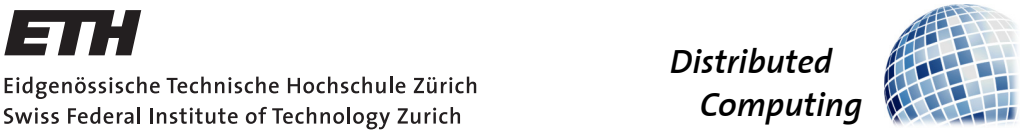

ConfSearch 2022

Bachelor's Thesis

Alex Thillen

athillen@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Dr. Ye Wang
Prof. Dr. Roger Wattenhofer

August 27, 2022

Acknowledgements

I thank Dr. Ye Wang and Prof. Dr. Roger Wattenhofer for supervising my thesis, giving me invaluable feedback, constructive advice and phenomenal guidance throughout the project. During our weekly meetings Ye has been able to nudge me into the right direction, whenever I encountered problems.
I thank Dr. Michael Kuhn for starting ConfSearch in 2007.
I thank all the people from the Distributed Computing Group for testing the web app, adding data prior to the release.
I also want to say thank you to all the people that will provide trusted data to the web app or even contribute to further developments and hereby allow *ConfSearch* to thrive for years to come.

Abstract

Finding a suitable conference to publish your paper can be a tedious endeavor. Conferences need to match the author's agenda, cover the right research topics and have a certain level of prestige.
For this reason, the Distributed Computing Group developed a tool called *ConfSearch* that allows users to find the right conference and display relevant metadata in a minimum amount of time and effort.
The goal of this thesis is to develop a renewed version of the 15-year-old search tool, that provides a more pleasing user experience, issues better search results and makes the project easier to maintain by using a new project architecture and employing state-of-the-art frameworks and technologies.
The updated web app is accessible at confsearch.ethz.ch. Further, we do a very basic qualitative analysis of the search results.

Contents

Chapter 1

Introduction

1.1 Motivation

In academia, researchers are mostly evaluated by their publications which they make at prestigious (peer-reviewed) conferences and in academic journals. Finding the right venue to publish your work, however, can be challenging due to the large variety of offered conferences out there. It is tedious to check out the targeted research focus, submission deadlines and other formal requirements for a submission – all while bearing in mind whether a publication at the selected venue will adequately underline the strength of the ideas presented in the research paper and lead a researcher to world-wide fame.

To resolve this issue, the *distributed computing group* has engineered a tool called *ConfSearch* that allows to search for keywords, authors or related conferences and proceeds by displaying matching conferences alongside meta-data such as a link to the conference web pages, the submission deadlines or the start of the conferences. To avoid confusion, the initial version shall be named *ConfSearch07* throughout the paper. Even though *ConfSearch07* is still a popular tool, one has to admit it's flaws.

The technologies used for the initial release have become dated. The initial web site was developed using *Jakarta Server Pages* (JSP), a collection of server-side programming technologies, released in 1999. JSP has lost in popularity over the last decade. It's market share is at 0.1%. [1] Reasons for the decline in popularity include the difficulty to debug and maintain JSP. Knowing that *ConfSearch07* has virtually zero documentation, it is unthinkable to try to adapt the old project.

From a user's perspective, the old web app looks like plain html for the most part and no longer lives up to today's user interface (UI) expectations. As is to be expected from a web page that predates the initial release of the iPhone, *ConfSearch's* usability on phones is poor. See Figure 1.1. Since the data used to populate the tables, stems from this era as well, the conferences available to the system are obsolete.

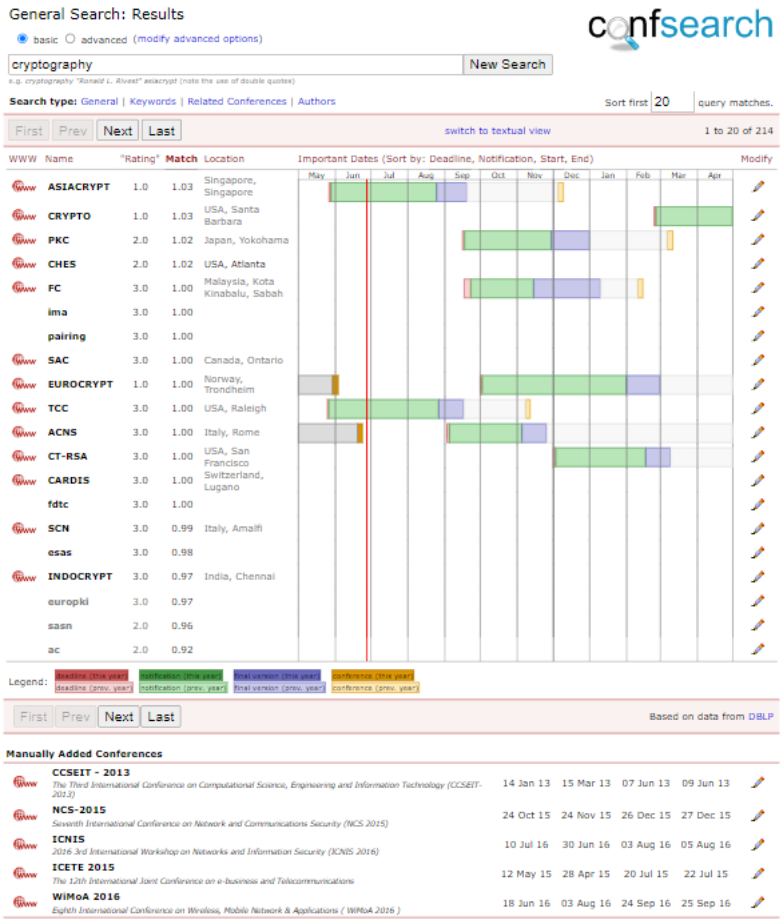

Figure 1.1: ConfSearch07 (viewed on Google Pixel 5)

To wrap up, the lack of documentation, the sheer outdatedness of the data, frontend and backend technologies used, yet a still active user base, make it clear that an update from *ConfSearch07* to a new version of *ConfSearch*, called *ConfSearch22* from now on, is much needed.

Chapter 2

Related Work

2.1 ConfSearch07

ConfSearch07 which serves as the foundation of this project has the same abstract goal, namely outsource the burden of figuring out at which conference to present a paper from the author to a search tool. By the means of easy-to-understand visualizations for when key events take place as well as displaying the most important information, such as rankings right next to it. *ConfSearch07* has been able to achieve this goal.

The visualizations of the key dates are arguably the main reason for the success of *ConfSearch* up until today. There is no similar website, that I know of, that displays key dates in such a concise way. This type of design that is usually known from music program (audio tracks) has therefore significantly influenced the new design. What makes it unique is that you immediately know which events take place before another event.

The search algorithm assures that the user is only provided with conferences that have the same research focus as the paper the user wants to publish. The search algorithm is based on layered graph models. [2] The project uses a 3-tier architecture. [3] I.e. the client/browser contacts a back-end server, that uses a database to compile an html web page, which it then forwards back to the client.

