Entity Extraction from the Web with WebKnox

David Urbansky, Marius Feldmann, James A. Thom and Alexander Schill

Abstract This paper describes a system for entity extraction from the web. The system uses three different extraction techniques which are tightly coupled with mechanisms for retrieving entity rich web pages. The main contributions of this paper are a new entity retrieval approach, a comparison of different extraction techniques and a more precise entity extraction algorithm. The presented approach allows to extract domain-independent information from the web, requiring only little human effort.

Key words: Information Extraction, Web Mining, Ontologies

1 Introduction

The amount of freely available data on the web is huge and it increases everyday at a rapid pace. Web sites often host high quality information about products such as CDs or books, travel destinations, people and so forth. No single human can have an overview of all that distributed information and thus can not get the entire picture easily. For example, a user wants to find out about new mobile phones, that is, he or she wants to know which mobile phones are available, what features they have, what other people think about them and who sells them, etc. Today, a user might know a good review site and can go there directly. However, chances are low that

David Urbansky
University of Technology Dresden, e-mail: david.urbansky@tu-dresden.de

Marius Feldmann
University of Technology Dresden, e-mail: marius.feldmann@tu-dresden.de

James A. Thom
RMIT, e-mail: james.thom@rmit.edu.au

Alexander Schill
University of Technology Dresden, e-mail: alexander.schill@tu-dresden.de

this site offers all the information the user is looking for so he or she needs to use a search engine and gather the desired information in a time consuming manner.

In this paper, we present the system "WebKnox" (Web Knowledge eXtraction). Our main contributions are a new entity retrieval and extraction approach (Focused Crawl Extraction), a more precise algorithm for entity extraction using seeds (XPath Wrapper Inductor), and that we require minimal user input in form of an ontology.

The remainder of this paper is structured as follows. Firstly, we give background information of tasks and techniques for information extraction from the web. Secondly, we give an overview of three state-of-the-art information extraction systems. Thirdly, we introduce the architecture of WebKnox and elaborate on the retrieval and extraction techniques that are unique to WebKnox. Before this paper concludes with a summary and outlook, we, fourthly, evaluate the main components from the architecture section.

2 Background and Related Work

In contrast to *information retrieval* (IR), in which the task is to find relevant information for a given query and to rank the results, *information extraction* (IE) is the process of extracting information to a given target structure such as a template or an ontology. The IE tasks are defined by an input (such as an HTML page) and an output (such as a populated database) [2]. There are several main tasks in web information extraction such as *Entity Extraction*, which is the task of discovering new instances of a concept, *Fact Extraction*, which is the task of finding values for given attributes of a given entity, and *Concept/Attribute Extraction* which is the task of discovering new concepts and attributes.

Information can be extracted using a variety of different techniques such as natural language processing, pattern learning [3], wrapper induction [2], visual-based [5], and semantic web based [6] approaches. Very common techniques are pattern learning and wrapper induction, in which free or semi-structured text is analyzed for repeating patterns which can then be generalized and used to find and extract new instances of entities or facts.

There is a great variety of systems that extract information from the web, but many work only in certain domains or require considerable user involvement. In this section, we review three significant systems for domain-independent entity extraction from the web.

KnowItAll [4] is a domain-independent, unsupervised system that automatically extracts entities and facts from the web. KnowItAll is redundancy-based which means it relies on the assumption that a fact or entity occurs many times on the web. The system's strength is finding new entities for a given class using pattern learning and wrapper induction. KnowItAll uses a set of domain-independent patterns and queries a search engine with these patterns. The input for KnowItAll is a set of concepts, attributes, and relations. The number of entities that can be found in those relations is however limited, for example very obscure actors are unlikely to

appear in the given patterns. Therefore, KnowItAll has also a list extractor that uses seed entities to find structured lists of entities. KnowItAll queries a search engine with a set of known entities, constructs a wrapper for each returned web page and tries to extract more entities that are in the same format as the seeds.

TextRunner's [1] only input is a corpus of web pages to enable the extraction of information from these pages. This is realized in three steps for every sentence read: The noun phrases of the sentence are tagged, nouns that are not too far away from each other are put into a candidate tuple set and the tuples are analyzed and classified as true or false using a self-supervised classifier [9]. Opposed to KnowItAll, TextRunner does not use extractions from lists and is limited to the entities that can be found in free text patterns.

Set Expander for Any Language (SEAL) [8] is a language-independent system that aims to extract entities from the web after having analysed a few *seeds*. It operates in three main stages, at first it fetches web pages from a search engine by querying it with seeds. After that, all prefixes and suffixes around the seeds are constructed and wrappers are built and finally, SEAL uses a graph algorithm to rank and assess the extracted entities.

