

WebKnox: Web Knowledge Extraction

David Urbansky

School of Computer Science and IT
RMIT University
Victoria 3001 Australia
davidurbansky@googlemail.com

James A. Thom

School of Computer Science and IT
RMIT University
Victoria 3001 Australia
james.thom@rmit.edu.au

Marius Feldmann

Department of Computer Science
University of Technology Dresden
Germany

feldmann@rn.inf.tu-dresden.de

Abstract *The paper describes and evaluates a system for extracting knowledge from the web that uses a domain independent fact extraction approach and a self supervised learning algorithm. Using a trust algorithm, the precision of the system is improved to over 70% compared with a baseline of 52%.*

Keywords Information Extraction, Web Mining

1 Introduction

Given the vast quantity of repeated information available on the web, it has become possible to more reliably extract factual knowledge about many different entities. Therefore it is useful to have a automatic approach that finds pages containing facts and extracts the best answers. This paper describes and evaluates a system WebKnox (Web Knowledge eXtraction) for extracting knowledge from the web.

WebKnox's input consists of the following parts:

1. The **concepts**, e.g. Car or Country.
2. The **attributes** for each concept. The attributes determine which facts are searched for each entity in the concept. E.g. for the Country concept attributes could be *population* and *capital*.
3. The **entities** for each concept. The entities and attributes together build the templates that are filled in the fact extraction process. E.g. for the Country concept, *Australia* and *Germany* are valid entities.

The following are the main contributions in this paper. We present a domain independent fact extraction approach that retrieves web pages with factual information, analyzes those semi-structured pages,

Proceedings of the 13th Australasian Document Computing Symposium, Hobart, Australia, 8 December 2008.
Copyright for this article remains with the authors.

and extracts facts from generic structures and formats that commonly occur on websites. We introduce a self supervised learning algorithm that automatically estimates the precision of the different structures used to extract the facts. We show how the extraction precision for numeric values can be increased by cross validating them with numeric values from other entities of the same concept. We demonstrate how a trust value can be assigned to the extracted facts which is used to rank the extractions and help the end user determine which facts can be trusted.

2 Background

2.1 Web Information Extraction

This section gives background information about information extraction in general and the extraction tasks for the web in particular.

In contrast to information retrieval (IR), where 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 (e.g. an HTML page) and an output (e.g. a populated database) [2].

There are several main tasks in information extraction.

1. *Named Entity Recognition* (NER) is the task that identifies entities in a given text. It is the easiest task, however it is more recognition than extraction because no new entities are extracted.
2. *Coreference Resolution* (CO) identifies identity relationships between entities in texts.
3. *Entity Extraction* (EE) is the task of discovering new instances of a concept.
4. *Fact Value Extraction* (FVE) is the task of finding values for given attributes for a given entity. e.g.

the entity “Australia” and the attribute “population” are given and the value for the attribute is searched.

5. *Fact Extraction* (FE) is a similar task to FVE but no attributes are given for the extraction process.

2.1.1 Web Information Sources

The choice of the information extraction technique depends on the format of the source. The world wide web consists of documents that belong to one of the three main types of sources: unstructured, semi-structured and structured.

In web information extraction, semi-structured sources are mainly HTML files. That is because they contain lots of unstructured data as texts but use tags to structure that data for rendering purposes.

Internal Representation Before examining the main techniques that access and extract from these web sources it is important to understand that the content can be represented in two main forms.

1. A *hierarchy of nodes* represents the source as a tree of nodes (such as element or text nodes). This representation is usually instantiated using the *Document Object Model* (DOM).
2. A *tokenized string* represents the source as a parsed string of words, numbers etc. (so called tokens). This representation is often used for free text as there is no other structure that can be used to represent the information.

2.1.2 Information Extraction Techniques

Natural Language Processing (NLP) can be employed for web information extraction using a set of techniques that make the natural language more machine readable. Those techniques are for example *tokenization*, *sentence splitting*, *orthomatching* (coreference resolution) and *regular expressions*.