2.2 ConfSearch20

ConfSearch20 is a project started in scope of a Bachelor thesis by Lucas Schmid. [4] His goal was similar to mine - modernize and improve *ConfSearch07*. We were not able to run his project in a reasonable amount of time, there was also a lack of documentation inside his repository. This screenshot 2.1 of his design convinced us that we wanted to redesign the tool anyways. Next to modernizing the design, he also tried to automate information retrieval. The idea is that periodically the server runs scraping code that parses website information. We tested one of his websites to scrape and found that the URL used has turned

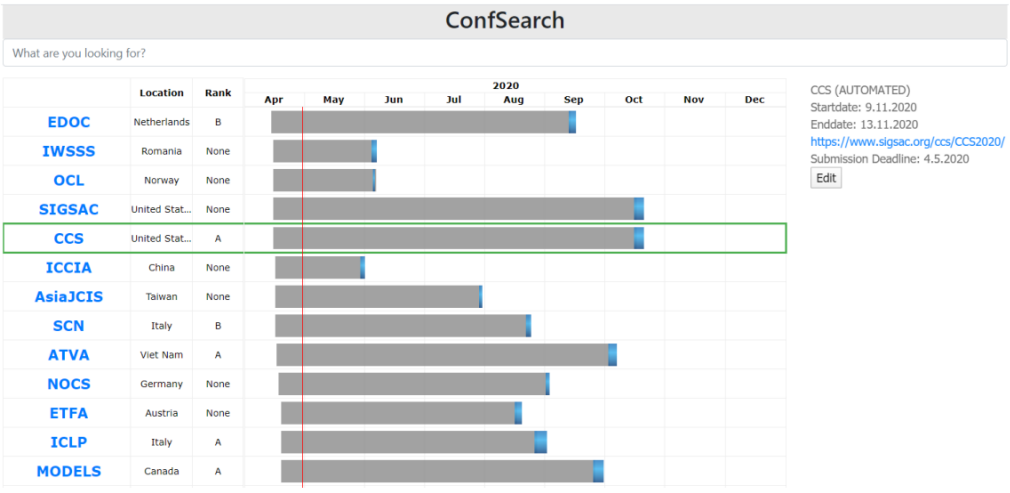

Figure 2.1: ConfSearch20

invalid in the meantime. This illustrates that web parsers are very susceptible to domain name changes or changes to the DOM. To prevent such issues we opted against making use of automated information retrieval for now, it would however be a nice to have feature for a future release. (See 5.1.2) The architecture used was Django for both the backend and the frontend, making use of the Django template language. Even though the project wasn't able to replace *ConfSearch07* it was able to provide me with invaluable lessons.

2.3 WikiCFP

WikiCFP[1] is a semantic wiki, which according to their website is used by over 100'000 researchers every month. It tackles the problem of gathering key dates for conferences, i.e. submission deadline, notification due date, final version due date, abstract registration due date, starting and end date of the conference. It's updated, as the word wiki implies, by its users. As the community that drives the wiki is very active, it contains the key dates of many conferences. We used WikiCFP for scraping many of the dates that can be found in the database initially. The use of information available on WikiCFP can be further expanded in future work, e.g. for doing suggestions when editing. (See 5.1.2)

[1] www.wikicfp.com/cfp/

Chapter 3

Procedure

3.1 Overview

The architectural decisions we have made during the development of *ConfSearch22* have been influenced by my analysis of both the strong points and the weaknesses of *ConfSeach07* and *ConfSearch20*. Both of the previous versions have turned out to be very difficult to maintain, improve and analyze. They both have an old-fashioned user interface.

What the two previous version have in common, is their architecture. In both cases, the user sends a request to some back-end server, this back-end server takes care of receiving the request, accessing the database, compiling a web page that includes the results and returning that web page to the user. This means that the design logic is also programmed on this single server, even though backend technologies are known to not be particularly good at this. In *ConfSearch20*, the back-end server additionally tries to scrape some hard-coded URLs for new information to populate its database using the so called `background_tasks` extension. See figure 3.1 for an illustration.

Having this kind of complex intertwining of functionality is in my opinion the root cause for why the previous projects are so difficult to build on and why some of the functionality, most notably the design or the periodic scraping for data, performed by a single module turn out to only have mediocre results. In this regard, I'm on the same page as Edsger W. Dijkstra, who was an advocate for

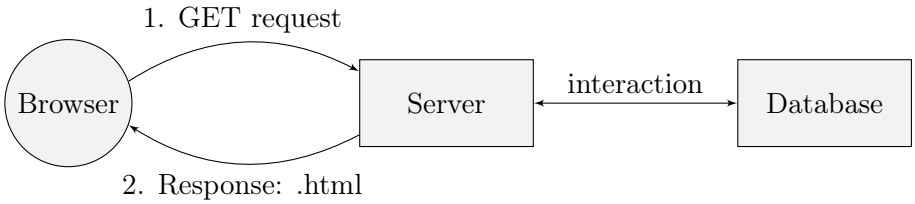

Figure 3.1: Architecture of *ConfSearch07* and *ConfSearch20*

the separation of concerns (SoC), a design principle that consists of separating computer programs into distinct sections.

> Edsger W. Dijkstra Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects. (...) But nothing is gained—on the contrary!—by tackling these various aspects simultaneously. It is what I sometimes have called "the separation of concerns", which, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts, that I know of. [5]
>
> - Edsger W. Dijkstra

This is why, we chose a radical different approach to structure our project. Instead of having one big module that takes care of virtually everything from design to web scraping, we split our project into three separate modules. This way we are not only able to benefit from all of the classical advantages of modularity including the ability to reuse code or the improved readability of programs, but it further allows us to use the best and most suited technology out there for implementing the different functionalities, not needing to pay attention to whether it fits some rigid development stack.

The basic structure consists of three units. The first unit is the *information retrieval module* and consists of the logic required to fetch all of the data needed to build the search engine around it. It takes care of getting keywords for conferences, scrapes the web for key dates, locations and ranks. The second unit is the *backend or API module*, it takes care of handling user request that need to interact with the database, such as search requests or requests to change the meta-data of a specific conference. The last unit is the *frontend module*. It is in charge of providing a nice user interface, while exchanging raw data with the API. Each of these aforementioned modules can be replaced in a future version, provided that they respect the current API conventions.

The new architecture is visualized in figure 3.2. Since information retrieval only happens prior to the user base interacting with the web application, it is not represented. Technically, it interacts with the database as well. As will be seen later, we will also need a reverse proxy to deploy the application. For simplicity, this is left out for now.

We will not only improve existing functionality, but we will also develop new ones. Among other things, we will give users the option to download the calendar dates by generating a .ics file.