The discussed state-of-the-art systems can be extended by using a combination of entity extraction from patterns and from structure. The next section describes the design of WebKnox and particularly elevates the novelties in entity extraction, which are the focused crawl entity extraction technique, the XPath Wrapper Inductor for seed extraction, and that we use a semantic foundation to describe the concepts and attributes the system should be searching for.

3 Architecture of WebKnox

WebKnox is divided into two main extraction processes. First of all, the entity extraction process gets a set of predefined concept names as input (for example "Movie", "Mobile Phone" or "Country") and then queries a multi-purpose search engine such as Google to retrieve pages with possible entity occurrences. After the knowledge base contains some entities, the fact extraction process reads those entity names and also gets predefined information about the attributes that are searched for the entity's concept. The process then queries a search engine again, tries to extract facts, assesses the extractions and writes the results back to the knowledge base. The focus of the remainder of this paper is set on the entity extraction process, detailed information about the fact extraction process can be found in [7].

3.1 Entity Extraction Techniques

Figure 1 shows the entity extraction process realized by WebKnox. The system uses the following three techniques to extract entities from the web: *Phrase Extraction*,

Focused Crawl Extraction and *Seed Extraction*. All three techniques have in common that they get the concept names from the knowledge ontology as an input and that they query a general purpose search engine such as Google to retrieve pages that are likely to have entities for the searched concepts. The Phrase Extraction extracts entities with a relatively high precision, but it does not discover many instances. The Focused Crawl Extraction is then used to find more entities by searching for explicit listings of entities. The Seed Extraction can only be used after another technique since it relies on previously extracted entities as seeds. This way, WebKnox eliminates the human involvement to specify seeds for this extraction process.

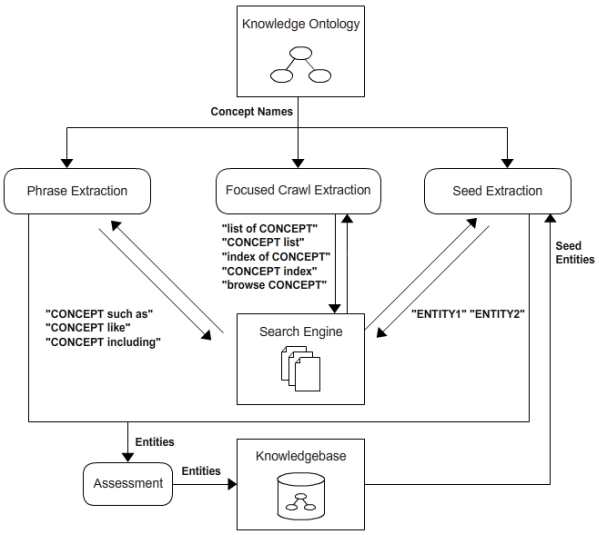

Fig. 1 Overview of the entity extraction process.

3.1.1 Phrase Extraction

The Phrase Extraction (PE) technique borrows the basic idea from the KnowItAll system [4] and queries a search engine with phrases that are likely to link a concept to several entities of that concept.

The following queries are used by the PE technique where *CONCEPT* is the plural name of a concept: "*CONCEPTs* such as", "*CONCEPTs* like", "*CONCEPTs* including". The quotes are part of the query, that is, only exact matches are sought.

For each concept, all queries are instantiated with the concept name and sent to a search engine. The phrases are then searched in the returned pages for each query and proper nouns after the phrase are extracted as entities.

3.1.2 Focused Crawl Extraction

The Focused Crawl Extraction (FCE) technique of WebKnox uses a novel retrieval-extraction combination in entity extraction systems and is explained in more detail. FCE queries a search engine with generic queries that aim to find pages with lists of entities on them (a *list* is an enumeration of entities and does not necessarily mean that the entities are encoded using the HTML `LI` tag). The focused crawler tries to detect a list on the page and, furthermore, it tries to find signs of a pagination. Pagination is often used to limit the results for one page and permits the web site visitor to view the data page by page. If a pagination is detected, the focused crawler starts on the first page it finds, tries to detect a list and then extracts entities from the list.

