Automated Meta-Data Extraction for Confsearch

Semester Thesis

March 16, 2011

Jochen Mattes
(jmattes@ethz.ch)
MSc. Student Electrical Engineering and Information Technology

Advisor: Dr. Michael Kuhn
Supervisor: Prof. Dr. Roger Wattenhofer

Distributed Computing Group
Computer Engineering & Networks Laboratory

Department Information Technology and Electrical Engineering
Swiss Federal Institute of Technology Zurich (ETH Zürich)

Abstract

Extracting meta-data from websites is an open field and up till now there exists no satisfying solution for extracting important dates (e.g. the Paper Submission Deadline) from conference websites. We present an automated way to extract the meta-data of an academic conference from its website. We aim to facilitate the manual update of such data on the conference directory Confsearch (http://www.confsearch.org).

Contents

Chapter 1

Introduction

1.1 General Problem

The World Wide Web is a huge source of semi-structured documents that mostly present information in a way that it is easily accessible to the human user. Yet extracting the information can be a very hard task for a computer. Several applications could greatly benefit from the information stored in these semi-structured documents if they were readable by computers. Question answering services such as Wolframalpha[1] benefit largely from semi-structured documents. For Confsearch as search engine for academic conferences it would be very interesting to have access to meta data about a conference stored in the document. To our best knowledge, other conference directories and conference search engines (cf. Section 2.1) have the same open problem. These services could also greatly benefit form the solution we are evaluating.

1.2 Confsearch

Confsearch is a conference search engine that lists many conferences from various fields related to computer science, information technology and electrical engineering. The unique selling point of Confsearch is its searching and ranking algorithm that works on basis of DBLP[2] publication records [1].

Unfortunately DBLP does not provide meta data for the corresponding conferences and so in Confsearch the user can modify the meta-data of upcoming events and add new conferences to the site. The meta-data stored are:

- URL of the conference website,
- Conference Date (start and end),
- Abstract Registration Deadline,

[1] `http://www.wolframalpha.com`
[2] `http://www.informatik.uni-trier.de/~ley/db/`

- Paper Submission Deadline,
- Date of the Notification of Acceptance and
- Deadline for the Camera-Ready Version of the Paper.

When adding a new conference to the website the user is asked to fill out at least 13 fields. Confsearch expects the dates to have a specific format, which makes copy & paste form the conference website impractical for most cases and adding or modifying a conference consumes a lot of time.

It would be much more convenient if the user only had to give the URL of the conference website and having an automated way to extract all the relevant information from the website.

Chapter 2

Related Work

We take a look at other conference directories and how they keep their data set up to date (Section 2.1), how information extraction is dealt with in general (Section 2.2) and what is special for the case of information extraction from websites (Section 2.3).

2.1 Conference Directory

Confsearch is sharing the market of conference directories / search engines with some other well-maintained websites and a lot of websites with only limited functionality and use; we list the most important ones (according to our research).

Allconferences.com [1] is a website that list conferences of all academic fields. One can add a conference manually through a very extensive form which makes adding a conference a very time consuming endeavor. The information entered by the user is evaluated by the website operator and then released.

WikiCFP [2] is a semantic wiki that lists a number of conferences in science and technology. A registered user can add new conferences and modify existing conferences and they rely on the "wiki-believe", that even though a single person might have bad intentions one can always trust the community. Therefore the data set is maintained by its users.

Conferencealerts.com [3] is a website that informs its user about new conferences and updates in the meta data of a conference. Every visitor of the website can add a new conference to the database, but modification of the meta data is only possible for the person who added the event.

[1] `http://www.allconferences.com`
[2] `http://wikicfp.com`
[3] `http://www.conferencealerts.com`

Other services There are many other sites on the web that are more specialized or managed by a single person.

2.2 Information Extraction

Grisham describes Information Extraction in the following way: "Information Extraction [...] involves the creation of a structured representation [...] of selected information drawn from the text" and states further: "[...] in Information Extraction we delimit in advance [...] the semantic range of the output: the relations we will represent, and the allowable fillers in each slot of extraction" [2]. When extracting meta-data like the submission deadline from the website we do exactly that: we delimit the "semantic range" to a date and extract that information from the website.

