ConfSearch 2020

Bachelor’s Thesis

Lukas Schmid
luschmi@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Roland Schmid, Pankaj Khanchandani
Prof. Dr. Roger Wattenhofer

May 20, 2020
Acknowledgements

This paper came to life thanks to an idea of the Departement of Distributed Computing at ETH Zurich and my desire to make ConfSearch more easily accessible for both programmers and users. It was a long road of continuously improving a project built from scratch. I had many design-decisions to make and am very glad of the help Pankaj and Roland offered. Despite not being able to meet after the first few weeks, they kept providing insights remotely. Special thanks to friends and family that helped testing the website and always had an open ear for my ideas.
The goal of this paper is to improve the old website at confsearch.ethz.ch with a new and improved version. This required a thorough analysis of the existing website to determine what functionality was in place, how data and user input has been handled and what could be improved upon in future versions. Especially since there was no documentation or any record of the person that initially developed it.

As quoted from the task description, the goal is to help "detect and organize conferences, as it presents deadlines and other relevant metadata conveniently". A big focus here is on the automation. While we always will have to rely on the users for some inputs, there are a lot of reliable websites providing information about upcoming conferences and/or their deadlines. We will scrape some of the most recognized ones to fill the ConfSearch database with the data provided by them.

The project still has work to be done and many of the extensions from the task description could further enhance the appeal to the user. I’m happy with my work towards replicating the functionality of the old website and the transfer of the core features into a newer environment.
# Contents

## Acknowledgements

Abstract

1 Introduction

1.1 Related work

2 Analysis of ConfSearch

2.1 Design

2.2 Data handling

2.3 User functionality

3 Basics of the new website

3.1 Idea

3.2 Software

3.3 Django Configuration

3.4 Data Storage

3.5 Javascript libraries

3.6 The current state

4 Automated retrieval of conference data

4.1 Association for Computing Machinery

4.2 Guide2research

4.3 IEEE

4.4 Call For Papers Wiki

4.5 CORE Conference Portal

5 Design Decisions

5.1 Logical Separation
The department of Distributed Computing has a website to display upcoming conferences called ConfSearch. Being based on Java Server Pages with hardly any documentation or visible structure, it was decided that it needs to be rebuilt.

From the many available web-frameworks, python-based Django has been deemed the most suitable one. Similar python frameworks like Flask or Bottle didn’t have the depth required and weren’t as strong in both community support and projected longevity. A python framework was the preferred choice, as that is where I have the most experience in.

This paper will provide documentation and aim to describe what reasoning lies behind the decisions made in the design process. It will also provide an easier entry-point for future changes.

With the aforementioned issues, nobody knows how the old website worked and where it got the data from. That means that it was not possible to really maintain it. While the inner working of the old site weren’t fully mapped out, a basic understanding of the processes behind it was achieved and worked into the development of the new page. This paper will describe all the processes within and introduce a more straight-forward version.

1.1 Related work

There have been websites not unlike ConfSearch, many of which we are pulling data from.

One that stood out as a pretty complete one was from the "Laboratoire d’informatique de l’École polytechnique" hosted by Miki (Nicolas) Hermann.

An Email enquiry then revealed that the website is managed by him personally. He updates the website daily with information he gathered himself or that he received by email from anyone that wants to submit a conference. He listed the following reasons for the manual update:

1https://www.lix.polytechnique.fr/~hermann/conf.html
1. **Introduction**

- There are many people on the web who would just add junk, just to have pleasure from destroying my work
- I do not want predator conferences to be listed on my website
- I do not want other doubtful conferences listed there either

He further emphasized that the website only was supposed to enclose a very narrow range of conferences that belong to his field of study.
This chapter introduces the old ConfSearch page and the technology defining it. The design will be discussed, as well as how data was processed. I’ll further describe the functionality of the old site and its datastructures.

2.1 Design

As seen in the image above, the homepage is held quite simple, including mainly a search bar and the different scientific fields by which one can filter.

Figure 2.2 illustrates the main part of website, where the conferences are shown. Each colored part indicates some time constraints concerning the conference. Therefore every line between two bars represents a relevant date. The dates indicated are in the following order: Abstract registration, submission, notification, final version, conference start, conference end.

This represents all the steps that usually need to be done going from the interest
in a conference to have a paper presented there. The conferences that are more transparent don’t have dates defined for this year and are therefore represented by a shadow of last year’s data. The acronym on the left is a link that leads to a more detailed page, where past conferences can be seen and where the conference can be edited by anyone. An example of such a page is shown in Figure 2.3. The only fields that can’t be edited are the Top Keywords and the Rating.