Wrapper Induction “A program that makes an existing website look like a database is called a wrapper.” [3]. Wrappers perform pattern matching to find the information of interest. The main goal in learning a wrapper is to find a general description of what the information that is supposed to be extracted looks like. Writing a wrapper by hand is labor intense and over the time more automation has been introduced. Chang et al. [2] classify these wrapper techniques in four categories with increasing automation:

1. *manually-constructed wrappers*. The user writes a wrapper for every web site he wants to extract information from which means that he has to have a profound understanding of programming languages. That requirement makes it however impractical for a broad domain approach.

2. *supervised wrapper construction*. For supervised learning a wrapper, a human labels information on a set of HTML pages he wants to have extracted. These labels are taken as positive examples for the learning algorithm. Non-labeled data serves as negative examples. The user does not need any programming knowledge but needs only to be able to mark up the content or use a graphical user interface to do that.
3. *semi-supervised wrapper construction*. Semi-supervised systems require the user not to give a whole exact labeled set of web pages but rather guess extraction patterns on given examples. The user then has to decide which pattern is the correct one, thus the extraction process becomes supervised.
4. *unsupervised wrapper construction*. The unsupervised approach requires no user interaction. While former approaches needed a user to specify the data of interest, the extraction target in supervised extraction are data rich regions of the website [2].

Wrapper induction techniques are primarily used for structured or semi-structured sources as many techniques rely on the DOM tree.

2.1.3 Correctness of Extracted Information

Traditional IE focuses on extracting as much information as possible from a small corpus whereas web information extraction systems often rely on the redundancy of web content [6]. That means that the focus of the extraction techniques should be set on the precision as the recall automatically comes with many mentions of the entity or fact that is extracted. One major problem with web information extraction systems is that the quality of the extractions can vary, i.e. extracted entities may not really belong to the concept they were assigned to or facts are wrong.

Simple Scoring For the fact extraction task a simple scoring, based on the number and quality of sources can be used to decide which fact extraction is correct and which is not [7]. The effectiveness of simple scoring relies however on the assumption that correct facts are extracted more often than incorrect facts which also depends on the extraction technique and the type of the fact. Rare facts for example might be extracted correctly but do not score very high.

Pointwise Mutual Information Etzioni et al. [4] use patterns as discriminators to ensure the correctness of an extracted fact or entity. That means they use these discriminators as queries for web search engines and calculate pointwise mutual information (PMI) between the extraction and the discriminator with the hit counts. If E is an extracted entity and D is the discriminator phrase, the PMI can be calculated as in the Equation 1.

$$PMI = \frac{Hits(E + D)}{Hits(E)} \quad (1)$$

2.2 State-of-the-art Systems

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 (EE task). To do that, it uses a set of domain independent patterns and queries a search engine with that pattern.

The input for KnowItAll is a set of concepts, attributes and relations. The extractor module queries search engines with extraction patterns and performs a shallow syntactic analysis. A discriminator is an extraction pattern with alternative text. The assessor module queries search engines with discriminators to validate a particular extraction and ensure the precision of the system. For that purpose, KnowItAll uses PMI.

KnowItAll is specialized in extracting entities and has its limitations in extracting facts. It can extract entity relations found in free text but much information, especially numbers (e.g. the population of a country) is given in table structures that are not evaluated by KnowItAll. Also the PMI score for validating the extractions would most likely not work with numeric extractions.

Texrunner [1] goes one step further beyond the capabilities of KnowItAll, as it does not require any user input which is more scalable and easier to apply for new domains. Its only input is the corpus of web pages, and information is extracted in a single pass. This happens in three steps for every sentence read: (1) The noun phrases of the sentence are tagged, (2) nouns that are not too far away from each other are put into a candidate tuple set and (3) the tuples are analyzed and classified as true or false.