Information Extraction is a wide field and the Message Understanding Conferences (MUC-X) concentrate mostly on Information Extraction. Still Information Extraction in its general definition as given by Grisham [2] concentrates on extracting information from plain text and not from websites. The more specific task to extract information from websites is part of the "Web Information Extraction".

2.3 Web Information Extraction

Laender et al. [3] categorize the Web Information Extraction Techniques as follows:

- HTML-Aware Tools, that transform the source code of the website into a parsing tree and operate on that,
- NLP-Based Tools, that apply filtering, part-of-speech-tagging and lexical semantic tagging to build relationships between phrases and sequence elements,
- Wrapper Induction Tools, that generate delimiter-based extraction rules,
- Modeling-Based Tools, try to locate parts of the website that conform with a given structure,
- Ontology-Based Tools, that rely directly on the data and use an ontology to locate constants.

Still the approach we take does not fit in either of these categories, as we rely on the parsing capabilities of the Mozilla Browsing Agent and extract relevant information from the DOM after the website has been parsed.

HTML-Aware Tools HTML-aware tools download the source code of the website and then transform it to a parsing tree. Tools like W4F [4], XWRAP or RoadRunner help users to build wrappers that extract information form existing websites. W4F[4] and XWRAP[5] generate solid website-specific wrappers. So the websites have to be known in advance which is not the case when we want to extract information from an arbitrary conference website.

RoadRunner [6] relies on extracting information from websites that represent objects of a database, as it uses the difference of two similar websites to build a wrapper that extracts the "different" information on the websites. This works well for any kind of website that is directly generated from a database [6]. Yet this approach does not work for our case, as there is no second website that one could use to calculate the differences.

NLP-Based Tools NLP-based tools like RAPIER [7] concentrate on extracting information from text documents and mostly ignore the HTML context. We might loose relevant information when discarding the structure of the website. But the main problem is that the meta data we are searching for is normally contained in tables or table-like structures. Using NLP-based tools does not seem to help here.

Wrapper Induction Tools Wrapper Induction tools like WIEN [8] always generate wrappers that are specifically built for a certain website. Wrappers help to extract information from documents that were generated from a database, which is not the case in our problem.

Modeling-Based Tools Tools following the modeling-based approach like NoDoSE [9] work with a user-defined model of the document. This is mainly done for plain-text documents as they don't provide additional cues that would allow it to use a wrapper.

Ontology-Based Tools Waterson and Preece state: "An onotology defines the terminology of a domain of knowledge: the concepts that constitute the domain, and the relationship between those concepts." [10] Ontology-Based Tools rely on a "previously constructed ontology to describe the data of interest, including relationships, lexical appearance, and context keywords. By parsing this ontology, the tool can automatically produce a database by reorganizing and extracting data present in documents or pages given as input." [3] As this method strongly relies on the careful construction of an ontology [3] it seems to be an overkill.

Chapter 3

Algorithms

3.1 General Procedure

At first we describe how to represent the website (Section 3.2) so that we can easily extract the information we need. Then we describe how to form word clusters (Section 3.3) and how to find and interpret the dates (Section 3.4) and labels (Section 3.5) in the word clusters. A label as depicted in Figure 3.1 is in this context any string that gives information to whatever class the date nearby belongs to. Having found all dates and labels on the website, we describe how to establish a relationship between them (Section 3.6) and how to find the best fitting of the possible pairs (Section 3.7). At last we describe how to check the found results for plausibility (Section 3.8).

3.2 Representing Websites

Out of the many possible representations of a website we mainly considered the following three: source code, image and DOM.