2.2 Data handling

The database has proven quite extensive, even if some tables were hardly used or seem to have been there for debugging purposes. The database in figure 2.4 represents the website https://dblp.org/ from which data has been pulled and saved into a local database at some point. There was
2. Analysis of ConfSearch

no indication of the table being updated in any of the files found in the websites folder.

The database in figure 2.5 shows where the conference data from the website is stored. Any edits via the website’s interface are written here and anything displayed is read from here. The ConfSearch module in the website gets it’s data from this database, while the supporting modules DBLPAutomation and DBLPParser read data from local files into the database.

The main file here is MyDBLPBhtParser.java which parses the data provided from DataFiles DBLP dblp.xml which seems itself to be replaced manually if at all. The same folder also contains static files for LibraRank, RogerRank, CiteseerRank and ExtNHRank which are all displayed on each conferences detailpage. The module DBLPConferencegraph provides a minimum spanning tree over all conferences, connecting those that are closest based on common keywords. There also seems to be a GraphVisualizer which isn’t connected to the website.

2.3 User functionality

As discussed with the project supervisors who themselves are regular users of the confsearch website, the most important aspect of the site is the gantt chart. It lets users quickly identify the upcoming deadlines and the location of the conference, to help with deciding whether the conference fits the paper they want to release.
2. Analysis of ConfSearch

Figure 2.4: dblp database

Figure 2.5: confcal database
This chapter describes the idea behind the new website, the tools used and the basis future features are built upon.

3.1 Idea

The rework should make it even easier to decide on a venue to publish on. Due to the age of the old site, newer technology should be used, while still keeping the sleek design principle and retaining the user base. With Django as the new framework, the possibility of keeping the old frontend was considered. While that would have provided some benefits, the effort that would have had to go into creating an interface between the two different systems largely outweighed them. Various data sources that have been deemed reliable will provide data towards the website to both stay up-to-date and to encourage users to complete partial entries.

The user experience will also be improved in collaboration with two regular users, the supervisors Roland Schmid and Pankaj Khanchandani. Further potential improvements include recommendation methods, smart completion and user-assisted webscraping.

3.2 Software

While the development VM ran on Ubuntu 18.04, the Django application should be compatible with any system that can run Python 3.x and a compatible website hosting service.

The new website will be running on Django 3.0, the latest stable version with guaranteed long-term support.

Django is a python framework for websites, that is intuitive to code in, modular and clearly states where which part of the website is located. Being based on python, a very well-known and easy to read language, the code should be easy to understand for anyone working on this website in the future.
3. Basics of the new website

Both the framework and the language used provide extensive documentation and a lively community, with python being the 3rd most used tag on stackoverflow.com and Django being on rank 30 as of 18.05.2020. HTML, javascript and CSS ranking in between the two.

With a focus on data acquisition, python provided libraries such as requests for HTTPRequests and the external package beautifulsoup4 to parse HTML data.

3.3 Django Configuration

The new website lives in a django-project called confsearch. That relevant files in this folder are settings.py and urls.py, the other two files are relevant for running and deployment of the server at a later date.

The settings file contains the sites, which are allowed to host the website. Here we have both the IP-Address and the domain name, as well as the names for local access. Furthermore there’s a list of plugins called INSTALLED_APPS which contains the default Django modules with the addition of our modules landingpage and manageconf, as well as the module background_task, which manages the recurring tasks. The second list called MIDDLEWARE lists a number of default security measures that prevent various cyberattacks. The DATABASES list has the credentials for our default and only local postgres database. For security reasons, that database is not accessible from anywhere but localhost.

The urls file contains the information about where certain urls need to be redirected to. It furthermore contains the scheduled task, how often it should be recurring and the specific interval in which it gets called.

The project has two modules called landingpage and manageconf each in their respective folder. The former defines the website, while the latter holds the models and the option to extend into more detailed managing and smart proposals. The views file in the landingpage folder contains the views that are linked by in the urls file, those views always return a rendered template file, which usually is an HTML from the static folder.

The dataGathering folder is independent of the Django installation. It contains python modules that the recurring update uses.

3.4 Data Storage

The models currently present in manageconf are the conference, tag and author objects. While tag and author are not fully integrated into the front-end, the conference object holds all the fields necessary for the users information.