GRAZER [7] is a system that corroborates and learns new facts. The input for GRAZER are seed facts (attribute-value pairs) for given entities. Entities and seed facts are automatically generated using specialized wrappers. For the given entities, relevant pages are obtained. Relevant pages are those that have a mention of the entity. On these pages the seed facts are corroborated and new facts are extracted. The system searches for mentions of the seed facts on the relevant pages and adds the source if the fact was found on the page. The corroboration happens in free text and in structured HTML as all tags are removed and only the area around the attribute name is searched for the mention of the value.

Although GRAZER does not need an ontology about the knowledge domain as an input it relies on a set of seeds for entities and facts. These seeds are obtained in a non generic way by inputting the data by hand, which is labor intense or by scraping sources with specialized wrappers. The same facts are extracted several times and are treated as new facts when they

have a different attribute which is just a synonym, e.g. “Birthday:17.01.1962” is another fact than “Date of Birth:17.01.1962”.

3 Design

This section introduces the design of our system for fact extraction from the web. First, the knowledge to be extracted is encoded in an ontology, then entities are given, and then the fact extraction process automatically finds the values for the specified attributes.

3.1 Knowledge Representation

Before the extraction process can start WebKnox needs to know what concepts, attributes and entities exist. This knowledge is called *prior knowledge*. The prior knowledge for WebKnox is modeled in an ontology using OWL. Therefore, all concepts and attributes are defined in the *knowledge ontology* and the entities and facts that are extracted are stored in another separate *data ontology*.

The purpose for the knowledge ontology is (1) to define the knowledge represented in the data ontology and (2) to serve as an input for the extraction process. In the knowledge ontology every attribute gets an *OWL data type property* assigned to it. This determines which type of value the attribute will have and can be used for (1) other programs reading the ontology, trying to parse the data and (2) for the extraction process to know which values on a source are candidates for the attribute. A datatype property can have any XSD datatype¹ but WebKnox only uses the following:

1. **String:** A string is a sequence of characters, WebKnox will however only consider proper nouns as fact candidates for a string attribute, i.e. only a sequence of words starting with a capitalized character or a number are considered to be possible answers.
2. **Boolean:** Attributes with a boolean value can either have true or false as a value. WebKnox searches boolean values only in tables and looks for “yes” and “no” occurrences.
3. **Decimal, Double, Float, Integer, Int, Long:** These are numeric attributes which are all handled equally by WebKnox. Every numeric attribute is handled as a double and only occurrences of numbers are extracted as fact candidates for the attribute.
4. **Date:** An XSD date is a string given in a standardized UTC format: YYYY-MM-DD. WebKnox will look for several representation of dates on web sources and tries to transform these back to the UTC format.

¹<http://www.w3.org/TR/xmlschema-2/#d0e11239>

5. **AnyType**: Attributes with values that do not match any previously mentioned data type can have the AnyType property. WebKnox takes all characters around or after the attribute on the web source into account when determining the fact candidates. Thus AnyType can be used for strings that are not proper nouns.

3.2 Fact Extraction

The first process is the retrieving of the source pages which gets an entity and its attributes as input. The extraction process then extracts the values for the entity's attributes from the websites retrieved. The extracted facts are normalized and eventually the trust in the extractions is calculated.

3.2.1 Retrieving Fact Pages

Retrieving relevant pages that host the searched facts is a crucial process that has to be tightly coupled with the extraction process. As input data the source retrieval process gets the names of the entities and attributes that are being searched for. The process then queries a search engine and outputs the retrieved pages together with information about which attributes are expected on the page. This output is fed into the extraction process. The focus lies on retrieving semi-structured HTML pages as they are easy to access via generic search engines as Google².

To retrieve pages that have the searched facts present, WebKnox uses two kinds of generic queries. The first kind is called *multi-attribute query*, it tries to find pages relevant to the entity and extract all searched facts from the retrieved pages (e.g. the query "Australia"). The second kind is the *single-attribute query* and is focused on each single attribute, i.e. it queries the search engine with attribute specific terms (e.g. the query "Australia population").

