract by Domain* module and use "zodiac signs" as the search query. By doing this, a list of extracted zodiac signs is presented to the user, as seen on **Error! Reference source not found.**b.

If the user wishes to create a multi-level search query, the interface will evolve during the process, along with the user's needs. If, at any time, the user chooses to use the result of one search as a term in another, the interface will dynamically add a new input section where the second search query can be defined. These secondary input sections are called variables and have the form of %1, %2, etc. Graphically, every new query to obtain the values for each variable appears below the one in which it is used, and one level deeper on the interface (**Error! Reference source not found.**c). This helps users to effectively resort to several variables at once without getting lost or confused.

In order to minimize the number of errors and not waste the user's time in vain, before initiating the final search query, which may take from a few seconds to minutes or hours, it is possible to do a preview search in a smaller scale. This way, the user gets a quick glimpse of the kind of results returned by the current query and can make any adjustments necessary before starting the real long search.

To increase the possibilities of query creation it is also possible to create Data Sets by importing users own personal data (CSV) through our prototype. Before the data is imported it is scanned and MetaBrain tries to guess what type of data is in each column (text, numbers, coordinates, etc.) Our guesses are then shown to the users so they can confirm and make any changes necessary. We'll discuss the importance of this type of information in the next section.

Visualization

Now that we have a good and flexible approach that allows even non-programmers to do customized IE from the Web, the next step is to provide them with the possibility to visualize this information in a more meaningful way than the one provided by simple tables. We started by identifying a set of requirements we would like the visualization creation process to follow:

- Since the table of extracted information has multiple columns, the user must be able to choose which columns she or he wants to visualize.
- The user should be able to choose from several different types of visualizations, from graphic bars to sunbursts or even maps;
- All the visualizations must have its set of configuration options, bar width for the graphic bars, palette color for the sunbursts, etc.;
- During this process it must be easy to change between different visualization types maintaining the users previously selected preferences, if these are applicable to the new type.
- The user must be able to always preview the visualization being created. Configuration changes to the current visualization should be applied instantly, without the need to refresh.

Taking all these requirements into account, we decided to divide the visualization process into 3 steps: choose data to visualize (which columns); choose the visualization type; preview and configure the visualization.

To address the first requisite we decided to let the user choose which columns to visualize by using a drag and drop metaphor. On the left side of the application a vertical list of names is visible. These are the names of the different columns existent in the selected data set and they are divided by the type of data they contain, this division makes the column selection easier for the user. On the right side of this list are two large horizontal boxes, representing the visualizations axis. The user is then able to drag columns from the left list and drop them in the axis input boxes. During the drag procedure these boxes are highlighted, making the user aware of valid drop inputs.

We decided to use two axis after concluding, in a study, that all the different visualizations we wanted to implement required at least two degrees of freedom.

The available visualizations list starts empty. While the user makes column selections, these (columns selected, their data type and position in the axis) are used to verify what visualizations are available for this selected data. This way we can minimize the errors of the user choosing a map visualization type when no geographical data is selected.

When the user has finished selecting the columns and has chosen the visualization, this information is used to instantly create a preview of his visualization. Also, next to his visualization a list of configurable options (colors, scale, canvas size, etc.) appears with they're default values selected. After changing any of these options values the preview is instantly refreshed. At any time during this process the user can change the selected columns or choose a different visualization. An example of a visualization being created is shown on Figure 3. When the users are satisfied with their visualization they can embed this visualization into their website by copying a piece of code into any webpage, much like embedding a YouTube video.

HEURISTIC EVALUATION

In order to test our design and interaction decisions we conducted a heuristic evaluation of MetaBrain, using Jakob Nielsen's usability heuristics[i]. This evaluation allowed us to find most major usability problems the interface might have.

After a quick introduction to the purpose of our work, four usability experts proceeded to freely test the prototype for a few minutes and then received a list of four tasks to execute. In two the users were asked to extract information from the web, from given domains, and in the other two to craft specific visualizations for that information. All were successfully completed by all users. Overall, only ten usability problems of relevant severity were identified. Most were related to the data extraction interface, especially to the fact of some search queries were taking some minutes to finish and there was no indication of progress, only a looping loading sign. This problem has been solved by adding to the search interface the number of queries to be performed and how many have already been completed. All evaluation experts enjoyed the clean and minimalistic design and the dynamic way in which they could interact with the system. After completing the tasks, some wanted to keep playing with the system, curious about what other information MetaBrain would be able to extract.

This preliminary evaluation allowed us to find and correct some usability problems. It is indicative that the interface can be effective and easy to use. Further validation of this will be provided by upcoming, more formal, user tests, where we'll take into account the number of errors and time taken to complete the tasks.

CONCLUSION

We have presented an interface that allows us to extract and visualize information from the web in meaningful manners. Unlike previous research we strove to make this task as simple and flexible as possible so that any type of users, from less to more experienced, can create customized solutions that fit their needs. A preliminary evaluation of our prototype, MetaBrain, showed positive results. Further user studies will allow us to better validate our choices.

REFERENCES

1. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., and Etzioni, O. Open information extraction from the web. In *Proc. of the IJCAI 2007*.

2. Banko, M. and Etzioni, O. The Tradeoffs Between Open and Traditional Relation Extraction. In *Proc. of ACL-08: HLT*, 28-36.

3. Bollegala, D., Matsuo, Y., and Ishizuka, M. Measuring semantic similarity between words using web search engines. In *Proc. WWW '07*, ACM Press (2007), 757-766.

4. Bostock, M. and Heer, J. Protovis: A Graphical Toolkit for Visualization. In *Proc. IEEE TVCG*, 15 (2009), IEEE CS (2009), 1121-1128.

5. Cimiano, P. and Staab, S. Learning by googling. *SIGKDD Explor. Newsl.*, 6 (2004), 24-33.

6. Etzioni, O., Cafarella, M., Downey, Doug et al. Web-scale information extraction in knowitall: (preliminary results). In *Proc. WWW '04*, ACM (2004), 100-110.

7. Kramer, A.D. An unobtrusive behavioral model of gross national happiness. In *Proc. CHI '10*, ACM (2010), 287-290.

8. Ku, L., Lee, L., Wu, T., and Chen, H. Major topic detection and its application to opinion summarization. In *Proc. SIGIR '05*, ACM (2005), 627-628.

9. Turney, P.D. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. *Machine Learning: ECML 2001*, Springer Berlin (2001), 491-502.

[i] http://www.useit.com/papers/heuristic/heuristic_list.html