The database connection is in the config file as mentioned above, currently running PostgreSQL 10.12. The pg_hba.conf file specifies that only local connections are allowed to decrease the risk of outside entities gaining unwanted access.
3. Basics of the new website

Figure 3.1: Conference model

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>characters</td>
<td>characters</td>
</tr>
<tr>
<td>acronym</td>
<td>characters</td>
<td>abbreviation</td>
</tr>
<tr>
<td>authors</td>
<td>ManyToMany</td>
<td>authors of all the papers</td>
</tr>
<tr>
<td>rank</td>
<td>characters</td>
<td>the rating by the CORE portal</td>
</tr>
<tr>
<td>tags</td>
<td>ManyToMany</td>
<td>fields of research</td>
</tr>
<tr>
<td>address</td>
<td>characters</td>
<td>physical location</td>
</tr>
<tr>
<td>link</td>
<td>characters</td>
<td>webpage</td>
</tr>
<tr>
<td>abstractHandinDate</td>
<td>date</td>
<td>date for abstract submission</td>
</tr>
<tr>
<td>submissionDate</td>
<td>date</td>
<td>date for paper submission</td>
</tr>
<tr>
<td>startDate</td>
<td>date</td>
<td>date of the start</td>
</tr>
<tr>
<td>endDate</td>
<td>date</td>
<td>date of the end</td>
</tr>
<tr>
<td>isExtracted</td>
<td>boolean</td>
<td>Whether it was scraped</td>
</tr>
<tr>
<td>lastUpdated</td>
<td>date</td>
<td>date of last change for this entry</td>
</tr>
</tbody>
</table>

to the database. Commands with user input still need to be sanitized, as the website backend does have ownership of the confsearchdb database. The user is called luschmi and has no other privileges over other local databases.

3.5 Javascript libraries

The fuzzyset\(^1\) library was used to calculate the levenshtein distance between two strings to sort by similarity to the searchterm.

The JSGantt\(^2\) library provided a basic gantt chart to build upon. Major modifications had to be made to this module to modify the look and behaviour according to our goals. Among many different gantt chart libraries, this one had the most similarity with the idea we wanted to achieve and a documentation that promised a high degree of customizability.

---

\(^1\)https://glench.github.io/fuzzyset.js/

\(^2\)https://jsganttimproved.github.io/jsgantt-improved/
3. Basics of the new website

The jQuery\textsuperscript{3} library allowed for more dynamic changes to the document and its functions.

3.6 The current state

The website currently lives on a virtual machine hosted on *ee-tik-vm012.ethz.ch* and is being run as a development server. That means it is only reachable by localhost or SSH tunnels into the virtual machine, as it is neither secure nor stable in that state.

To be made available to external users, it needs to be installed on a webserver with a supported deployment platform. One of those can be found in the Django documentation\textsuperscript{4}.

Because the scraping of the websites happens in parallel, two separate processes need to be started for the full functionality.

\begin{verbatim}
python3 manage.py runserver 0.0.0.0:8000
python3 manage.py process_tasks
\end{verbatim}

The process\_tasks command comes from an external library called Django Background Tasks\textsuperscript{5} that allows to run commands in predefined intervals.

\begin{itemize}
\item \textsuperscript{3}https://jquery.com/
\item \textsuperscript{4}https://docs.djangoproject.com/en/2.1/howto/deployment/
\item \textsuperscript{5}https://django-background-tasks.readthedocs.io/
\end{itemize}
Chapter 4

Automated retrieval of conference data

This chapter describes the methods used to retrieve conference data from various websites. All the methods take no arguments and return a tuple containing the title of the conference, its acronym, the CORE-rank\(^1\), the latest submission date for the paper, the start date and end date of the conference, both the place and the country where it happens and finally the link to the conferences homepage.

4.1 Association for Computing Machinery

As is obvious from Figures 4.1\(^2\) and 4.2\(^3\) both websites are built very similarly, which also let’s the first part of the two scraping functions look pretty similar. Both websites HTML content is requested via the python requests package. Two parameters have been identified to be relevant.

\[ \text{startDate0} \text{ is a date in the format YYYYMMDD and identifies the month that will be displayed.} \]
\[ \text{view0} \text{ defines the format of the page which can either be day, week or month. To lessen the amount of requests required, we’ll leave it at month.} \]
\[ \text{eventType0} \text{ is CallsToPapers for the submissions page and Conferences for the conference-events page.} \]

Using beautifulsoup to get a queryable object, one can easily isolate all li tags with the "day" class. This leaves us with a list of pure day classes and some "day other-month" classes at both ends that indicate the previous and the next month.