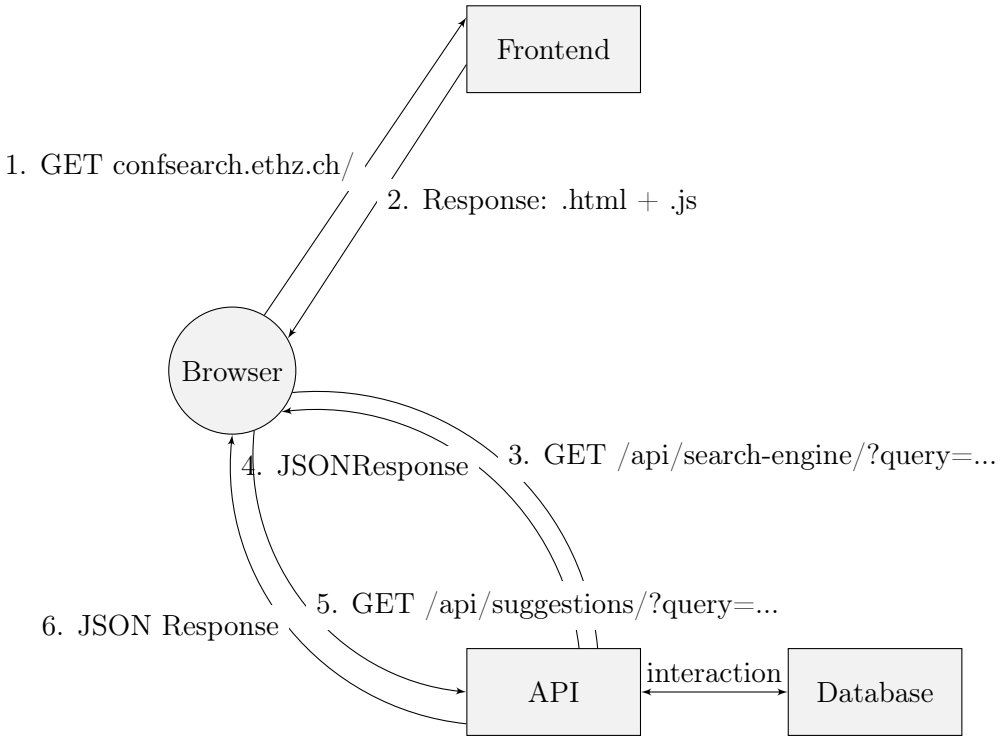

Figure 3.2: Architecture of *ConfSearch22*

3.2 Information retrieval

3.2.1 Goal

The goal of the information retrieval module consists of gathering all the data required to launch *ConfSearch22*. To enable the search engine, we use keywords. Each conference is associated with a list of keywords, that is generated based on the paper titles that were published at the given conference.

We also need the authors and the dates of their papers to enable author search. Further, we need the important dates (deadline, notification, start of the event), the location, the website of the conference and the CORE ranking in order to be able to display all the relevant information to the user.

Thus, the data required consists of the following:

- A list of all the papers, including the conference at which those papers were presented.
- A list of conferences and their associated ranking based on CORE.

- A list of conferences and the matching important dates.

3.2.2 Paper titles

dblp.org has a very complete collection of papers and meta-data on these. Users can simply download this collection as a big `.xml` file. Due to the size of the `.xml` file, some parsers such as the provided by the iconic BeautifulSoup4 python library are unable to cope with the files this big. I finished by minimally modifying an existing open-source dblp parser based on the lxml python library.

The following `xml` entries are of interest to us: inproceedings and proceedings. In simple terms, inproceedings correspond to articles published inside a proceeding. A proceeding of a conference is a collection of papers that were accepted at this conference. Here are two examples copied from dblp[6]. We marked the important fields.

```
<inproceedings key="conf/er/Norrie08"
mdate="2008-10-20">
  <author>Moira C. Norrie</author> <!-- important -->
  <title>PIM Meets Web 2.0.</title> <!-- important -->
  <pages>15-25</pages>
  <year>2008</year> <!-- important -->
  <booktitle>ER</booktitle> <!-- important -->
  <ee>http://dx.doi.org/10.1007/
978-3-540-87877-3 3</ee>
  <crossref>conf/er/2008</crossref>
  <url>db/conf/er/er2008.html#Norrie08</url>
</inproceedings>
```

```
<proceedings key="conf/er/2008"
mdate="2008-10-20">
  <editor>Qing Li</editor>
  <editor>Stefano Spaccapietra</editor>
  <editor>Eric Yu</editor>
  <editor>Antoni Oliv&eacute;</editor>
  <title>Conceptual Modeling - ER 2008,
27th International Conference on Conceptual
Modeling, Barcelona, Spain, October 20-24,
2008. Proceedings</title> <!-- important -->
  <volume>5231</volume>
  <year>2008</year>
  <isbn>978-3-540-87876-6</isbn>
  <booktitle>ER</booktitle> <!-- important -->
  <series href="db/journals/lncs.html">Lecture
Notes in Computer Science</series>
  <publisher>Springer</publisher>
  <url>db/conf/er/er2008.html</url>
</proceedings>
```

The *author* corresponds to a person that has written or contributed to writing the article. The *title* corresponds to the title of the article. The *year* to the year

Conference Name	Acronym

Table 3.1: Conference Table

Article Title	Authors	Year	Acronym

Table 3.2: Inproceeding Table

of publication. The *booktitle* contains the acronym of the conference at which the article was published. The *crossref* field contains a reference to corresponding dblp proceeding in the database. One has to point out that the data is partially corrupted, frequently there are conference names, instead of the acronym being used as booktitle or there are used several booktitles alongside some location as booktitles. I.e. the data required lots of processing before being usable.

The `title` of the proceeding corresponds to the title of a book and not the name of the conference itself. However, we want to match the papers with the corresponding conference, thus we are interested in the name and acronym. Therefore, we decided to get a list of conference names and the corresponding acronym elsewhere. Namely from a seperate list available on dblp[1] as well. The idea was that we could simply match papers with their corresponding conferences, based on the acronym. This turned out to be a very tedious task as we will see shortly.

The first hurdle was, that the information scraped from the list comes in no coherent ways. Three frequent formatting styles include:

- 3D Data Processing Visualization and Transmission (3DPVT)
- 3DIPM - Three-Dimensional Image Processing, Measurement, and Applications
- Adaptive and Learning Agents and Multi-Agent Systems; European Symposium on ... (ALAMAS)

After parsing all the different formatting styles, we were left with a table containing acronym and name.

The next and way more devastating challenge are *ambiguous acronyms*.

Definition 3.1 (Ambiguous acronyms)**.** An acronym that is used by two or more conferences.

Our plan to simply match papers with there respective conference based on the acronym used fails as some conferences use the same acronym as others. For example 'SAM' is the acronym of 6 different conferences, among those are : 'Software Audit and Metrics', 'International Conference on Security and Management'

[1] https://dblp.org/db/conf/index.html

Article Title	Authors	Year	Acronym	Conference Name

Table 3.3: Parsed DBLP Table

Acronym	Conference Name	Deadline	Start Date	End Date	Location

Table 3.4: WikiCFP Table

and 'IEEE Sensor Array and Multichannel Signal Processing Workshop'. About one in ten conferences uses an ambiguous acronym.

This combined with the fact that many of the booktitles provided by the dblp database did not correspond to the matching acronym and our scripts not being able to recover all of the acronyms, required a different approach. We matched all of the entries we could, i.e. that the `xml` file contained correct data for and that did not use *ambiguous acronyms*. This sufficed for about 3/5 of the total number of articles. For the remaining entries, we wrote a script that scrapes dblp by searching for the paper titles and hereby is able to retrieve the respective conference names and acronyms. Since, this had to be performed on almost a million entries, we included some tricks to speed up the process and not having to do 1 millions requests.

At the end of this process, we were left with a table containing the fields represented in 3.3 for each of the papers.

