# Information Retrieval - Course Project 29

Nanny Search

Brian Pulfer

December 4th, 2019

# Contents

# 1 Introduction

This document represents a personal report for the Information Retrieval course's individual project. The course took place by the *Università della Svizzera Italiana* (USI) in the fall semester of the academic year 2019/20.

The goal of the project is to apply in practice the theoretical notions learned during the course. In particular, each student has to submit by the deadline (December 4th, 2019) a working prototype of an information retrieval system for a specific task and user needs.

Following, the assigned project's description, the relative implementation, the evaluation given by 3 classmates and finally the results and conclusions.

# 2 Assigned project - Nanny Search Engine

The goal of the project is to develop an information retrieval system for potential users who are interested in the search of nannies. This project is called **Nanny Search**.

## 2.1 Requirements

The final system should display a graphical interface for the user to search, browse and present results.

Also, the search of nannies has to be parameterizable by specifying:

- location
- age
- level of experience

The information should be extracted from the websites **childcare.co.uk** and **nannyjob.co.uk**.

# 3 Design

The app design is fairly simple: web pages regarding nannies are cached locally through **apache nutch**, a software that allows web crawling. An inverted link (given a word, find the documents that contains the word) is built and is used to allow fast ranking of relevant documents. **Apache solr** is a piece of software that is used to execute queries on the locally cached documents and allows for fast retrieval and good ranking.

A web application using the **Spring** framework is displayed to the user that can insert a query and adjust filters (age, experience and location). The web application uses the services provided by solr to quickly retrieve relevant documents and display them to the user with relative hyperlinks to nannies profiles. The results are displayed in the web page by the use of the framework **thymeleaf**. Also, the webpage was made responsive using the **bootstrap** framework.

The home page of the Nanny Search applications is made such that the user can easily apply the required filters (age, experience, location) with less possible mistake chance.

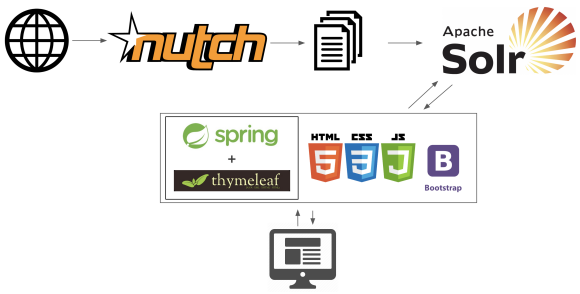


Figure 1: High level representation of the application's architecture

# 4 Implementation

## 4.1 Crawling

By giving a quick look at the two websites, patterns can be found in the URLs to nannies profiles. In particular, for **childcare.co.uk**, it is possible to consult the profile page of an user with id PROFILE_ID by following the link:

*https://www.childcare.co.uk/profile/PROFILE_ID*

while for **nannyjob.co.uk**, the respective link would be:

*https://www.nannyjob.co.uk/cv/PROFILE_ID*

By extracting every profile id present in both websites, is possible to crawl every profile of every nanny. By accessing the web page of research of nannies for **childcare** without applying any search filter, it is possible to extract the profile ids from the HTML links to the profiles pages. The same can be said for **nannyjob**.

By writing a tiny script (python), it was possible to create a **seed.txt** file with only the links to the nannies profiles. Unfortunately, for **nannyjob**, profile credentials were needed when effectuating the HTTP requests, which was a more complicated task to do and was thus abandoned. The seed.txt file was used to start the crawling with nutch.

In the end, just the nannies profiles from **childcare.co.uk** were crawled (using apache nutch), for a total of 1'000 links and **977 nannies profiles**.

## 4.2 Indexing

Indexing is done through Apache Solr using the nutch core to enable fast research. When a user submits a query from the web page, the Java backend sends HTTP request to the locally running instane of apache solr (a new instance is automatically created if no instances are running). The HTTP response is then used to display the results.

## 4.3 User interface

The user interface was developed using html5, css, javascript, thymeleaf for displaying the query results and bootstrap for making the page responsive and avaiable on every device.