The retrieved websites are then passed to the fact extraction process. The fact extraction process also gets information about the type of the query so that it only looks for a single attribute on single attribute pages and tries to find all attributes on general fact pages retrieved by multi-attribute queries.

3.2.2 Exploiting Structure and Format of Web Pages

The quality of extracted facts can be increased by using different extraction structures for different types of fact appearances. As covered in the background a common approach for fact extraction is to use the complete website content and simply remove all HTML tags (as done by the GRAZER system [7]). That however also removes all advantages that come with the semi-structured type of HTML documents. WebKnox differs from current approaches as it takes the *extraction structures*, i.e. the different generic formats and structures

into account that are used to represent facts on web pages.

Definition 1 (Extraction Structure). An **extraction structure** is the pattern or format the extracted fact is represented on a source.

These extraction structures are *phrases*, *tables*, *colon patterns* and *free text*.

Phrases are natural language representations of facts for a specific entity. For example the phrase "The capital of Australia is Canberra" is used on a website. The phrase covers the fact `capital:Canberra` for the entity `Australia`. Ideally the searched value for the attribute appears right after the "is" in the phrase. WebKnox uses only two phrases: the `ATTRIBUTE of ENTITY is` and `ENTITY's ATTRIBUTE is`. These phrases are also used by the source retrieval process to discover pages that state these phrases.

Tables are important HTML structures on the web that are used to represent many facts, which led to numerous wrapping techniques. Keeping the HTML structure allows to traverse in the DOM Tree of the website and find corresponding attribute-value pairs in tables. Figure 1 shows an example³ of a rendered HTML table in a) and the DOM representation of that part in b). That is a very easy example of a table but also a very common one. By identifying the `td`-element with the attribute, the sibling `td`-element with the value can be found and only the text inside that element is extracted.

a) HTML rendered table

Dimensions	99 x 53 x 21 mm, 90 cc
Weight	120 g
Type	TFT, 16M colors
Size	240 x 320 pixels, 2.6 inches, 40 x 53 mm

b) DOM representation of the table

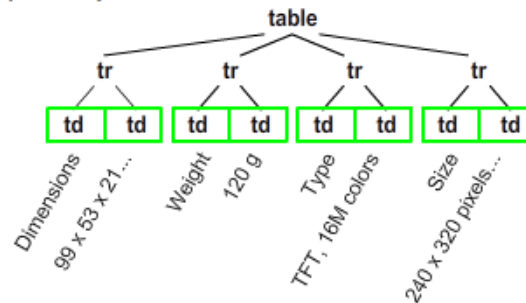


Figure 1: A table for mobile phone specifications

Colon pattern is the text that is right after a colon ("."). Often facts are given in an unstructured way (no tags) but with the format `ATTRIBUTE:VALUE` so that only the text after the colon needs to be extracted. Figure 2 shows an example⁴ of this representation, where

³Table from http://gsmarena.com/nokia_n95-1716.php

⁴Data from <http://engadget.com/2008/08/30/msis-wind-u90-to-boast-8-9-inch-display/>

²<http://www.google.com>

a) depicts the HTML rendered version while b) shows the text as it is seen when the separating tags are removed (replaced with whitespace). If one would try to extract the processor attribute (PROC), expecting a numeric value and not noticing the format, the 2008 would be extracted as it is closer to the processor attribute than the correct value 1.6GHz after the colon. The colon pattern can therefore help increasing the fact extracting precision in a very simple manner.

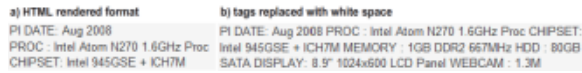


Figure 2: An example for fact representation in a colon pattern, a) shows the presentation in rendered HTML, whereas b) shows the data when tags are removed (replaced with white space)

Free text is the absence of structure (tags) and additional format (phrase or colon pattern). Facts can also appear in long paragraphs of text but as no further information about the structure and format is given, all text around the attribute has to be considered as a valid answer for the attribute's value. It is assumed that always the next matching value closest to the attribute is extracted. WebKnox takes the sentence in which the attribute appears as the boundary. This way incorrect information further away is not extracted as well. Using information found in free text increases the recall and must be considered, especially for rare facts that do not appear in tables or other structures and formats.