Source Code The main advantage of using the web site's source code is simplicity as the source code is very easy to obtain. However the line distance of two objects in the source code does not necessarily relate to the pixel distance of the two objects after the website has been parsed by a web browser. As the human visitor of the website

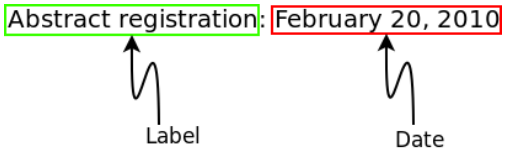

Figure 3.1: Illustration of Label and Date

draws conclusions about the relation of two objects from the pixel distance of these objects, it is a serious drawback to lose that information.

Image The problem with the source code, that we do not know the absolute position of an object after it is parsed, can be overcome when choosing the image representation of the website. This could be obtained by calling the website using a browser, letting the browser parse it, taking a screen shot of the parsed website and afterwards running a OCR tool to obtain the text nodes and their position on the website. The big advantage with this method is that we can access the information stored in the images on the website. A noticeable number of websites do only publish the conference date in this way. Still the effort necessary to implement this procedure seems to be too high.

Document Object Model According to International Standards Organization for the World Wide Web (W3C) the "Document Object Model" (DOM) is "[...] a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents." [11]

This allows to access the text nodes on the website including their absolute position including height and width of the text box as well as further information about the text node. Even though it is therefore possible to directly access all the information that is stored on the website in clear text, one still had to use OCR tools to extract the information stored in the images.

Chosen Solution The DOM representation seems to be the most promising. To make work even simpler we decided to extract only the relevant parts of the DOM tree. The procedure is the following:

1. Download the website to the local file system,
2. load the website in firefox,
3. wait for the browser to finish parsing the website,
4. run a JavaScript script to extract the relevant parts of the DOM Tree
5. and callback the Java application with the extracted data as parameter.

We have to download the website to the local file system as the Cross Scripting Countermeasures of the browser will not allow us to execute the JavaScript function, when a website uses frames with different origin top level domains.[12]

The JavaScript function walks through the DOM Tree top down, splits each text node into "words" (strings that do not contain white spaces) and extracts the information we want to use in further processing.
The extracted information namely is:

- the node id which is an incrementing number that uniquely identifies the text node,
- the actual text in the text node,
- the x and y position of the left top corner of the text node,
- the height and width of the box,
- whether or not the word is struck out,
- and the "test font size" which is a measure for the font size[1].

3.3 Clustering Words

The data the Java application receives form the JavaScript function is not yet very helpful as we lost the context of the individual words when extracting them from the website. Therefore we combine words that are direct neighbors into a word cluster. The methods that follow will either work directly on the words or use the word clusters.

3.4 Finding and Interpreting Dates

After combining the words into clusters we can search for dates and interpret them. This in done in two steps: Marking and Extraction.

Marking The marking step makes extraction easier as it maps multiple representations with the same meaning to an equivalence class as illustrated in Figure 3.2. It will further split the words into strings that do only contain alphanumeric characters. We will then translate possible representation of a day, month or year into a unique representation. So "Jan" and "January" are both translated to "%month(1)%". As the marking algorithm does not have any information of the context, it has no way to figure out whether the input string "1." stands for the first day of the month, or for January. It therefore translates it to "%dayOrMonth(1)%.".

Extraction After Marking extraction method uses the additional information we obtain from the context. The extraction method requires a list of possible date formats (either individual dates or date periods) to search for and extract the obtained results. Because we are using solely context information and are operating on the marked text, we can extract date periods as well as single dates in the same step.

Therefore we used the following date formats listed in Table 3.1. This list is not claiming to be complete, but we are not aware of a single conference website, that cannot be interpreted correctly using this list.