We get the submission date from either counting through the pure day classes

\(^1\)http://www.core.edu.au/conference-portal
\(^2\)https://www.acm.org/conferences/upcoming-submission-deadlines?startDate0=20200520&eventType0=CallsForPapers
\(^3\)https://www.acm.org/conferences/conference-events?startDate0=20200520&eventType0=Conferences
or reading the number in the field, we can then extract all events by looking for links as a tags within the li objects we found. Those links directly point to the individual conference sites, which we can read directly from the href attribute. Furthermore we can split the title attribute at the ":" to get both the acronym and the conference title.

For the conference website we need to add a post-processing step, that detects consecutive days with the same conference, so they can be merged into one database entry.

4.2 **Guide2research**

This page\(^4\) has an even stronger partition into list elements, but does feature more condensed data which we need to separate. To get all the rows we can iterate over all the beautifulsoup objects that admit to the type *div* and the class "grey myshad" which is the specific coloring of the list elements. We do have to start from the second element though, as the first one contains the headings. Using a RegEx-Search we can identify the title and split it into the acronym and

\(^4\)http://www.guide2research.com/conferences/page-1
the conference-name. For the submission date on the right, the `datetime.strptime` function "%a %d %b %Y" can translate weekday, date, month and year into a date object, which we then write to the database. Using another RegEx-Search, we split the conference start-date, end-date and city into three elements, parsing the dates with the same formula as above. Another RegEx search then results in the country and therefore the final field from this page. Since this website has had malformed dates before, we make some sanity checks to verify they have the correct order and the conference has a reasonable length.

4.3 IEEE

While IEEE’s Computational Intelligence Society⁵ doesn’t provide submission dates for their papers, they still had a lot of information about conferences and the links to the webpages. Even though for the current extent of the website, this data isn’t that useful, it can still be completed by findings of other websites or user input.

The list consists of `div` elements with the class `conf-full-item`, which allows for easy extraction. Each row is formatted as "Conference (Acronym)" with it’s location in a `div` element and the dates in a span element. The dates are in the Unix Epoch⁶ format and therefore require a different parsing function.

⁵https://cis.ieee.org/conferences/conference-calendar
⁶Number of seconds since 00:00:00 UTC on 1 January 1970
4. Automated retrieval of conference data

Figure 4.3: Guide2Research’s conference page

4.4 Call For Papers Wiki

The HTML elements on this page are nearly indistinguishable, so I had to use some form of reliable index. Our assumption here is that there will always be 5 conference links in the Popular CFPs, so the sixth conference link can be used to identify the table relevant to our query. After getting the grand-grand-parent of that element, the rows have to be parsed pairwise, as each differently shaded row consists of two HTML-rows. The first yields both the acronym and the name of the conference, the second one provides start date, end date and the location.

4.5 CORE Conference Portal

This page provides the rankings displayed on the new website. While all the other pages had a limit on how far into the future we’d parse them, this page gets read fully every time. The page indicators at the bottom serve as indicator how many pages we’ll have to loop through. Iterating through all table rows except the header proves easy, as the website is very minimalistic, the rows of interest are the acronym used to match to the conferences already in our database and the rank. The rank is checked against it’s

---

7http://www.wikicfp.com/cfp/
8http://portal.core.edu.au/conf-ranks/?search=by=all
length, as there are four kinds of unranked conferences, including Australasian, Unranked, National and Regional. This function returns the acronym and the rank of the conference.
Figure 4.6: CORE’s conference page
This chapter will talk about the higher-level decisions made during development.

5.1 Logical Separation

To increase modularity of the code, every step from the conference data to the display on the webpage has its own space. In the folder called dataGathering, the conferences are read from their respective sites and coerced into the required datatypes and one global format, which will later allow us to iterate over all the generators in one loop. If some type of data is not provided from a specific source, the values will be returned as None, while still maintaining the agreed upon format.

The data then gets processed by the urls.py file in the main confsearch folder. This is the place where all the routing happens. It’s an interface to the outside world that defines where HTTP GET requests for the page go and where the

<table>
<thead>
<tr>
<th>title</th>
<th>G2R</th>
<th>ACM</th>
<th>IEEE</th>
<th>WikiCFP</th>
<th>CORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>acronym</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>rating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>submissionDate</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>startDate</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>endDate</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>location details</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>country</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>link</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Figure 5.1: Scraped conference data
data gathered from external websites goes. It’s folder contains all the configuration for the website in the settings.py file.

