Web IR Report

Louis Kusno
2025403026

Abstract

a

2

Contents

1 Background & Motivation

At first we wanted to make a search engine for humans. Much like OSINT where you can search for people and find their social media accounts, contact information, and other personal data. We thought this would be a great idea because it would allow people to find others easily and quickly.

However, we realized that most of the data available online is from social media sites such as: LinkedIn, Facebook, Twitter, Instagram, etc. And that scraping these sites is against their terms of service. So we decided to make a search engine for professors from the top ranked universities in the world. University web pages are publicly accessible and more "permissive" to crawl than LinkedIn.

What we ended up with is **Sorgle** (profes**sor** goo**gle**), a search engine that allows users to search for professors from the high ranked universities in the world.

Sorgle

Figure 1: Sorgle Logo

2 Related Work

Early inspiration came from OSINT (Open Source Intelligence) tools used in cybersecurity to gather publicly available personal data. For example, ContactOut [1] scrapes LinkedIn profiles to provide email addresses and phone numbers.

Another notable OSINT tool is Mr.Holmes [2], which is designed to automate the process of gathering intelligence from various online sources. Mr.Holmes aggregates data from social media platforms, public records, and other web resources.

However, a big part of Mr.Holmes is the use of google dorking, which is a technique that uses advanced search operations to find specific information on the web. For example, a typical dork query to find publicly available PDF files on a specific domain might look like:

```
site:example.com filetype:pdf
```

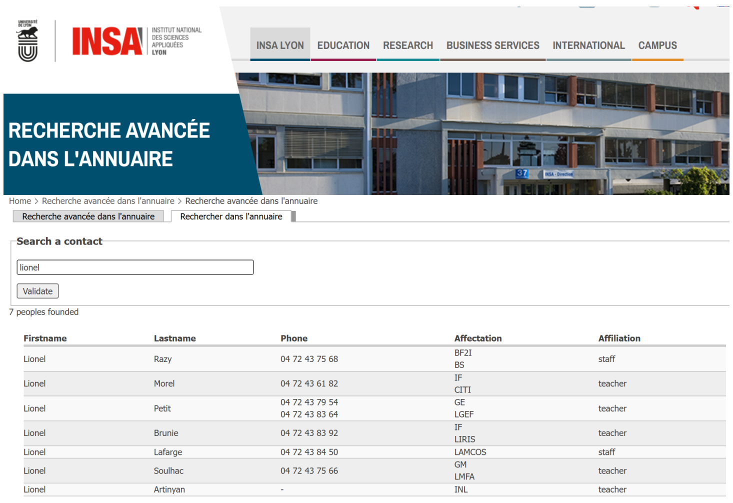
This technique uses an already existing search engine, and it wouldn't make sense to build a new search engine that relies heavily on it.

3 System Design Overview

A search engine comprises two components: a *crawler* that collects data and an *indexer/search component* that retrieves and displays results. To parallelize development, one team member built the crawler while the other implemented the search interface.

3.1 Crawler

We both started by crawling our own universities, mine being INSA Lyon in France. I used **Selenium** to send requests and **BeautifulSoup** to parse the HTML content. **Selenium** was used because INSA Lyon's website is a single page application that only allows users to search people by name, as shown in Figure 2. This means that we cannot scrape the entire list of people from the website, but we can search for them by name. To collect all professor listings, we queried every two letter combination ("aa," "ab," . . . , "zz") with a polite rate limit, saved results to CSV, and filtered out duplicates, students, and staff.