[1] The "test font size" is the area covered by the capital 'A' and seems to be a more appropriate measure for the perceived size of a font than the font size itself.

date format
weekday#month#day##weekday#month#day#year
weekday#weekday#month#day#day#year
day#month#year##day#month#year
month#day##month#day#year
day#month##day#month#year
month#day##day#year
day##day#month#year
day##day#month
month#day##month#day
month#day##day
day#month#year
weekday#month#day#year
month#day#weekday#year
weekday#day#month#year
month#day#year
day#month#year
year#month#day
weekday#month#day
weekday#day#month
month*#day#weekday
month*#day
day#month*

Table 3.1: Table of recognized date formats used in the Extraction step. '#' separates two date elements. '##' separates two dates of a time period and month* is a placeholder for strings that can only be interpreted as a month (e.g. Jan, January, but not 01)

Sometimes the date format does not explicitly state the year (e.g. Jan 4th). For this reasons a preferred year can be specified.

3.5 Finding and Interpreting Labels

Having found the dates on the website we have to find the interesting labels as they occur on the website (e.g. "Submission Deadline") and assign them to the correct class (e.g. Paper Submission Deadline). This is again done in two steps as illustrated in Figure 3.3: in the first step (Scoring) we calculate a score that expresses our "belief" that the label belongs to a certain class and in the second step (Classification) we assign that label to the class with the highest score.

Scoring In the first phase we take an individual word cluster and calculate a score that expresses our "belief" that this word cluster belongs to the specified class. (Even

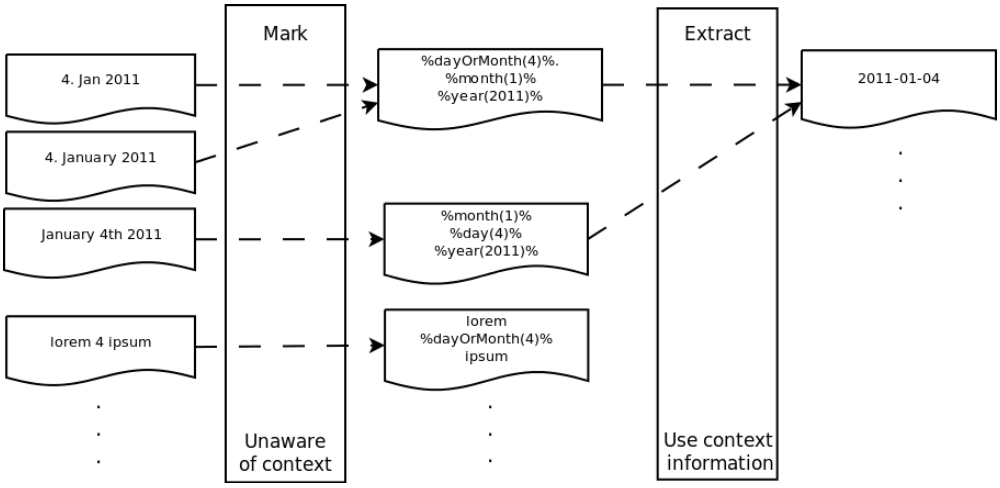

Figure 3.2: Two step procedure to mark and extract the dates.

though using probabilistic vocabulary, this is a deterministic approach.)

The algorithm splits the word cluster (a potential label) into words (splitting at white spaces). For each word it checks the predefined keyword list for the word and adds the associated score to the overall score if the word is listed. If the word is not found in the keyword list, the overall score is decreased by 100 points. The algorithm then returns the positions of the first and last word that was found in the keyword list as well as the calculated overall score.

The following pseudo code describes the principle in more detail.

```
function calculateClassScore(String wordCluster, int classId)
{
  int score = 0
  int firstHit = -1
  int lastHit = -1

  keywordList = get keyword list for class with the correct classId

  for each word in wordCluster
  {
    if word is in keywordList
    {
      if have not found start point yet
      {
        firstHit = current word location
      }
```

```
      curScore = score assigned with keyword
      lastHit = current word location
    }
    else if found start point
    {
      curScore = -100
    }

    score += curScore

    if score < 0
    {
      break
    }
  }
  return (firstHit, lastHit, score)
}
```

The implemented algorithm is more complex than the pseudo code as it keeps track of multiple possible start and end positions of a label in the word cluster and returns all the labels in the word cluster, whereas the pseudo code shown above only return one label and does not account for the possibility that there might be multiple possible interpretations for the same string (label candidates might overlap).