The focused crawler processes pages that are retrieved with the following queries that have shown to return web pages with entity lists: "list of *CONCEPTs*", "*CONCEPT* list", "index of *CONCEPTs*", "*CONCEPT* index" and "browse *CONCEPTs*". In each query, *CONCEPT* is the name of the concept. These queries explicitly aim to find pages that state to have a list of entities on it. Another helpful synonym is "index" which can be queried equivalently to the list queries. The "browse" keyword aims to find pages that allow a user to browse entities of a concept and thus, possibly, leads to paginated pages with lists.

For each page that is returned by one of the queries, WebKnox tries to detect a list, that is, list detection aims to find the XPath that points to all entities of a list. This is a complicated task since lists can be encoded in a great variety of ways and often it is not even clear whether something should be considered as a list or not. For the list detection algorithm a list must have the following features. (1) The entries of the list are encoded in a very similar way (format and structure), (2) there are at least 10 entries in the list, and (3) the list is in the content area of the web page.

Finding the correct path can rarely be achieved by looking at the page's DOM tree only; hence, also the content of a list is analyzed in order to find the sought list. The list detection algorithm explained here makes use of many heuristics which were determined by analyzing a wide variety of list pages.

Entities are expected to be in one of the following tags: `LI`, `TD`, `H2`,`H3`, `H4`, `H5`, `H6`, `A`, `I`, `DIV`, `STRONG` and `SPAN`. The list detector does not construct an XPath to every tag that can be used in HTML since some tags are not made for having text content such as `TABLE`, `TR`, `SELECT` etc. Every constructed XPath addresses exactly one node, in order to find the XPath that addresses all entities, that is, multiple nodes, the indices of the nodes in the XPath are removed. An XPath with its indices removed is called *stripped XPath*. After the indices are removed, all XPath instances that lead to the same stripped XPath can be counted and sorted by the number of occurrences. It is assumed that finding the XPath with most occurrences on a web page is one that encodes a list, since list entries are most often encoded the same way. The list detector favors longer XPath over shorter ones because it is assumed that the deeper the hierarchy, the more precise the node text will be, so less noise around an entity is extracted.

As explained in the beginning of this section, lists must have certain features in order to be detected and used for entity extraction. Most of the time, lists are used to present something else than entities, for example links to blog entries, numeric values such as prices and so forth. Therefore, heuristics have to be employed in order to find out whether the detected XPath really addresses a list of entities that should be extracted. Through analysis of many entity lists on web pages, we created criteria for a valid entity list. A detected list is only used for extraction if all of the following criteria are fulfilled.

1. Less than 15% of the entries in the list are entirely numeric values.
2. Less than 50% of the entries in the list are completely capitalized.
3. On average, each entity consists of not more than 12 words.
4. Less than 10% of the entries in the tables are duplicates.
5. Less than 10% of the entries in the list are missing.

One of the necessary features for the list detection was that a list has to appear in the content area of the web page. Often the navigation or footer of a web page contains many entries that might look like a list and thus must be filtered out. The list detector tries to find all elements on a web page that are not page specific content by comparing it to a *sibling web page*. A sibling web page is found by analyzing all links from the target web pages, that is, all contents of the `href` attribute of the `A` tag. The link to a URL with the highest similarity to the URL of the target web page is taken as the sibling URL pointing to the sibling page. The similarity between two URLs is calculated by the number of characters from left to right that the two URLs have in common. This way it is likely to retrieve a sibling web page that has a similar structure as the target web page. All stripped XPaths to the content tags are constructed for the target and the sibling page. If the content overlap is over 70% the XPath is considered to point to a non-content area and is dismissed.

The pagination detector recognizes two main types of pagination, namely, uppercase letters and a series of numbers. Since pagination elements are almost always links, the pagination detector constructs all XPaths to `A` tags and adds those to a candidate set. The indices for the elements `A`, `TR`, `TD`, `P`, `SPAN`, `LI` are removed since those elements are more often used to encode pagination lists. The highest ranked XPath is then taken as the XPath addressing pagination links.

3.1.3 Seed Extraction

The Seed Extraction (SE) technique aims to implicitly find pages with lists of entities by querying the search engine with seeds. The retrieval using seeds has excessively been done by the KnowItAll system [4]. WebKnox uses automatically obtained entities (extracted by PE or FCE) as seeds for the seed extraction technique, thus, no human intervention is necessary.

The XPath Wrapper Inductor (XWI) aims to find the XPaths that point to the seeds and to generalize it so that all entities that are encoded in the same way as the

seeds are addressed and can be extracted. XWI needs at least two seeds in order to find such a *generalized XPath*. It then works as follows.