INSA
INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON
INSA LYON
EDUCATION
RESEARCH
BUSINESS SERVICES
INTERNATIONAL
CAMPUS
RECHERCHE AVANCÉE DANS L'ANNUAIRE
Home > Recherche avancée dans l'annuaire > Recherche avancée dans l'annuaire
Recherche avancée dans l'annuaire
Rechercher dans l'annuaire
Validate

Search a contact

lionel

7 peoples founded

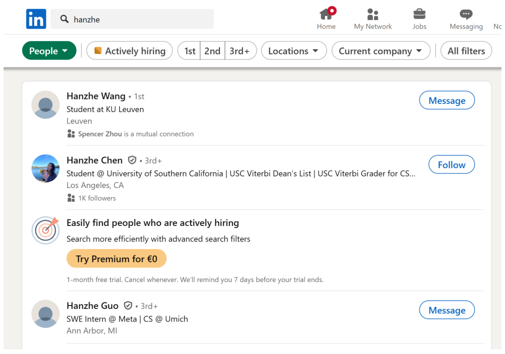
Firstname	Lastname	Phone	Affectation	Affiliation
Lionel	Razy	04 72 43 75 68	BF2I BS	staff
Lionel	Morel	04 72 43 61 82	IF CITI	teacher
Lionel	Petit	04 72 43 79 54 04 72 43 83 64	GE LGEF	teacher
Lionel	Brunie	04 72 43 83 92	IF LIRIS	teacher
Lionel	Lafarge	04 72 43 84 50	LAMCOS	staff
Lionel	Soulhac	04 72 43 75 66	GM LMFA	teacher
Lionel	Artinyan	-	INL	teacher

Figure 2: INSA Lyon's Results Page

Now that I had data from my university, I started designing the layout of the search engine. I wanted a simple and clean design.

3.2 Search Engine

I started by looking at different people search engines: LinkedIn, Instagram, Facebook, etc. I wanted to see how they display the results and what information they show. I also looked at different search engines such as Google to see how they display the results. Since our data is professor-centric, each result displays name, title, university, department, and—if available—a profile picture. We modeled Sorgle's layout on LinkedIn's search results (Figure 3 [3] vs Figure 4) and profile page (Figure 5 [3] vs. Figure 6).



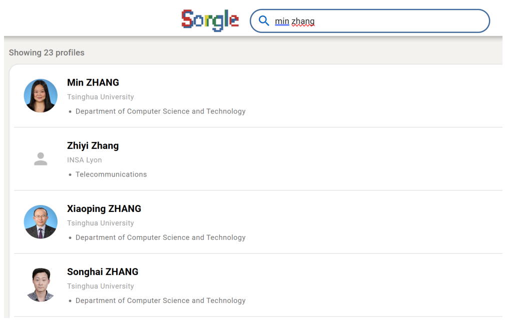
Hanzhe Wang • 1st
Student at KU Leuven
Leuven
Spencer Zhou is a mutual connection

Hanzhe Chen • 3rd+
Student @ University of Southern California | USC Viterbi Dean's List | USC Viterbi Grader for CS...
Los Angeles, CA
1K followers

Easily find people who are actively hiring
Search more efficiently with advanced search filters
1-month free trial. Cancel whenever. We'll remind you 7 days before your trial ends.

Hanzhe Guo • 3rd+
SWE Intern @ Meta | CS @ Umich
Ann Arbor, MI

Figure 3: LinkedIn Results Page [3]



Showing 23 profiles

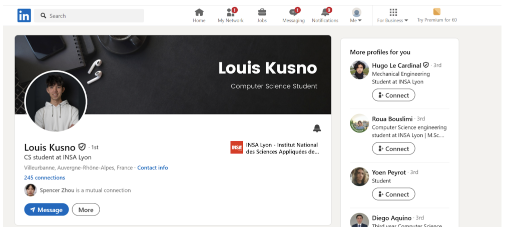
Min ZHANG
Tsinghua University
• Department of Computer Science and Technology

Zhiyi Zhang
INSA Lyon
• Telecommunications

Xiaoping ZHANG
Tsinghua University
• Department of Computer Science and Technology

Songhai ZHANG
Tsinghua University
• Department of Computer Science and Technology

Figure 4: Sorgle Results Page



Louis Kusno · 1st
CS student at INSA Lyon
Villeurbanne, Auvergne-Rhône-Alpes, France · Contact info
245 connections
Spencer Zhou is a mutual connection

INSA Lyon - Institut National des Sciences Appliquées de...