3.2.3 Dates & CORE Ranking

Now, that we have all the information needed to build the search engine, we needed to get the data the users are interested in. Namely, the CORE ranking and the important dates.

For both these ventures, we had to scrape the respective web pages of WikiCFP[2] and CORE[3].

WikiCFP

For WikiCFP, we wrote a script that for each of the conference names available in our dblp database table, sends a search request to WikiCFP. It then proceeds to parse the table that WikiCFP outputs for each search request. The parsing is done using BeautifulSoup4, which is an easy to use html/xml parser. After parsing, the script inserts the new data into a database table of the form seen in 3.4.

[2]http://www.wikicfp.com/cfp/
[3]http://portal.core.edu.au/conf-ranks

To merge the two tables using a left join is again not possible as we again have the *ambiguous acronym problem*. Also WikiCFP does not use the same conference name as dblp does, but rather it uses the event title. Since WikiCFP relies on user data, the naming doesn't follow any clear conventions. Let me give you a few examples 3.5.

Conference Name (dblp)	Acronym (dblp)
Dependable Systems and Networks	DSN
Symposium on Computer	SBAC-PAD

Conference Name (WikiCFP)	Acronym (WikiCFP)
The 52nd Annual IEEE/IFIP International Conference on Dep ...	DSN 2022
The 34th IEEE International Symposium on Computer	SBAC-PAD 2022

Table 3.5: Comparison of names and acronym in WikiCFP vs. DBLP

Even though in 3.5 the acronym of WikiCFP appears to be '<acronym> <year>', this is notalways the case. We resolved this problem by doing some sophisticated matching between dblp entries and WikiCFP entries based on both the name and the acronym. Among other things, we removed the numbers from the strings, checked how long the longest shared sub-string of the two conference names were compared to the size of the strings and whether the acronym of dblp is contained in the acronym of the WikiCFP. In the end, we ended up with a decent number of dates for my conferences.

It's also worth pointing out that the quality and quantity of the data contained on WikiCFP seems to vary over time, e.g. during the initial scraping the A* conference 'ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing' appeared on WikiCFP. During a re-scraping it had disappeared from the Wiki only to reappear a couple of days later. This is additional motivation for not using repetitive scraping to update data.

CORE Ranking

I used a similar approach to get the data from CORE Ranking. The problems turned out to be very similar as well:

- different spelling for the same conferences.
- ambiguous acronyms.

I used similar techniques to resolve matching issues as in WikiCFP.

At the end of both these stages we were left with a table containing the following entries:

- conference name
- conference acronym
- deadline
- start & end
- location
- CORE rank

combined with the article table, consisting of the conference name, acronym, paper title and authors, we were able to construct a table having all the relevant information for building the search engine and displaying relevant data to the user.

Since, the conference names fortunately are not ambiguous, we were able to join on the conference name, yielding a table with the following fields.

- paper title
- paper authors
- conference name
- conference acronym
- deadline
- start & end
- location
- CORE rank
- WWW (conference website)

This table serves as the foundation of the backend, we are going to build next.

3.3 Backend & Search Engine

3.3.1 Setup & API Features / Goals

The backend is written in python using the Django framework, but it can easily be rewritten in any other language or framework. We chose Django as it is easy-to-use, python is my favorite language and it is in the top 10 of the most popular web frameworks according to the StackOverflow Developer Survey of 2022.[7]

The goal of the backend is to expose an API to the frontend. The user visits the website of *ConfSearch22* and retrieves the frontend. The frontend consists of all the code contributing to the graphical user interface that the user sees, it is executed in the user's browser. Depending on the actions the user takes using the user interface, the frontend sends requests to the API in order to receive data that can then be shown inside the frontend.

The final API supports five kinds of requests:

- Editing POST requests that allow to change details of a conference.
- Add POST requests that allow to add a new conference.
- Compute keywords POST requests that allow to compute weighted keywords.
- Search GET requests that allows to retrieve all the conferences that match the keywords, are related to the searched for author or conference or correspond to acronym keys - a kind of special bookmarking tags associated with each conference.
- Detail GET requests that allow to get all the details about a single conference.
- Search suggestion GET requests that allow to give hints to the user. If he searched for an author but misspelled, the suggestion will point him to the correct search query. The same holds for conference names.

I will not dive into every single one of these features. I will explain on a high level how the search functionality works.

3.3.2 Keyword Search : Generation of keywords and weights

Once the request enters the backend server, it is checked whether it consists of *bookmarking tags* aka *acronym keys* only, an *author name* or a *conference name*. If all of these checks fail, keyword search is performed.

We first create a mapping W for each conference that returns a weight for each keyword. This mapping is created as follows. We first go over the titles of all the papers and create a histogram of all the words that appear, called global histogram. In the nest step we create a histogram for each individual conference, called local histogram or histogram for conference c. Then we create a matching algorithm that upon receiving a query, returns a weight for each conference that represents how good the conference matches the keywords.

Definition 3.2 (Global histogram)**.** A mapping from a word to the number of occurrences of this word in all the inproceeding titles.

$$GH : \{Words\} \mapsto \mathbb{N} \tag{3.1}$$

$$GH(k) = \#\ occurrences\ of\ word\ k\ in\ all\ inproceeding\ titles \tag{3.2}$$

Then we create a histogram for each conference. By going over the titles of each inproceeding individually.

Definition 3.3 (Histogram for conference c_i)**.**

$$H_{c_i} : \{Words\} \mapsto \mathbb{N} \tag{3.3}$$

$$H_{c_i}(k) = \#\ occurrences\ of\ word\ k\ in\ inproceeding\ titles\ published\ at\ c_i \tag{3.4}$$

Further, we define the cardinality of the histogram as follows:

$$|H_{c_i}| = \sum_{k \in Words} H_{c_i}(k) \tag{3.5}$$

On the basis of these two histograms, we will then generate a weight associated with each keyword.

Definition 3.4 (Importance of keyword k at conference c_i)**.**

$$I_{c_i} : \{Words\} \mapsto \mathbb{R} \tag{3.6}$$

$$I_{c_i}(k) = \sqrt[1.4]{\frac{H_{c_i}(k)}{50 + GH(k)^{1.15}}} \tag{3.7}$$

The reasoning behind the importance is as follows:

$H_{c_i}(k)/GH(k)$ gives the share of appearances of a given word k at conference c_i with respect to the total number of appearances in all conferences and is thus a proxy of the relative importance of that word in conference c_i taking into account the general frequency of that word.

To account for very rare words, we add 50 as normalization constant. Otherwise if a word appeared only once in total, it would get a very high importance compared to other words. Since, we want to add an additional penalty to very frequent words, we also add 1.15. The motivation behind this is as follows. Assume that we have:

- $GH(\text{"}distributed\text{"}) = 10'000$
- $H_{c_i}(\text{"}distributed\text{"}) = 1'000$
- $GH(\text{"}the\text{"}) = 400'000$

- $H_{c_i}(\text{"}the\text{"}) = 40'000$

, then in both cases we would get the same importance[4] of

$$\frac{H_{c_i}(k)}{GH(k)^1} = 0.1$$

. However, 'distributed' is way more meaningful. Using the additional penalty of adding a power of 1.15, we get

$$\frac{H_{c_i}(\text{"}distributed\text{"})}{GH(\text{"}distributed\text{"})^{1.15}} \approx 0.025 > 0.0144 \approx \frac{H_{c_i}(\text{"}the\text{"})}{GH(\text{"}the\text{"})^{1.15}}$$

.