For each seed, all XPaths that point to the seed occurrences on a web page are constructed. After that, a generalized XPath is searched by comparing each index for each element of the XPath. If the index changes, the index is deleted and thus the number of elements the XPath is addressing is increased. Figure 2 shows that process for a simple example and two seeds. In a) and b) the XPath to `Seed1` and `Seed2` are marked with green rectangles in the DOM tree respectively. Both XPaths are the same but only the last index is different, indicating that more nodes with the same structure exist. The index is deleted and in Figure 2c) all siblings of the two seeds are addressed by the generalized XPath. If one or more seeds appear in different elements on the web page, the stripped XPath with the highest count is taken.

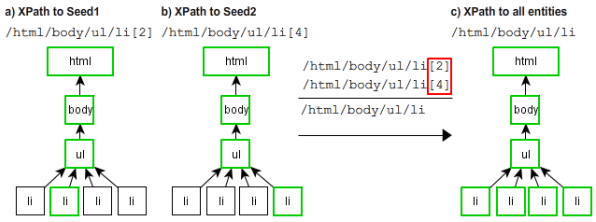

Fig. 2 a) and b) show the XPath for two seeds, in c) the XPath is addresses all elements of the same structure as Seed1 and Seed2.

XWI also makes minimalistic use of prefixes and suffixes around the seeds. This is necessary because the generalized XPath addresses the complete tag content but sometimes there is information around the seed that should not be extracted. To avoid extracting unwanted parts before and after the seed, a two character prefix and suffix is constructed around the seed instances. The complete wrapper for this web page now consists of the generalized XPath and the small prefix and suffix. Applying this wrapper will extract new entities without any noise around them.

The extraction results of the XWI are also checked for uniformity to ensure that fewer incorrect lists of entities are extracted. XWI aims for high precision and rather extracts nothing from a web page than a few correct entities with low precision.

4 Evaluation

In this section, the entity extraction techniques are evaluated. First of all, the difficult part of list and pagination detection for the FCE technique is assessed, secondly the new XWI is compared to another state-of-the-art Wrapper Inductor and, finally, the entity extraction techniques are compared across all concepts.

WebKnox extracted over 420,000 entities from about 24,000 web pages in the following 10 concepts from the web: `Movie`, `Actor`, `Mobile Phone`, `Notebook`, `Car`, `City`, `Country`, `Song`, `Musician` and `Sport`.

List and Pagination Detection

The list detector has been tested on 60 web pages across 10 concepts. The web pages have been found by querying Google with the concept name and the term "list" (and variations). Pages with actual lists and without lists have been chosen for evaluation. The correct XPath was assigned for about 55% of the pages. The list detector aims however to reject a page if the list is not likely to be a list of entities so that the actual accuracy of accepted list pages is about 77%, that is, from 23% of the pages that are analyzed with the list detector, wrong lists are detected.

The pagination detection has been tested on over 70 web pages in 10 concepts. Web pages with pagination and without pagination were chosen for evaluation. The pagination XPath was assigned with an accuracy of about 90%.

Wrapper Inductor Comparison

We compare WebKnox's XPath Wrapper Inductor (XWI) to the entity extractor from the SEAL system [8], since it is a state-of-the-art algorithm that also outperformed the Google Sets algorithm. We implemented the SEAL algorithm as described in [8] and call it Affix Wrapper Inductor (AWI) for simplicity.

For the evaluation 40 list pages across 20 different concepts were collected by querying the Google search engine with the terms "list of *CONCEPTs*" or "*CONCEPTs* list". The first pages that really had a list of the sought concept present, were taken for evaluation.

For each web page, two seeds were chosen randomly and the precision and recall of both wrapper inductors were determined. Precision and recall were then averaged over the 40 test pages. Figure 3 shows the two wrapper inductors in comparison. The two red bars on the top show the precision (pr) and recall (re) for the Affix Wrapper Inductor, whereas the two blue bars in the bottom depict the precision and recall of the XPath Wrapper Inductor. We can see that the AWI has a slightly higher recall than the XWI. The XWI however aims for high precision which is achieved at about 98.94%. The F1-Score for the AWI is about 67% while the XWI reaches about 88%. A paired t-test of the F1-Scores for both wrapper inductors showed that the higher score for the XWI is statistically significant with p being 0.034.