More profiles for you

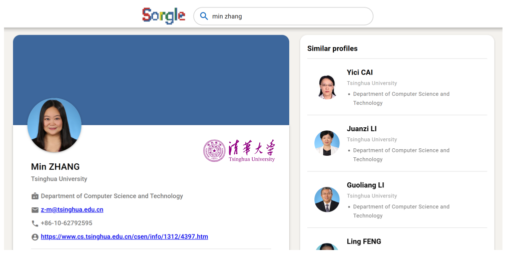
Hugo Le Cardinal · 3rd
Mechanical Engineering Student at INSA Lyon

Roua Bouslimi · 3rd
Computer Science engineering student at INSA Lyon | M.Sc....

Yoen Peyrot · 3rd
Student

Diego Aquino · 3rd
Third year Computer Science

Figure 5: LinkedIn Profile Page [3]



Min ZHANG
Tsinghua University
Department of Computer Science and Technology
z-m@tsinghua.edu.cn
+86-10-62792595
https://www.cs.tsinghua.edu.cn/csen/info/1312/4397.htm

Similar profiles

Yici CAI
Tsinghua University
• Department of Computer Science and Technology

Juanzi LI
Tsinghua University
• Department of Computer Science and Technology

Guoliang LI
Tsinghua University
• Department of Computer Science and Technology

Ling FENG

Figure 6: Sorgle Profile Page

Results pages feature a top aligned search bar with concise entries (name, university, department) and a light mode color scheme inspired by LinkedIn. Even thought I set all my apps to dark mode, I think that light mode allows for a better looking website. Profile pages display all scraped fields - name, university, department, profile picture, phone, functions, ORCID, and bio - using icons to avoid clutter. We also added a *similar profiles* sidebar to utilize empty space and improve user retention.

Finally, Sorgle's homepage mimics Google's minimalistic design to maintain simplicity and consistency.

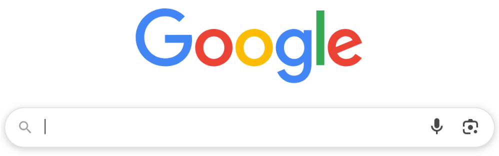


Figure 7: Google Main Page [4]

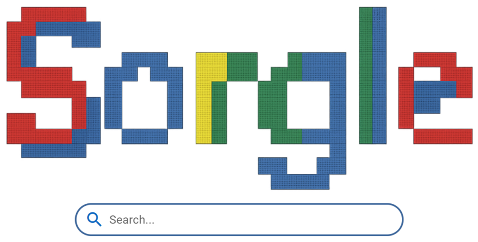


Figure 8: Sorgle Main Page

4 Technical Details

Efficient and relevant retrieval is crucial for user satisfaction. Below, we outline our search-algorithm design.

4.1 Search Algorithm

We use a fuzzy matching approach rather than exact string match. Fuzzy search handles variations and typos, which is essential for name based queries.

4.2 Fuzzy Search

We measure similarity via the Levenshtein distance, defined as the minimum number of single character edits (insertions, deletions, or substitutions) needed to transform one string into another:

- insertion: `cot` → `coat`
- deletion: `coat` → `cot`
- substitution: `cot` → `cat`

4.3 Implementation

We compute a weighted distance over three fields: name (d_{name}), university (d_{univ}), and department (d_{dept}). The combined score is

$$\text{score} = 0.6\, d_{\text{name}} \;+\; 0.25\, d_{\text{univ}} \;+\; 0.15\, d_{\text{dept}}\,.$$

Higher weights for names reflect their shorter length and greater importance. University and department fields receive lower weights to prevent overly long strings from dominating the score.

For *similar profiles*, we compute the same weighted distance between the current profile and all others, then sort by increasing score.

5 Experiments and Results

We crawled four universities—INSA Lyon, KU Leuven, Imperial College London, and Tsinghua University—and aggregated all professor profiles. Figure 9 shows the total number of professors per institution. Figure 10 lists the most common first names in the combined dataset.