Some extraction structures are more reliable than others. It is also necessary to take all possible extraction types as it increases the recall and some facts can only be found looking in a certain structure. The trust in the fact values extracted by a structure must therefore take the employed extraction structure into account. The next section describes how the trust in extracted facts is calculated.

3.2.3 Calculating the Trust in Extractions

Once values for attributes have been extracted, they need to be ranked in order to determine the value that is most likely to be the correct one for the attribute. It is now necessary to find the correct ones by assigning *trust* to each extraction.

Definition 2 (Trust). The **trust** is a non negative number. The higher the number the more reliable the extracted value.

The following equations assign trust values and aim to improve the ranking of the extracted values, i.e. to put the correct ones on top.

The easiest way to rank the extracted values it by just counting the number of extractions. The more often a value has been extracted, the higher the trust value. Equation 2 shows how the trust value is calculated in that case, with N being the number of extractions for the given value, and x being a tuple

consisting of *concept*, *entity*, *attribute* and *value*, $x = \langle x_{concept}, x_{entity}, x_{attribute}, x_{value} \rangle$. This way of assigning a trust value is called "Quantity Trust" from now on.

$$\text{QuantityTrust}(x) = N \quad (2)$$

The Quantity Trust does not make use of additional information like **where** (the source) and **how** (extraction technique/structure) the fact was extracted. This information must be considered when determining the trust for an extraction.

Determining the Source Trust Some pages that are retrieved for the extraction process mention the attribute and its value several times. For example, suppose a page that is retrieved, when searching for the entity Nokia N95 and the attribute talk time, mentions the attribute several times, two times with the correct value of 6.5 hours but three times with different values that do not relate to the entity but to other mobile phones. The source trust can therefore be reduced whenever there is more than one value for the searched attribute as shown in Equation 3, where D is the number of different values found for the given attribute and source. The source trust can have values between 0 and 1 with one being highest trust and zero being no trust.

$$\text{SourceApplicability}(attribute, source) = \frac{1}{D} \quad (3)$$

Determining the Extraction Structure Trust

Extraction structures have different precisions that must be taken into consideration when calculating the trust for a fact value. The values determined in the test set are not representative for all possible concepts and domains. Since WebKnox aims to be domain independent, the precisions determined for the test set cannot be taken as references. WebKnox uses self supervised machine learning to automatically estimate the trust for the four extraction structures used. The trust value for the extraction structures is an estimated precision, i.e. it is a number between 0 and 1 with one being highest trust (all extractions were correct) and zero being no trust (all extractions were incorrect).

For all extraction structures e , information about the number of extractions $N(e)$, and the number of correct extractions $C(e)$ is kept. The ExtractionStructureTrust is then calculated as the ratio of correct extractions to total extractions (Equation 4):

$$\text{ExtractionStructureTrust}(e) = \frac{C(e)}{N(e)} \quad (4)$$

Initially all extraction structures are initialized with a trust value of 0.5. The three steps are then as follows:

1. The input for the first step is the extraction result with an assigned trust. In the first step the highest trusted fact is searched throughout all concepts

and attributes. It is then assumed that this fact is really a correct one, since it has a high trust. All extraction structures used to extract that very fact value get credit for a correct extraction, i.e. , $C'(e) = C(e) + 1$ and $N'(e) = N(e) + 1$. Extraction structures that led to wrong fact values for that attribute, get credit for a wrong extraction, i.e. $N'(e) = N(e) + 1$. In the next iteration that highly trusted fact is not considered anymore when looking for the highest trust.