The 1.4 is of the root is meant to prevent some keywords from becoming extremely important, while preserving the relative ordering of the keywords.

Definition 3.5. [Weight of keyword k at conference c_i] Let A_i be the array that contains the 3000 most important keywords of conference c_i. Let $|I_{c_i}| = \sum_{k \in A} I_{c_i}(k)$ be summed importances of the most important keywords. Then the weight of keyword k at conference c_i is defined by

$$W_{c_i}(k) = \begin{cases} 3000 \cdot \frac{I_{c_i}(k)}{|I_{c_i}|} & , k \in A_i \\ 0 & , else \end{cases} \tag{3.8}$$

This weight gives at the same time a good proxy for the importance of the keyword, while assuring that

$$\forall c_i, c_j . |W_{c_i}| = |W_{c_j}| = 3000$$

, where $|W_{c_i}| = \sum_{k \in W} W_{c_i}(k)$. I.e. each conference has the same relevance or total weight. One has to note that for the support of conferences with less than 3000 keywords, some extra steps are required. This only happens for very little training data.

3.3.3 Keyword Search : Matching keywords

Definition 3.6 (Match of query)**.** Let $k = [k_1, ..., k_n]$ be the list of keywords that the user submitted. Further let $k^1 = [k_1]$, $k^2 = [k_1, k_2]$, $k^n = k$ and so on. We define the match of a conference c_i with a sequence of keywords k as $\bar{M}_{c_i}(k)$.

$$\bar{W}_{c_i}(k) = \frac{W_{c_i}(k)}{max_{c_i \in \ Conferences} W_{c_i}(k)} \in [0, 1] \tag{3.9}$$

[4] We leave out the 50 as normalization factor for the sake of simplicity

$$\bar{M}_{c_i}(k) = \frac{M_{c_i}(k)}{max_{c_i \in \ Conferences} M_{c_i}(k)} \in [0,1] \tag{3.10}$$

$$M_{c_i}(k^n) = \begin{cases} \bar{W}_{c_i}(k_n) + 4 \cdot \sqrt{\bar{W}_{c_i}(k_n) \cdot \bar{M}_{c_i}(k^{n-1})} + \bar{M}_{c_i}(k^{n-1}) & n > 0 \\ 0 & else \end{cases} \tag{3.11}$$

The intuition behind this matching is that we take into account the weight of each keyword that matches, but give strong preference to conferences for which multiple keywords match.

In the real implementation, we also give a *dynamic boost* to conferences that have a high ranking. The dynamic boost factor is given by

$$b = \frac{1}{max_{c_i \in Conference\ with\ rank\ =\ A*,\ A,\ B} \bar{M}_{c_i}(k)}$$

, which is an indicator of the distance between the best matching conferences and the best matching, well-ranked conference. The boosted conference then gets an updated weight according to:

$$\bar{M}_{c_i}^{boosted}(k) = \left(\bar{M}_{c_i}(k) \cdot b \cdot r + \bar{M}_{c_i}(k) \cdot (1-r)\right)^{r'}$$

. The constants r and r' depend on the rank. The intuition on why we have to boost famous conferences is twofold:

- it is more likely that people want to have higher ranked conferences in their search results.
- higher ranked conferences tend to cover a broader spectrum of research topics and are thus less sensitive to specific keywords.

3.3.4 Author Search

Assume that the user searches for an author name a_j. Then we want to output the conferences that are most relevant for this author. The more often and the more recent author a_j has published at a conference c_i, the more relevant this conference is to him.

Let $t_{a_j,earliest}$ denote the earliest year in which author a_j has published an article. In the same way, let $t_{a_j,recent}$ denote the most recent year in which author a_j has published an article. Let $t(p)$ denote denote the year at which paper p was published. Further, we say $p \in c_i$ if p was published at conference c_i, similarly we say $p \in a_j$ if p was published by author a_j.

Definition 3.7 (Weight of paper p written by author a_j based on linear interpolation)**.** Let a_j be the be the author of the paper p. Then we say that the weight

of p for author a_i is given by:

$$w_{a_j}(p) = \frac{t(p) - t_{a_j,earliest}}{max(1, t_{a_j,recent} - t_{a_j,earliest})} + 0.1 \tag{3.12}$$

This type of weight is higher for newer conferences, but never 0.

Definition 3.8 (Author conference weight)**.** The author conference weight is an indicator of the relevancy of conference c_i for author a_j:

$$A_{a_j}(c_i) = \sum_{p \in c_i, p \in a_j} w_{a_j}(p) \tag{3.13}$$

Whenever, the user searches for an author, we output the conferences sorted by the *Author conference weight* which is an indicator for relevancy.

3.3.5 Other search queries : Bookmarking and related conferences

There are two more search queries not yet discussed. One of the two has the purpose to generate URLs for bookmarking. Basically, every conference is associated with a key that uniquely identifies the conference - this key is generated based on the acronym. We call these keys acronym keys or bookmarking tags. Whenever the user enters two or more of these bookmarking tags, he is displayed the exact conferences these conferences refer to.

The other type of search query is for looking up related conferences. The user either enters exactly one acronym key or he enters the exact name of a conference. For this type of request, the backend looks up the top 25 keywords of the referred to conference. Let v_{c_i} be the weight vector of these 25 keywords for conferences according to 3.5. The indicator for how related / matching some conference c_j is to c_i is given by

$$v_{c_i}^T v_{c_j}$$

.

In all of these searches, the backend sends the top 100 best matching conferences to the frontend running on the client. Except for the bookmarking requests - they have no matching algorithm.

3.4 Frontend

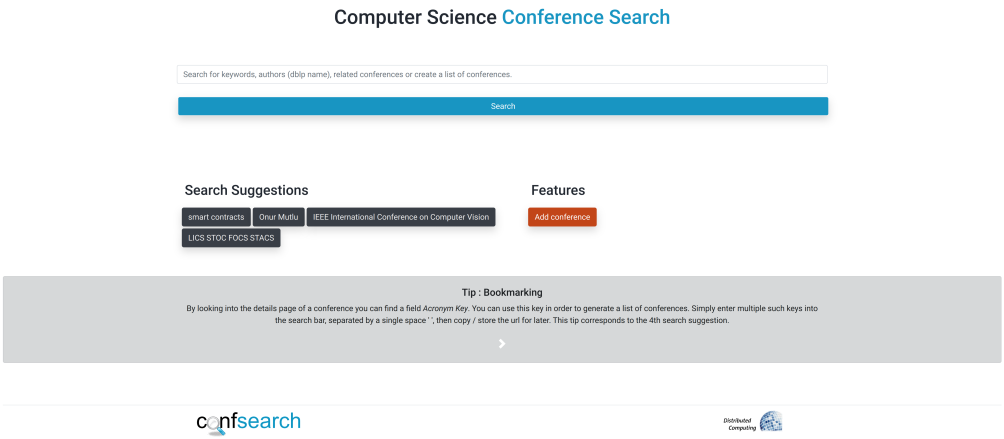

Figure 3.3: ConfSearch22 : Home Page

3.4.1 Setup

The frontend of a web site consists of the parts that users interact with directly when surfing a website, it is thus the most important piece when it comes to user experience. To build the frontend we used the ReactJS framework. ReactJS, thanks to it's components, perfectly fits the modular architecture of the rest of the project. Furthermore, it has been deemed the most popular and second most popular web framework in the years 2021 and 2022 respectively, according to the StackOverflow Developer Survey.[8, 7] To build a responsive user interface (UI), we used Bootstrap CSS styles as the basis of our frontend. Bootstrap is known to run well on all types of end-devices, it is delivered by fast content-delivery networks (CDN) and is fully customizable and modifiable. Using it allowed us to save a lot of time.