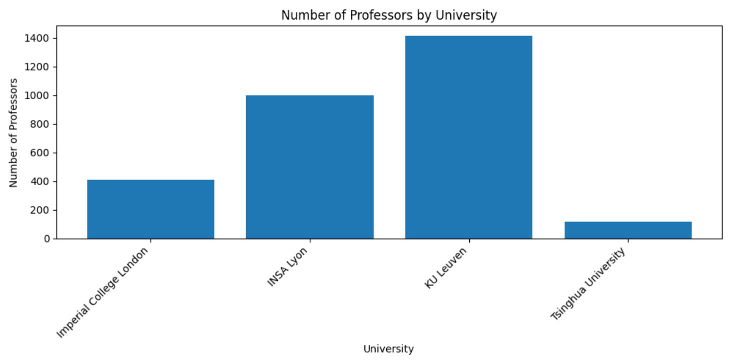


Figure 9: Number of Professors by University of the whole dataset

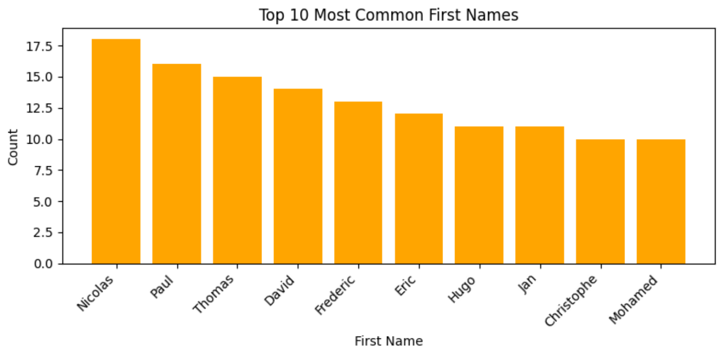


Figure 10: Most common first names among professors in the dataset

6 Conclusion & Future Work

6.1 Future Work

The biggest limitation of our search engine is the lack of data. We only have a small subset of professors from four universities, which limits the search engine's effectiveness. To improve the search engine, we need to crawl more universities and gather more data. Not only the number of professors but also more fields such as research interests, publications, and full contact information would be beneficial. The easiest way to do this, would be to use the ORCID API, which provides information about researchers, including their publications and affiliations.

Moreover, the search engine could be enhanced by implementing more advanced search features, such as filtering by research interests, publications or their role (eg: teacher, researcher, etc.). This would also have to be implemented for the similar profiles. However, I'm not sure if fuzzy search would be enough for the more advanced search features, setting up the weights for each field and the search algorithm could require some finetuning.

Another simple but effective improvement would be to add an autocomplete feature to the search bar. Moreover, the current system stores all the data in a single json file. Creating a dedicated backend with a database would allow for more efficient data management and retrieval. This would allow us to implement profile views, top searched professors, and account management features.

Lastly, we have done minimal treatment to the data we crawled. One example is the names of the departments, which are often not standardized. For example, at INSA Lyon we have ”Computer Science”, at Tsinghua University we have ”Department of Computer Science and Technology”, and at Imperial College London we have ”Department of Computing”. Standardizing these names will improve search accuracy and similar profile recommendations.

6.2 Conclusion

In conclusion, we have built a simple search engine that allows users to find professors based on their names, universities, and departments. While functional, the system's utility depends on broader data coverage, richer profile attributes, and backend improvements. We hope this work serves as a foundation for future academic search tools.

6.3 Ethical Use & Privacy

All data is sourced from publicly accessible university pages; no protected content is scraped. We exclude student records and only display faculty information. Users must refrain from misuse, such as spamming or harassment.

References

[1] ContactOut, "Contactout." https://contactout.com/. Accessed: 2024-06-10.

[2] Y. Wang and Y. Li, "Mr. Holmes." https://github.com/Lucksi/Mr.Holmes. GitHub repository, accessed June 7, 2025.

[3] LinkedIn Corporation, "LinkedIn: Professional Networking Platform." https://www.linkedin.com/, 2025. Accessed June 7, 2025.

[4] Google LLC, "Google Search." https://www.google.com/, 2025. Accessed June 7, 2025.