Classification The classification method uses the scores generated in the score phase and assigns the label to the class for which it got the highest score. In this step we also take labels from different classes that are overlapping. Again the label with highest score is kept.

The String "Camera-Ready Submission Deadline for Papers" could be interpreted as deadline for submission of the camera-ready version (interpreted substring "Camera-Ready Submission Deadline") and as submission deadline for the presentation version of the paper (interpreted substring "Submission Deadline for Papers"). When the possible interpretations overlap, the implemented algorithm returns the interpretation with highest score.

3.6 Establishing a Relationship between Label and Date

Having found the dates and labels on the website we now have to investigate the relationship between these. This can be done by going through all the labels and find the date that minimizes some distance measure. The easiest meaningful distance measure is probably the euclidean distance between the centers (center distance) of the boxes representing label and date. The problem with this approach is that these boxes tend to have a larger width than height. So the center-distance for two boxes

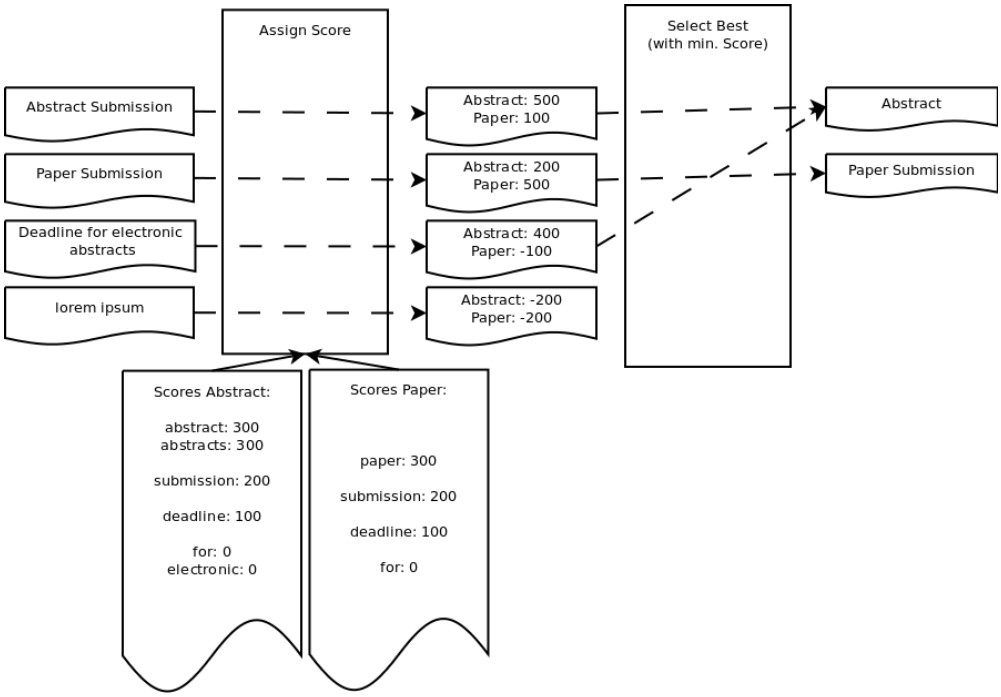

Figure 3.3: Two step procedure to extract the labels.

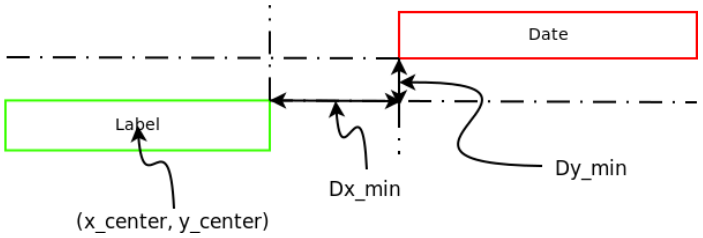

Figure 3.4: Difference between central euclidean distance and minimal distance.