When developing a web site, it all boils down to maximizing the UX. The doctrine that guided us in building the frontend web application is simplicity. Nielsen's Usability Heuristics[9] proved especially useful to auto-evaluate the design, without having to do expensive user studies for which we had neither the time nor the human resources.

3.5 Features / Goals

In an as easy-to-use as possible way, we want to expose to the user the following features:

- search for keywords, authors and related conferences
- bookmarking allows store a list of selected conferences
- visualization of the most important dates for conferences
- easy to compare different conferences with respect to different metrics, such as the rank or the deadline.
- sort conferences according to metrics like the rank or the deadline
- select a subset of the conferences obtained from searching, for further processing.
- users can use an edit page to enter new information or update old information. They can also add a new conference using the add conference page.
- the calendar icon allows the user to download the key dates in the form of a .ics file.

3.6 Design Choices

To make the interface as easy-to-use and as intuitive as possible, we have put a special focus on not cluttering the screen with information. With the use of tool-tips, boxes that pop-up whenever the user hovers over a certain area, and information boxes, boxes that pop-up after the user takes some action, we are able to provide the user with all the information he needs at the right time, while keeping a clean surface.

Further, we provide several suggestions (see 3.3) and tips on the home screen or when editing. These suggestions make sure that the user never becomes stuck, because he is not capable of performing the desired action.

To illustrate the effectiveness of our design choices, we will how they conform with the different usability heuristics provided by Nielsen.[9, 10].

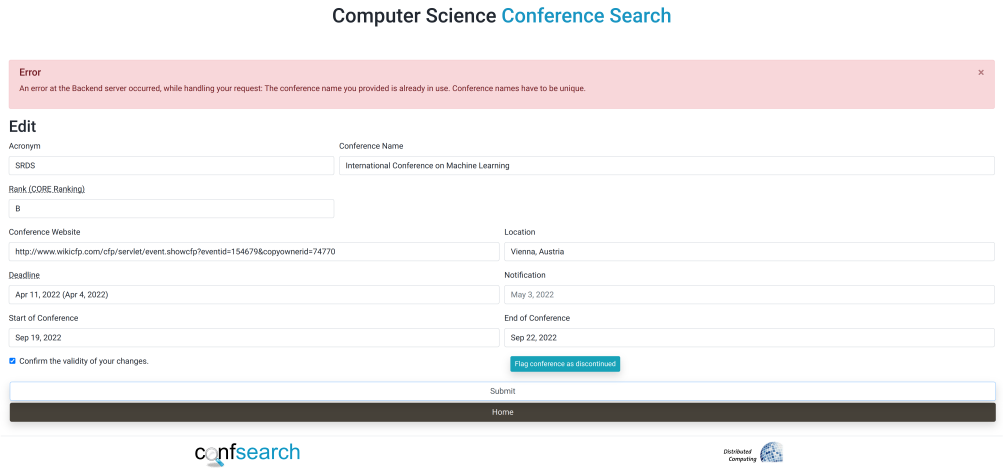

Figure 3.4: ConfSearch22 : Edit Page

- Visibility of system status : Through the use of information boxes (see 3.4), we inform the user whenever he completes a request with side-effects. If he edits a conference, he is informed on whether the changes have been submitted successfully, analogously he is informed about why his request failed and what actions he needs to perform next.
- Match between system and the real world : By using icons that are symbolic for events in the real-world3.5, it is ensured that user needs little time to adapt to the new interface.
- User control and freedom : At every stage, the user is able to click on the web page title to return the homepage. Further, he can use a special home button, whenever he is in the midst of performing critical action like adding a new conference.
- Consistency and standards : By the use of bootstrap, the styling is ensured to follow classical design standards. Further, we ensure that the website is consistent in structure compared to the old web site. This further reduces adaptation times.
- Error prevention : By requiring the user to confirm his changes, we ensure that the user checks if he committed any mistakes himself.
- Recognition rather than recall : We reuse designs across the board. This is is made especially easy thank to ReactJS's components. E.g. the editing, details and add conference page are all based on the same style.

- Aesthetic and minimalist design : A simplistic design has been our mantra throughout the project. It should be clear from the screenshots that we did not include any unnecessary information.
- Help users recognize, diagnose, and recover from errors : This is again ensured through the use of special error boxes. (See 3.4)
- Help and documentation : For users that need more information, the tips on the home page provide detailed information on how to use the different features provided.

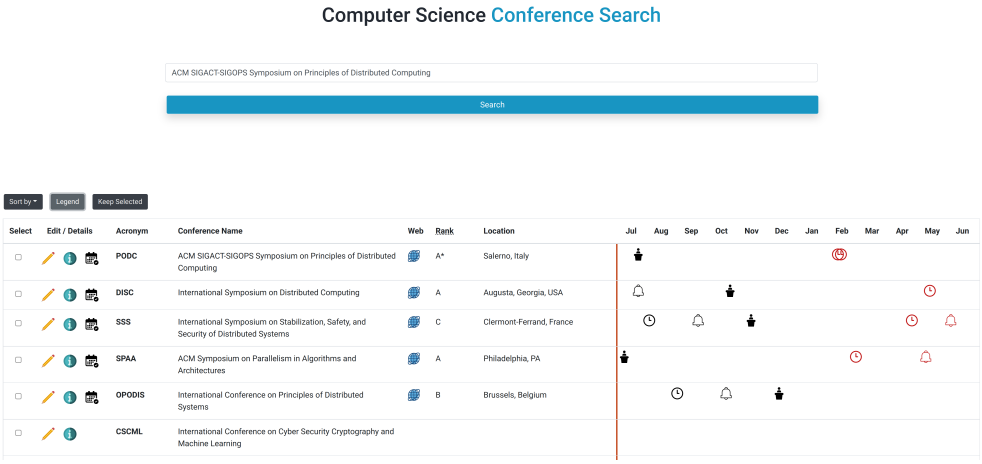

Figure 3.5: ConfSearch22 : Search

The design is a make-or-break characteristic that determines the success of a web page. Through the use well-known usability principles and renowned frameworks, we have been able to design a pleasing tool both from an optical and a practical point of view.