2. In the second step, the trust for the extraction structures is updated based on the number of correct and total extractions that have been revised in the former step, i.e. the extraction structure trust is recalculated using Equation 4.
3. In the third step, the trust for all extracted values is recalculated by using the updated trust for the extraction structures. After this step, the ranking of the extracted values for each attribute might change. The newly ranked list is then again input for the first step to repeat the process. The iteration can be stopped when the trust for the different extraction structures converges. In case the trust does never converge, the iteration will only stop after all highest trusted facts have been evaluated in step one.

Combining Source and Extraction Structure Trust

Taking both, the source trust and the trust in the extraction structure, into consideration, the trust for an extracted value can be calculated as shown in Equation 5. S is the set of sources the given fact has been extracted from, $\text{ExtractionStructureTrust}(e)$ is the trust of the extraction structure e used and $\text{SourceApplicability}(s)$ is the trust for the source s . The trust will therefore be high, when the value has been extracted in many trustworthy sources using numerous highly trusted extraction structures. This trust formula shall be called ‘‘Combined Trust’’.

$$\text{CombinedTrust}(x) = \sum_{s \in S} \left(\sum_{e \in E} \text{ExtractionStructureTrust}(e) * \text{SourceApplicability}(x_{\text{attribute}}, s) \right) \quad (5)$$

Normalization Facts can be represented in different formats which still represent the same thing. For example dates can be written in many ways, such as January, 17th 1962 or 17/01/1962. Also many numeric facts have units. Not taking the unit into account leads to the extraction of two different facts where actually only one is mentioned, e.g. 2 inch and 5.08 cm is the same fact. Normalization helps to find facts from different formats and to cluster them.

Validating Numeric Fact Values across Entities Another problem with extracted facts is that some

attributes do not have a single absolutely correct value. The population attribute for example is not mentioned correctly on any website on the entire web as it changes almost every second. Instead there are values that are *almost* the same and can be considered correct. Fact values for attributes with fuzzy values tend to not corroborate well. For example, the following fact values might have been extracted for the population attribute for Australia:

```
300 (3 times)
21000000 (1 time)
21340000 (1 time)
22578420 (1 time)
20452340 (1 time)
```

The problem here is that the exact same number for the population is not mentioned on more than one source. The incorrect extraction 300 however is extracted several times and therefore gains higher trust.

To solve that problem, two assumptions are made:

1. The order of magnitude (OOM) for numeric facts is often the same for entities within the same concept; there are exceptions such as the population of countries.
2. There are well-known entities where the information about the numeric attribute can be extracted with relatively high trust because they appear on very many pages.

Both assumptions were supported in our test set for most of the fact values. A bigger test set with more entities (and more well-known ones) would most likely further support that assumption.

To make advantage of the fact that the OOM is often the same, WebKnox uses a validation process across all entities for a given attribute. This process is called ‘‘Cross Validation’’ and is part of the second step in the self supervised learning loop. It works as follows:

1. For all numeric attributes, an OOM distribution is constructed.
2. If the highest trusted value from the first step of the learning loop is a numeric value, the number is considered to be correct and the OOM of that number is given credit in the attribute’s OOM distribution.
3. In the next iteration the trust for the fact values for the same attribute will be calculated as shown in Equation 7. The *CrossValidationFactor* for a numeric fact value is one plus the *support* of the OOM, which is a number between 0 and 1 with 1 being 100% support (all other entities of the concept had values with exactly the same OOM for that attribute) and zero being 0% support (no other entity of the same concept had the same OOM for that attribute).

$$\text{CrossValidationFactor}(x) = 1 + \text{support}(\lfloor \log_{10}(x_{value}) \rfloor, x_{concept}) \quad (6)$$

$$\text{CrossValidationTrust}(x) = \text{CombinedTrust}(x) * \text{CrossValidationFactor}(x) \quad (7)$$

4 Evaluation

The Test Set contains six different concepts (five entities each) and five data types. In total there are 255 facts to extract.

The entities for each concept were chosen manually by applying following criteria to gain a more representative sample for each concept: Notebooks, Mobile Phones and Cars were chosen from different manufacturers; small and large Countries were chosen; Movies were chosen based on popularity; and only well known Actors were chosen.