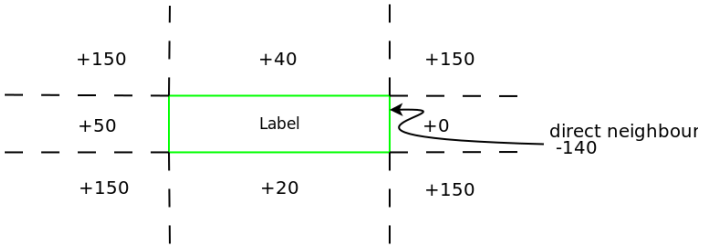

Figure 3.5: Additional penalty for the direction.

that are direct horizontal neighbors will be larger than the center-distance for the case when the boxes are vertical neighbors. This is unpleasant as the date is most likely to be on the right of the label.

Therefore we calculate the minimal distance between the boxes which is defined by the the euclidean distance of the corners that are closest to each other. The min-distance is defined as $d_{min} = \sqrt{[(\Delta x)_{min}]^2 + [(\Delta y)_{min}]^2}$ as illustrated in Figure 3.4. Still when having two direct neighbors, one right of the label, one above the label, we would like to prefer the one that is right, as the analysis of the conference websites show that the correct date is likely to be on the right of the label. To realize this we add an additional penalty for the direction which is depicted in Figure 3.5. In addition we prefer adjacent neighbors and therefore reduce the score if the two boxes are direct neighbors.

When the distance between label and date becomes too large, it is not likely that there exists a relationship between them, so we ignore pairs that have a calculated distance larger than a certain threshold.

3.7 Finding the Best Date-Label Pair

When establishing the relationship between Label and Date, we went through all the labels we found on the website and looked for their best partner. Now we have to find out which is the best of these pairs for every class.

This problem seems to be trivial at first, as one might be tempted to simply select the pair with the highest score. But this is problematic as we might use the same

	min	max
"Paper Submission Deadline" - "Notification of Acceptance"	15 days[1]	150 days[2]
"Notification of Acceptance" - "Camera-Ready Version"	6 days[3]	108 days[4]
"Paper Submission Deadline" - "Camera-Ready Version"	22 days[3]	175 days[5]

Table 3.2: Minimal and maximal time distance between two events as found my manually scanning 259 conference websites.

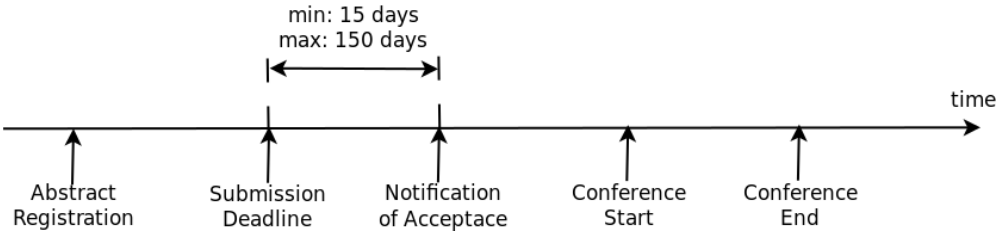

Figure 3.6: Minimal and maximal time difference between two events.

date for two different labels. To prevent doing this, we sort all possible pairs by their score in descending order. Then we take the pair with the highest score, make it the best pair for the specific class and remove all pairs from the list, that either involve the date we assigned to the first class or the class itself. By doing this repetitively for all labels, we have ensured that we only assign one date to one class and all dates are assigned at most once.

3.8 Checking for Plausibility

When directly accepting the label date pairs obtained from the above step, it might occur that the sequence of dates is invalid. In the simplest case: a paper cannot be accepted before it is handed in; so the Notification of Acceptance has to be after the Paper Submission Deadline. Analyzing the data set, that was manually extracted from 259 conference websites, we found the minimal and maximal time difference between two deadlines listed in Table 3.2

To make use of the knowledge we postulate a model in which a tuple of dates is only valid if all time differences between the events are in the intervals given in Table 3.2. By doing this we accept all data sets that are entirely "valid".