3.7 Deployment

In order to deploy our application and go into production, some minor changes to the architecture had to be made. Namely, since both the frontend and the Django API need to be accessible on the same domain, we had to configure a reverse proxy. All requests have to be directed to the frontend, except if the path starts with '/api/'. The architecture thus looks as follows:

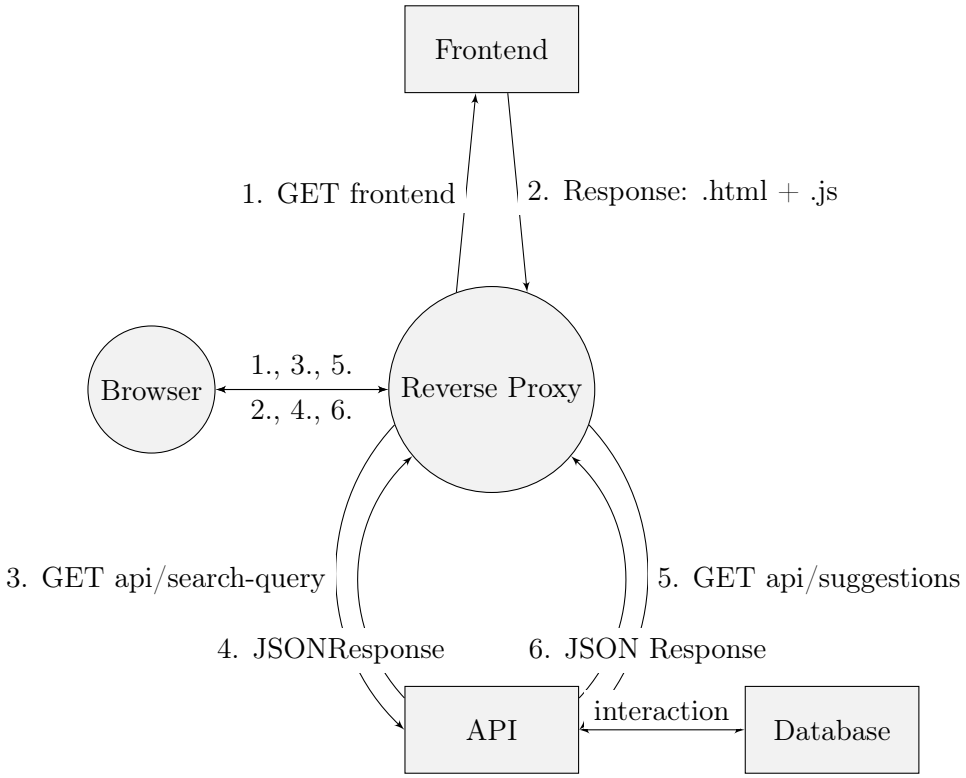

Figure 3.6: Architecture of *ConfSearch22* with Reverse Proxy

Due to the fact that we wrote a python application using Django, we also need a so called WSGI HTTP server. In simple terms, this is a module that can be contacted by the reverse proxy, this module then translates the request such that it can passed on to the python application. Thus strictly speaking, we need another piece between the reverse proxy and the Django API. For simplicity it is not shown. We used *Nginx* for setting up the reverse proxy and *Gunicorn* for setting up the WSGI HTTP server.

Chapter 4

Conclusion

Even though we faced many backlashes over the course of the project, we have been able to address all of the key issues identified in previous versions of *ConfSearch*. We have built a simple and modern search tool that is adaptable, relatively easy to maintain and comes with robust documentation.

The user gets to learn the tool with the help of suggested examples and is ready to get the full potential out of the tool after a few minutes. Thanks to more than 3'000 additional conferences, which corresponds to a 250% increase compared to the previously running version, the user can now be sure to find the best conference for his needs efficiently.

Through the modular approach used, future improvements no longer need to reinvent to build a new project from scratch but rather they can exchange single part such as the frontend or parts of the frontend.

Chapter 5

Future Work

5.1 Future Work

During the entire process of scraping the data, building an API and designing the frontend, there were many ideas that entered our thought process, but could not be implemented due to time constraints. In this chapter, I will go into some of them.

5.1.1 Mobile application

Thanks to the new architecture, it is now easy to write a mobile app for *ConfSearch22*. The app can simply send the same requests as the users browser does when running the frontend to the API. The API does not have to be changed in any way. This might come at the cost of requiring additional CSRF configurations.

5.1.2 Frontend

Scraping for data using the frontend

Right now, editing is the most tedious part in using the application. The user has to look up updated information online and insert it correctly formatted into the form. There would be the possibility that, once the user visits the edit page, a scraping algorithm starts running and suggests edits based on the scraped data. The user can then simply confirm information and no longer has to copy and paste information from different websites.

Calendar Widget

To prevent users from entering dates in a false format, instead of letting the user enter a string for the date, on could let the user select the date using a

calendar widget. However, this needs to be implemented carefully, as the current implementation supports multiple dates for deadlines and notification dates.

5.1.3 Backend

Restricted edits

Right now, the user is quite powerful. He can edit every conference in any way he wants, as long as he respects some basic rules, e.g. he cannot violate the primary key constraint on the conference name. If it turns out that users exploit this loose security policy, one has to tighten the security policies. Right now, there is already a feature that restricts edits to the acronym and name. This can be enabled by setting the *protected field* inside the conferences table of the database to true. This way each conference can be protected separately. If this is still not enough, i.e. users enter wrong dates or remove correct information, one might have to add an authentication system. Only registered users are allowed to make changes to the database.

Scalability

Depending on the number of active users, at some point the search tool might turn out to not be efficient enough anymore. I.e. there are too many requests and the server is no longer able to handle them in a timely fashion. Fortunately, there is an easy way to scale up the maximum supported server load. One can simply increase the count of workers for gunicorn. This is a parameter found in the `gunicorn.systems` file. However, in the current implementation this introduces race conditions. Right now, the database is loaded into the application when the server boots, after that it only interacts with the database when a user writes to the database. If there is only one instance of the application running, there is no race condition. However, when having two workers and the first worker writes to `A` and the second worker reads `A`, he will still read the old value of `A`. In order to resolve this issue, we need to reload all the items from the database, whenever the user wants to read them. Unfortunately, we did not think of this earlier. For the same reason, changes made on the Django Admin Page are visible only after a restart of the server.

5.1.4 User Studies

During the course of this thesis, we had to heavily rely on heuristics and intuition in order to finish the work. To get a better understanding of how people use the tool, which features take them the most time to grasp and which features they desire to have in a future release, one would have to conduct quantitative and qualitative user studies.

Bibliography

[1] "JavaServerPages," https://webtechsurvey.com/technology/javaserver-pages, accessed: 2022-06-22.

[2] M. Kuhn and R. Wattenhofer, "The layered world of scientific conferences," in *Progress in WWW Research and Development*, Y. Zhang, G. Yu, E. Bertino, and G. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 81–92.

[3] "JSP Architecture," https://www.geeksforgeeks.org/jsp-architecture/, accessed: 2022-06-22.

[4] L. Schmid, "ConfSearch20," https://pub.tik.ee.ethz.ch/students/2019-HS/BA-2019-38.pdf, accessed: 2022-06-22.