A search bar allows users to insert the query, while 4 selects underneath allow for filtering **location**, **experience**, **minimum age** and **maximum age**.


Figure 2: Application's home page

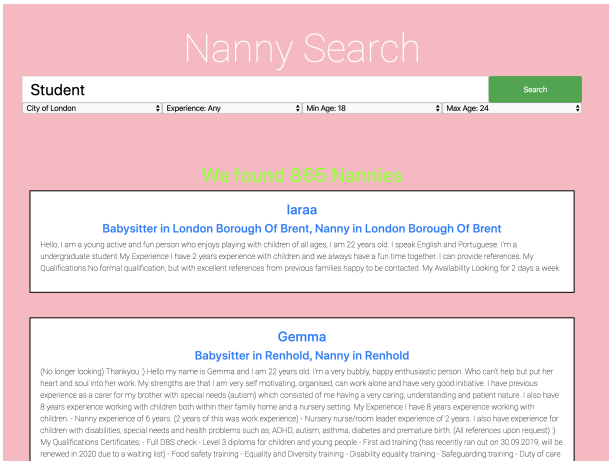


Figure 3: Example of a search result (student in london within 18 and 24)

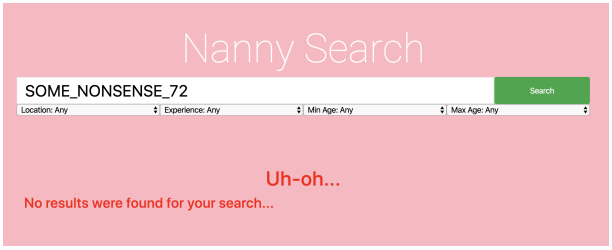


Figure 4: Example of a search fail

# 5 Evaluation & Results

The final result of the project was evaluated by 3 classmates. The goal of the user evaluation is to learn how the information retrieval system could be improved under the user experience aspects.

For this goal, 3 distinct tasks were given to the users.

## 5.1 User evaluation tasks

The tasks built for users all have a "solution" (a nanny that satisfies all requirements) and are in increasing order of difficulty.

- Find any nanny near Oxford.
- Find a male nanny that is at least 27 years old in Sutton
- Find a student nanny in London between 18 and 24 years old with at least 2 years of experience and also speaks portuguese

## 5.2 Evaluation

The user evaluation was conducted on 3 classmates. The users all had similar profiles, since they shared almost the same age (20 - 23) and similar backgrounds (computer science).

What emerged from evaluation is the fact that the **location filter** could be reworked since when asked to look, for example, for a nanny near oxford, most users didn't use the location filter specifying *Oxfordshire*. The location filter wasn't used by most of the researches, which suggests that the user should be able to type the location and the system could then adapt it internally. The fact that none of the evaluators was a british resident however, which the software is oriented to, might be the cause of the problem, since british residents are more likely to be able to map a city to the relative country.

Some users also had problems in typing the word *"portuguese"*, since most of them were not native english speakers. This suggests that the information retrieval system could be extended by adding an **auto-completion** feature.

When asked to look for a nanny with at least 2 years of experience (task 3), different users used different **levels of experience** (low, medium). This suggests that the user might be made free to type the number of years/hours of experience.
Finally, filters about the minimum and maximum age seemed to worked well instead. When asked about the graphical aspect of the page, users agreed that the colors looked good and the elements are well placed.

## 5.3 Results

The information retrieval system created successfully retrieves relevant nannies according to the query. The fact that **nannyjob.co.uk** required a registered account to access the nannies profiles made the crawling of the website more complicated and was thus abandoned, leading to a limited number of cached documents. Still, the information retrieval system relies on a rich collection of around 1'000 nannies profiles collected from **childcare.co.uk**.

The web application was implemented using apache solr to allow fast search and it was also made responsive using some web frameworks.

The user evaluation allowed to find out about design aspects that could be improved, such as the age and experience filters, or the auto-completion feature that is currently missing.