For the evaluation, 2420 HTML pages were retrieved using the REST web service from Yahoo!⁵. Each entity was searched for using two multi-attribute queries and each attribute of an entity resulted in three single-attribute queries. For each query, only the top eight retrieved URLs were used for the fact extraction process. Not all queries led to eight answers from the search engine, in that case all answers were taken.

The standard measures for comparing extraction systems are *precision* and *recall*. In web information extraction, the precision measures the ratio of correctly extracted facts or entities to the total extractions, and the recall measure determines the ratio of the performed extractions and the extractions expected. Additionally, the measure *found* is used several times. *Found* is the ratio of extracted facts (correct or not) to expected facts. A found value of one means, that for every attribute at least one value has been extracted. If not otherwise stated, the measures always relate to the complete test set of the WebKnox system.

Baseline Basically two different approaches are used today: (1) wrapper induction and extracting from tables and (2) treating the website as a long (tokenized) string by removing the tags. As the developed approach extracts information not only from tables it is appropriate to compare it to the latter technique.

The baseline extraction works similar to the technique from the GRAZER system [7]. All tags are removed from the website, all occurrences of the attribute are evaluated and the corresponding fact values are expected before or after the attribute. Only the first 150 characters before and after the attribute are searched for the matching value to delimit noise. The trust is calculated only by counting the number of extractions (Quantity Trust).

Evaluation of Source Retrieval WebKnox uses a set of generic queries to retrieve websites from a search engine that are likely to have a mention of the facts searched for. When retrieving the top eight results, 95% of the facts were found. All further evaluations for the fact extraction rely on the test set that was gained by taking the top eight results from Yahoo! for all queries.

Evaluation of Extraction Structure Trust Learning

Figure 3 shows the learned trust values for the four extraction structures after every iteration of the learning loop. The dashed lines visualize the manually determined precision values in the test set for each extraction structure and the solid lines are the automatically calculated trust values from the learning loop. All trust values were initialized with 0.5. The graphic shows that the *free text* (red) and *table* (green) extraction structure do not change very much after the first forty iterations. The *phrase* and the *colon pattern* extraction structure however, seem to drop and raise quickly even after 40 iterations. This behavior is due to the occurrences of these structures. The “found” value for these two structures was lower than for *free text* and *table*, which means that the extraction structure occurs more rarely and therefore it takes longer to gain a stable trust value. The loop did not stop before all iterations have been performed since the trust values did not converge so far. 172 iterations was the maximum for the test set with 255 facts since not all facts have been found. After 172 iterations, three of the four extraction structures got an automatically assigned trust value that is in a 4% margin to the “correct” precision value for the extraction structure in the test set. The highest discrepancy can be seen with *phrase* that is 5.2% away from the correct trust value. This again can be explained by the low occurrence number, only every fourth fact can be found by the *phrase* extraction structure. The black line in Figure 3 depicts the overall precision of WebKnox. It is shown that the precision does in fact increase as the extraction structures get trust values closer to their real precision. Through the learning loop, an overall precision gain of 7.4% is gained, recall is also affected positively with an increase of about 7%.

Cross Validation When comparing the extraction precision for the numeric data type we find that (both with and without crossvalidation) WebKnox found 143 of the 145 numeric facts in the test set. The learning loop was performed 172 times in both cases, until no more iteration was possible. Without cross validation 63.19% of the extracted numeric values were correct. Using cross validation showed a gain in precision of almost 7% to 70.13%. The difference between the baseline and WebKnox for numeric fact values now increases to over 25% in precision.