[1] euro ssc 2010, `http://www.esl.fim.uni-passau.de/eurossc2010/calls/calls.html`

[2] IEEE Infocom 2010, `http://www.ieee-infocom.org/2011/`

[3] CoNext 2010, `http://conferences.sigcomm.org/co-next/2010/Workshops/StudentWorkshop/cfp.html`

[4] ALGO 2010, `http://algo2010.csc.liv.ac.uk/waoa/`

[5] International Conference on Software Engineering, `http://2011.icse-conferences.org/important-dates`

A single outlier (e.g. the Notification of Acceptance is extracted incorrectly in a way that it now lies before the Paper Submission Deadline) would lead to rejection of the whole tuple. It is easy to evaluate that a time difference between two events is not valid (e.g. "Notification of Acceptance" is five days before the "Paper Submission Deadline"), but without further information it is impossible to tell which date is wrong.

To clean up the dates in a smoother way, we evaluate the time differences between all events and discard the dates for which both calculated time differences violate the model. Therefore we discard heavy outliers. If there are still time differences that violate the model, we discard both dates that contribute to the time difference, as we cannot tell which of them is wrong.

3.9 Special Case: Conference Date

Some of the conference websites only print their conference dates as part of the header image on the website. It is therefore not always possible to extract the conference dates without the help of OCR. The approach used to extract other important dates from the website does not work here, as most websites would not use a "standard" label such as "Conference date", but use the conference name as label (e.g. IEEE Infocom: April 10-15). Extracting the conference name first and run the date extraction procedure with a conference specific keyword list would make the system more vulnerable and decrease the performance of the overall system. Therefore we exploit the fact, that conferences are normally held for several days and the conference date is therefore displayed as time interval. We extract all time intervals found on the website and use the one that occurs most often.

Chapter 4

Unsuccessful Ideas

4.1 Table Extraction

Some of the conference websites present the important dates in the form of a table. One could use that structural information to make extraction more precise. The implementation of a modified (decreased number of necessary iterations) version of the T-Rex algorithm [13] worked well for cases in which the standard approach presented in this report worked well and had problems in the cases we could have benefited from being able to extract data from tables.

4.2 Column Splitting

Most websites are organized in multiple columns. Most common is the three column layout in which the left column is reserved for navigation, the middle column is for the actual content and the right column is used to display additional information. Being able to split the document into these columns and interpret each column as a separate scope, would eradicate the possibility that a label is incorrectly coupled with a date from another column. Using a simplified version of the algorithm proposed in "Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices" [14] didn't lead to satisfying results, as the algorithm implemented failed more often than we benefited from it.

Chapter 5

Results

To validate the approach we used a database dump from Confsearch with 259 conference websites. To establish a Ground Truth we checked every website in that database and updated the database accordingly. Furthermore we downloaded the websites and stored them locally so that we would not have to update the database whenever a conference is changing its website. Still "wget" was not able to download all the websites, so these websites are directly downloaded by the web browser whenever we run a test. This might lead to some faults when the websites are newer than our database. But it will most likely decrease the precision and therefore we can only underestimate the performance of our method.

The Hit-rate ($\frac{\#(correct\ result)+\#(incorrect\ result)}{\#(correct\ result)+\#(incorrect\ result)+\#(no\ result)}$) expresses in how many cases our approach returns a result and the precision expresses the percentage of correct results.

One can clearly see that the hit rate and the precision for the Abstract Class is very low. This is mainly due to the fact that the keyword list for that class is badly tuned. We would overcome this when using machine learning techniques to tune the keyword list.

The results for the other classes look better and we are confident that the precision of the method could still be increased with supervised machine learning and stemming [15]. Yet the low hit rate stays a problem.

Label	Hit-Rate	Precision
Abstract	.31	.21
Paper Submission	.48	.91
Notification of Acceptance	.51	.91
Camera Ready	.52	.88
Conference Start	.48	.84
Conference End	.50	.83

Table 5.1: Hit rate and precision obtained from the data set of Confsearch.