Entity Extraction Techniques

Figure 4 shows a comparison of 12 queries for the three extraction techniques. For the evaluation, a random sample of maximum 100 entities per query has been evalu-

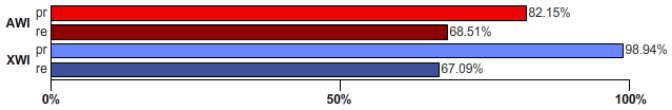

Fig. 3 Precision and recall of the AWI and the XWI.

ated. The red bars in the figure show the precision (left y-axis) of the query averaged over 10 concepts, the blue bars depict the number of extractions, for each query in a logarithmic scale (right y-axis). The green bars next to the red and blue bars visualize the standard deviation. The horizontal red and blue lines show the weighted average precision and average number of extractions per extraction technique.

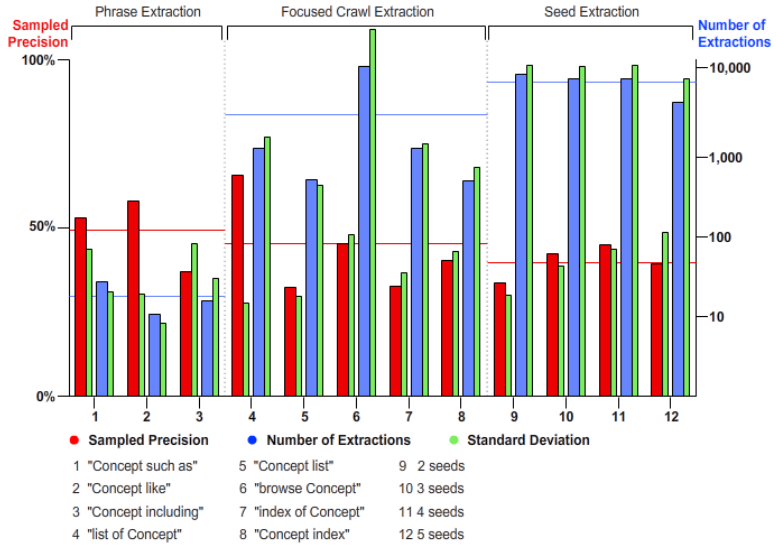

Fig. 4 Comparison of the three entity extraction techniques and their queries.

We can see that the phrase extraction technique (first three queries) has the highest average precision but leads to the fewest extractions. The "list of" query from the focused crawl extraction technique has the highest average precision with about 65% and extracted over 1,300 entities per concept on average. All seeded queries lead to the largest number of extractions, but with the lowest average precision compared to the two other extraction techniques. It is important to note that all queries were very concept dependent, which means that for some concepts they worked exceptionally well whereas they did not reach a high precision for others. The high standard deviation for precision and number of extraction values shows this dependency.

5 Conclusion and Future Work

In this paper we have shown the architectural overview of the web information extraction system WebKnox. The system improves the seed extraction by using an XPath Wrapper Inductor that has shown to be more precise than a wrapper inductor that only uses affixes. We compared different extraction and retrieval techniques and showed their strengths and weaknesses.

In future work we will research different information sources and their impact on entity and fact extraction quality. For example, sources such as REST web services and RDF markup provide information in a more structured form and thus simplify the extraction. We will find different sources and investigate how to generically extract information from them while avoiding human intervention. Furthermore, we will study how to assess the extracted entities to ensure a high precision of the information in the knowledge base.

References

1. Michele Banko, Micheal J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670–2676, 2007.
2. Chia-Hui Chang, Mohammed Kayed, Mohed R. Girgis, and Khaled F. Shaalan. A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, 2006.
3. Doug Downey, Oren Etzioni, Stephen Soderland, and Daniel S. Weld. Learning Text Patterns for Web Information Extraction and Assessment. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining*, 2004.
4. Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Steven Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.
5. Wolfgang Gatterbauer, Paul Bohunsky, Marcus Herzog, Bernhard Krüpl, and Bernhard Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th international conference on World Wide Web*, pages 71–80. ACM, 2007.
6. Borislav Popov, Atanas Kiryakov, Dimitar Manov, Angel Kirilov, Damyan Ognyanoff, and Miroslav Goranov. Towards semantic web information extraction. In *Workshop on Human Language Technology for the Semantic Web and Web Services*, 2003.
7. David Urbansky, James A. Thom, and Marius Feldmann. WebKnox: Web Knowledge Extraction. In *Proceedings of the Thirteenth Australasian Document Computing Symposium*, pages 27–34, 2008.
8. Richard C. Wang and William W. Cohen. Language-Independent Set Expansion of Named Entities Using the Web. In *The 2007 IEEE International Conference on Data Mining*, pages 342–350, 2007.
9. Alexander Yates. *Information Extraction from the Web: Techniques and Applications*. PhD thesis, University of Washington, Computer Science and Engineering, 2007.