Overall Fact Extraction Performance Figure 4 shows the comparison between the baseline and the fact extraction process of WebKnox for all concepts of the test set. The evaluated fact extraction process used

⁵<http://developer.yahoo.com/search/>

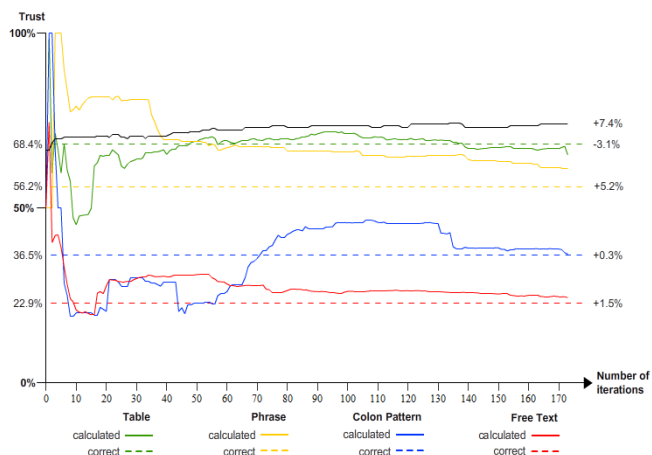


Figure 3: Evaluation of the self supervised learning loop for the extraction structure trust.

Equation 7 (with cross validation) and stopped after 172 iterations to weight the extraction structures and apply cross validation for numeric facts. The measures in the figure are *pr* for precision and *re* for recall. There are two bars for each measure and concept, where the left is the one for the baseline and the darker right one is the measured value for WebKnox. In five of the six concepts, WebKnox reaches a higher precision and recall than the baseline. For the car and notebook concept it does not perform considerably better than the baseline. That is because the normalization step sometimes fails to normalize the numbers correctly. In the car and notebook domain most of the facts are numeric facts and several times there is no unit given with the fact. Overall, the system achieves precision and recall over 70% compared with the baseline of just approximately 52%.

5 Conclusion and Further Work

We showed that we can increase the fact extraction performance by searching facts in different formats and structures of HTML documents. Furthermore, we introduced an algorithm that can learn a trust value for those structures in a self supervised manner. We are able to assign a trust value to the extracted facts based on a source trust and the trust for the extraction structures. Further work needs to be done especially:

1. Determining how well the trust value indicates for the end user the reliability of the automatic extraction.
2. Finding further criteria to calculate the source trust more accurately for the extraction process.
3. Investigating in further domain independent formats and structures that are used to represent facts on websites.
4. Automate identification and extraction of entities (extending the work of Vercoestre et al. [5] on entity ranking from Wikipedia).

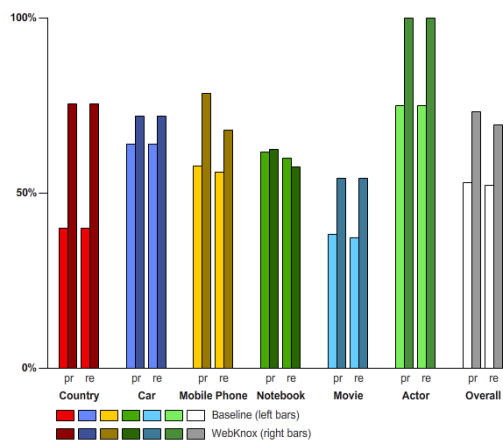


Figure 4: Evaluation of the WebKnox system against the baseline across six concepts.

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*, Volume 18, Number 10, pages 1411–1428, 2006.
- [3] William W. Cohen, Matthew Hurst and Lee S. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *Proceedings of the 11th International Conference on World Wide Web*, pages 232–241. ACM, 2002.
- [4] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 100–110. ACM, 2004.
- [5] Anne-Marie Vercoestre, James A. Thom and Jovan Pehecvski. Entity ranking in Wikipedia. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1101–1106. ACM, 2008.
- [6] Alexander Yates. *Information Extraction from the Web: Techniques and Applications*. Ph.D. thesis, University of Washington, Computer Science and Engineering, 2007.
- [7] Shubin Zhao and Jonathan Betz. Corroborate and Learn Facts from the Web. In *KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 995–1003. ACM, 2007.