[5] E. W. Dijkstra, *On the Role of Scientific Thought*. New York, NY: Springer New York, 1982, pp. 60–66. [Online]. Available: https://doi.org/10.1007/978-1-4612-5695-3_12

[6] M. Ley, "DBLP - some lessons learned," *Proc. VLDB Endow.*, vol. 2, no. 2, pp. 1493–1500, 2009. [Online]. Available: http://www.vldb.org/pvldb/vol2/vldb09-98.pdf

[7] "Stack overflow developer survey 2022." [Online]. Available: https://survey.stackoverflow.co/2022/#section-most-popular-technologies-web-frameworks-and-technologies

[8] "Stack overflow developer survey 2021." [Online]. Available: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks

[9] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '90. New York, NY, USA: Association for Computing Machinery, 1990, p. 249–256. [Online]. Available: https://doi.org/10.1145/97243.97281

[10] J. Nielsen, "10 usability heuristics for user interface design," https://www.nngroup.com/articles/ten-usability-heuristics/, (Accessed on 07/02/2022).

Appendix A

Appendix : Searching for a paper title to find conferences

Since we trained the keyword search engine on paper titles, we decided to do very basic performance evaluation the performance of our keyword search when searching for paper titles on a very basic level. We have selected 3 conferences at random. Then, for each of these 3 conferences we took 15 of the most recent publication titles. For each of the titles, we looked at the number of the row in which the conference name matches the name of the conference at which the publication was actually made.

In the following you can see the conference name at which 15 paper titles were published. For each conference there is a histogram that shows the frequency at which a given conference appears with a certain row number in the result table, when searching for the title of a publication made at this conference.

- International Conference on Distributed Artificial Intelligence
  - The Power of Signaling and Its Intrinsic Connection to the Price of Anarchy
  - Uncertainty-Aware Low-Rank Q-Matrix Estimation for Deep Reinforcement Learning
  - SEIHAI: A Sample-Efficient Hierarchical AI for the MineRL Competition
  - BGC: Multi-agent Group Belief with Graph Clustering
  - Incomplete Distributed Constraint Optimization Problems: Model, Algorithms, and Heuristics
  - Securities Based Decision Markets
  - MARL for Traffic Signal Control in Scenarios with Different Intersection Importance
  - Safe Distributional Reinforcement Learning

  - The Positive Effect of User Faults over Agent Perception in Collaborative Settings and Its Use in Agent Design
  - Behavioral Stable Marriage Problems
  - FUN-Agent: A HUMAINE Competitor
  - Signal Instructed Coordination in Cooperative Multi-agent Reinforcement Learning
  - A Description of the Jadescript Type System
  - Combining M-MCTS and Deep Reinforcement Learning for General Game Playing
  - A Two-Step Method for Dynamics of Abstract Argumentation

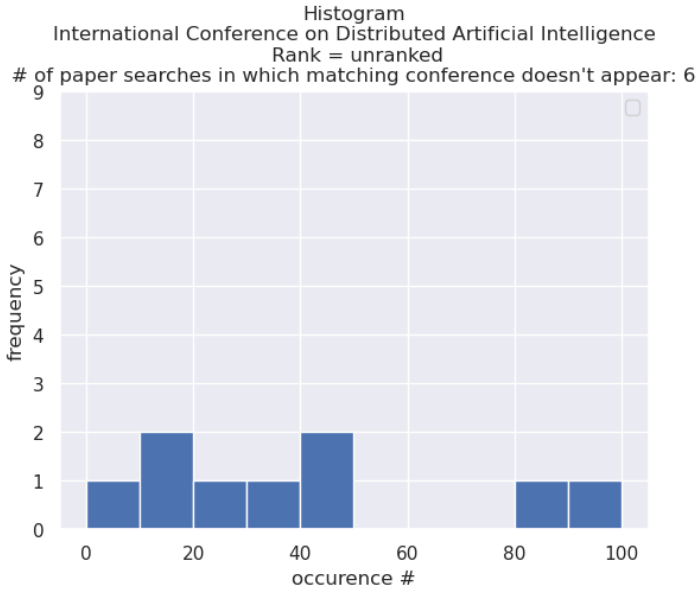

Figure A.1: Histogram : International Conference on Distributed Artificial Intelligence

- International Conference on Machine Learning
  - Global Optimization Networks
  - Spatial-channel Token Distillation of MLP-like Vision Models
  - Omni-Granular Ego-Semantic Propagation for Self-Supervised Graph Representation Learning
  - State Transition of Dendritic Spines Improves Learning of Sparse Spiking Neural Networks
  - Smoothed Adaptive Weighting for Imbalanced Semi-Supervised Learning: Improve Reliability Against Unknown Distribution Data

- QSFL: A Two-Level Uplink Communication Optimization Framework for Federated Learning
- MonePipe: Accelerating Momentum Network Training with Pipelines
- Bitwidth Heterogeneous Federated Learning with Progressive Weight Dequantization
- On Collective Robustness of Bagging Against Data Poisoning
- PAC-Net: A Model Pruning Approach to Inductive Transfer Learning
- Generic Coreset for Scalable Learning of Monotonic Kernels: Logistic Regression, Sigmoid and more
- ProGCL: Rethinking Hard Negative Mining in Graph Contrastive Learning
- Thresholded Lasso Bandit
- Evolving Curricula with Regret-Based Environment Design
- DynaMixer: A Vision MLP Architecture with Dynamic Mixing

Figure A.2: Histogram : International Conference on Machine Learning

- International Conference on Semantics, Knowledge and Grid
  - Knowledge Recommendation Based on Item Response Theory.
  - Automatic Wrong Option Generation Based on Confusion Degree.
  - Utilize Discourse Relations to Segment Document for Effective Summarization.

- Recommendation of Research Collaborator Based on Semantic Link Network.
- A Survey of Probabilistic Resource Space Model.
- The Link between Collaboration and Citation.
- Sentiment Analysis Based on Bi-LSTM Using Tone.
- JEDoDF: Judicial Event Discrimination Based on Deep Forest.
- KB-Transformer: Incorporating Knowledge into End-to-End Task-Oriented Dialog Systems.
- Learning Tibetan-Chinese Cross-Lingual Word Embeddings.
- Applications of Deep Learning Using Quaternary Hash Codes for Image Retrieval.
- A Mission-Oriented Dynamic Distribution Method of Battlefield Situation Information with Knowledge-Based Feedback Mechanism.
- Top-k Nearest Keyword Search in Public Transportation Networks.
- GRASP-Based Approach for Minimum Initial Marking Estimation in Labeled Petri Nets.
- Efforts towards Combining Graphics, Uncertainty, and Semantics: A Survey.
- Automatic Generation of Survey Paper Based on Template Tree.

Figure A.3: Histogram : International Conference on Semantics, Knowledge and Grid

The results indicate that in about 7 out of 9 cases *ConfSearch22* is able to output the conference at which the paper was presented, when searching for the paper title. In about 1 out of 4 cases, the conference is even among the top 20 best-matching conferences. It also appears that the rank has little effect on these findings. Considering that there are over 5000 conferences in the system, many of which are overlapping in research areas, the result in this random experiment are decently good. Also users are generally capable of using better keywords than a simple paper title. It is very likely that all of the 100 displayed conferences, which correspond to the 2% best matching conferences for the query, are in the right research topic and thus a good match.

Note that the sample size is very low. To get more conclusive results one needs to introduce bigger sample sizes and additional statistical measures. It is however difficult to tell whether the search result represents a good match as obviously the conference at which a paper was presented is not the only well-matching conference. In many cases there might even be better matching conferences. A user study in which researchers can give feedback on the search results would be more conclusive.