Chapter 6

Conclusions

Keyword List As became clear in the Results section, tuning of the keyword list used in label recognition is critical. The badly tuned keyword list for the abstract deadline class lead to an intolerably low precision. As the used method is basically a weak, deterministic approach for classification, one would expect to obtain a higher precision when using more sophisticated methods from the supervised machine learning domain. Precision and hit-rate could be further increased by methods like stemming [15] and lemmatization [16] as these methods would reduce the necessary size of the keyword list and make it more likely that a word in the word cluster is found in the keyword list.

Mechanical Turk The initial motivation for this thesis was to reduce the amount of time *highly educated* and/or *highly payed* people spend maintaining the website. The concept of a mechanical turk as advertised by various vendors would solve this problem in a different, yet more promising way. Even though there exist a small number of conference websites that require a diploma of higher education, having a human parsing the websites would always lead to a higher hit-rate (as defined in the results section) than obtained by the presented approach.

Application Even though the presented approach does not deliver a satisfying hit-rate, it one might still want to use the method, when the application of stemming and supervised machine learning lead to a precision greater .95. In that case the user would benefit from the application of the tool if it returns a result (as this result is precise) and the user experience wouldn't be harmed if the tool does not return a result.

Chapter 7

Acknowlegements

I would like to thank Prof. Roger Wattenhofer to support the Semester Thesis and Dr. Michael Kuhn for his great support productive criticism and his patience with me, when I tried something that would most likely fail. Furthermore I would like to thank Alexa McDorman for supporting me with the English language.

Bibliography

[1] M. Kuhn and R. Wattenhofer, "The layered world of scientific conferences," *Progess in WWW research and development: 10th Asia-Pacific Web Conference*, 2008.

[2] R. Grishman, "Information extraction: Techniques and challenges,"

[3] A. H. Laender, B. A. Riberio-Neto, A. S. da Silva, and J. S. Texeira, "A brief survey of web data extraction tools,"

[4] A. Sahuguet and F. Azavant, "Wysiwyg web wrapper factory (w4f)," in *Proceedings of WWW Conference*, 1999.

[5] L. Liu, C. Pu, and W. Han, "Xwrap: An xml-enabled wrapper construction system for web information sources," in *In ICDE*, pp. 611–621, 2000.

[6] V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: Towards automatic data extraction from large web sites," in *VLDB*, pp. 109–118, 2001.

[7] M. E. Califf and R. J. Mooney, "Relational learning of pattern-match rules for information extraction," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 328–334, 1999.

[8] N. Kushmerick, "Wrapper induction: Efficiency and expressiveness," *Artificial Intelligence*, vol. 118, no. 1-2, pp. 15–68, 2000.

[9] B. Adelberg, "Nodose & a tool for semi-automatically extracting structured and semistructured data from text documents," in *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, (New York, NY, USA), pp. 283–294, ACM, 1998.

[10] A. Waterson and A. Preece, "Verifying ontological commitment in knowledge-based systems," 1999.

[11] W. W. W. Consortium, "Document object model (dom)." `http://www.w3.org/DOM/`.

[12] "Same origin policy for javascript." `http://www.mozilla.org/projects/security/components/same-origin.html`.

[13] E. Oro and M. Ruffolo, "Pdf-trex: An approach for recognizing and extracting tables from pdf documents," in *10th International Conference on Document Analysis and Recognition*, 2010.

[14] Y. Chen, W.-Y. Ma, and H.-J. Zhang, "Detecting web page structure for adaptive viewing on small form factor devices," in *WWW '03 Proceedings of the 12th international conference on World Wide Web*, 2003.

[15] J. B. Lovins, "Development of a stemming algorithm," in *Mechanical Translation and Computational Linguistics*, vol. 11, 1968.

[16] J. Plisson, N. Lavrac, and D. Mladenic, "A rule based approach to word lemmatization," in *SiKDD 2004 at multiconference IS-2004*, 2004.