All parts concerning the actual website are located in the landingpage folder. The admin and models file import the models from the manageconf folder, where the model definitions are located. The views.py file defines all direct interactions between our webpage and the database. It also renders the webpage by replacing all Django level variables and code with the specified data. Since there is only one page, there is only one template in the templates folder, which defines the HTML and Javascript needed to display the data given by it’s view.

5.2 Visuals

The visuals were consciously kept simple without colorful CSS frameworks. The display focus is on Desktop PCs, no testing was done on smartphones. For the table and Gantt chart within, the JSGantt\footnote{https://jsganttimproved.github.io/jsgantt-improved/} open source library was used. It was heavily modified during development to adhere to our vision of the website. Only the rows crucial for making decisions about the conference were kept with some additional data displaying once the users hovers or clicks the conference. These were the location and rank, along with the acronym and timeline. The acronym links to the webpage of the conference to allow the users to verify the dates and quickly submit their paper. In case that no link was found, the users will be redirected to google, automatically searching for the conference in the current year. The look of the gantt bar was adjusted to mimic that of the original confsearch site.
Since the automated data collection will likely not catch every information there is about a conference, the website allows for users to complete existing entries with additional data.

6.1 Editing conferences

Each entry has a "Edit" button, that is accessible to any user on the website. The assumption is that people using the website will be willing to contribute to it, so the community can improve the amount of information available. If that functionality gets abused, a login system should be introduced to restrict access to verified users that can be held accountable for their edits.

Editing a conference will directly write the data to the database and notify the user that the operation was successful. Due to the current status of the gantt table, the website needs to be reloaded to display the changed data.

Once a conference has been edited, it will not get updated by the scripts anymore. Any conferences in their original form which get found with new data will get overwritten.

6.2 Creating conferences

The website currently doesn’t allow the creation of new conferences. If any noticeable conferences are missing, they can be created via the admin page, requiring a admin issued account.

Having all the biggest sites as data sources, we assume that all reputable conferences already exist and only need to be edited.
The goal was to create a website that improved upon the old one in terms of technology, presentation and data acquisition. Django introduced a more recent, more readable framework and language. It also allowed to strictly separate the different layers of application, so we have clear interfaces and every part can be changed independently of everything else.

The removal of the individual pages for each conference made the experience more predictable, but also reduced the amount of information gained. This was necessary to improve the efficiency of the users search.

Tags and the ability to search by field has temporarily been removed, but should be re-introduced if the need arises in the future.

7.1 The Gantt library

The javascript library JSGantt used in the website was responsible for both the table with the acronym, the location and the rating of the conference, as well as the Gantt chart in the right-hand side of the table. It worked in such a way, that the left and right table were disconnected and only seemed connected through the same CSS formatting that keeps both sides aligned.

This caused various issues late in the project when trying to sort the table. Those issues were solved in two different ways. The default sorting was shifted to the Django view module that delivered the data. Since the data was already retrieved sorted, the table would be sorted too once created. For local sorting recreating the gantt object is more efficient than sorting both sides individually while trying to ensure they line up. For future changes one might consider further changing the JSGantt library to create a globally consistent ID for both the tables and the edit functionality, so that only the HTML elements can be reordered to improve on the performance.

Another issue here is, that the library only allows one bar per task, so the displayed conferences are actually multiple tasks grouped up on one line. The grouping works by having them after each other in the list of tasks while the later tasks
reference the parent by its ID. For the sorting to work on that level, all children have to be removed, the parents get sorted and subsequently the children of each parent get added back. This could be improved in speed by adding the same data to all children and using a stable sorting algorithm. The data being handled this way also is a reason for the data not instantly being updated upon editing a conference. It would either have to update the javascript list entry and all its children or check which row is highlighted and translate the changes into changes there and a redrawing of the gantt bar on the right side.

With many features of the library unused and a plethora of changes made, it should be considered that writing a new library from scratch or finding one that fits this usecase better may improve future results.
While the main goals have been achieved, there still remains some room for improvements within the website. The design is more modern and presents the data without the extra steps required in the old version.

With the various websites providing data, there is no doubt that over time new corner cases will appear and require some sort of user intervention. With the possibility that every user can identify such issues and correct them, there is a lot of potential for a clean and complete database. While manually adding conferences is the exception now, some of the websites providing data might shut down. That should not impede the data collection, but may lead to some conferences going missing. With the structure built, newly appearing sites are easy to add and can be seamlessly integrated into the existing dataflow.

This automation shows that a lot of data can be gathered without requiring user interaction and only requires minimal interaction to stay up-to-date for an arbitrarily long time.