Treball final de grau

GRAU D'ENGINYERÍA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

ConnectedFood

Daniel Bautista Miralles

Director: Àlex Pardo Fernández
Created at: Departament de Matemàtica aplicada i analisi

Barcelona **June 27, 2015**

Contents

1 Introduction

Nowadays, we have fewer time to dedicate to cooking. A good diet requires an effort in preparing the plates. Fast food or pre-cooked foods are not the best option. Because of that, we need some help that allows us to save a precious time.

There are several websites to find recipes from different criteria (e.g. AllRecipes[1]), or even to know about the possible substitutes of an ingredient (e.g. MyRecipes[2]). But they are simply a large database of recipes or ingredients.

ConnectedFood is an application that lets you find ingredient substitutes and similar recipes depending on the similarity of its ingredients. It is based on an ingredients net built from several sources. It means this application adds the new feature to find ingredient alternatives to your recipes or to help you create recipe variants.

1.1 Motivation

1.1.1 Use case example

It is Sunday morning. Your friends come home for lunch. You have to cook a dish that you promised them. But you realize you don't have a couple of ingredients. There is no time to go shopping, and you decided to use ConnectedFood.

I like cooking and sometimes I have been in a similar situation, not necessarily that, but similar. This solution could be a useful one.

1.1.2 Extra personal interest

As I explain at the Development section, I decided to use MongoDB as the persistence layer for the project. I wanted to learn about it because I am interested in NoSQL databases. At work (I am a J2EE developer) we use MySQL, and everyday I can see its advantages and inconvenients. So I started to read about a document-oriented database, and I felt curiosity. In fact, I completed an online course at the MongoDB University as you can see here[3].

[1] http://allrecipes.com/
[2] http://www.myrecipes.com/
[3] https://university.mongodb.com/course_completion/d599259239874e858fcb38b0baf31f49

1.2 Goals

The original idea was to build a "social network of ingredients" and an application that exploits this network with a recommender that could learn of the users preferences. As this was very ambitious, while the idea was turning to a project, we saw that it would be better to develop a prototype instead of an entire application. Later, if there would be time, I could develop more features from there.

I spent some time trying and researching with that idea in mind. Until, finally, when it was time to begin the true development we realized that the main task was to build the ingredients network and one or two examples of exploitation. This was laborious enough.

2 Development

The development process has 2 different parts: the graph construction (and everything related to the data collection) and the web app's development. But before that, I will first explain the technologies used in this project.

2.1 Technologies

When I thought about what would be the end-user interface, I figured that an Android application would be great, rather than a web page. But then I realized that I would need a server side. For that reason, and because I would have to spend extra time learning Android, I decided to develop a simple web page. I am more familiarized with it than with the Android OS.

2.1.1 Python

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms[1].

In addition to that, Python is one of the best choices to develop using graphs and document databases. It was the first language I learned on this degree and, in my opinion, it is the most appropriate language to learn programming; not only for its simplicity and intuitiveness , but also because it forces you to write properly indenting. Moreover, Python is not a compiled language, it is interpreted; which makes easier the evolution of the project structure. For those reasons, and because I just like it, Python is the language I have chosen for the project. With it, you can have things running very quickly.

2.1.2 Python modules

pymongo

Pymongo is a Python distribution containing tools for working with MongoDB, and is the most recommended way to work with MongoDB from Python[2]. It contains the basic MongoDB shell commands and several high-level functions.

bottle

Bottle is a fast, simple and lightweight Web Server Gateway Interface micro web-framework for Python[3]. With bottle, it is very easy to develop a quick and simple web application.

networkx

NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks[4]. It allows to create graphs of several ways and export them into different formats.

requests

Requests is an HTTP library. It allows you to send HTTP/1.1 requests. You can add headers, form data, multi-part files, and parameters with simple Python dictionaries, and access the response data in the same way[5].

re

This module provides regular expression matching operations. The functions in this module let you check if a particular string matches a given regular expression[6].

lxml

The lxml XML toolkit is a Pythonic binding for the C libraries libxml2 and libxslt. It is unique in that it combines the speed and XML feature completeness of these libraries with the simplicity of a native Python API[7]. Lxml provides tools for parsing XML and HTML documents.

json

json module exposes an API to users of the standard library marshal and pickle modules[9]. It is very useful to convert JSON files into Python dictionaries.

2.1.3 HTML and CSS

HTML (HyperText Markup Language) is a markup language that web browsers use to interpret and compose text, images and other material into visual or audible web pages.

Default characteristics for every item of HTML markup are defined in the browser, and these characteristics can be altered or enhanced by the web page designer's additional use of CSS[8].

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language like HTML[10].

2.1.4 MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling[11].

The advantages of using documents are[12]:

- Documents correspond to native data types in many programming languages.

  The documents that MongoDB uses have the structure of JSON documents. This means it is very convenient to use Python because of its data type called "dictionary". These dictionaries has the same structure than a JSON document, so it is very easy to retrieve and save data in MongoDB using Python. So much so, that the mapping of a response of a MongoDB query is almost direct; even entirely direct with the help of *pymongo*.

- Embedded documents and arrays reduce need for expensive joins.

  In order to improve performance, data can be nested to avoid the creation of an extra collection or unnecessary operations to retrieve what we need. In addition, MongoDB does not guarantee the referential integrity, so with the embedded documents we can ensure the data consistency.

- Dynamic schema supports fluent polymorphism.

  As we don't have the need to define how our collections will be , we can save documents with different structure and only worry about the existence of the important fields. We always need to think about what our applications need.

2.1.5 JSON

JavaScript Object Notation is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primar-

ily to transmit data between a server and web application, as an alternative to XML[13]. As a value, you can set other documents, a list of data or simply a primary value. The list can be formed by any of the data type supported by JSON.

2.1.6 Other technologies

PyCharm

PyCharm is an IDE used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester and integration with version control systems.

Notepad++

Notepad++ is a free source code editor that supports several languages[14]. It is very useful to modify large text files thanks to the column editor and the regular expressions replacement.

Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.[15] I decide to work with this because I usually use it, not only at work, also by my own. It is very comfortable to use and I like it.

Bitbucket

Bitbucket is a hosting site for the distributed version control systems (DVCS) Git and Mercurial[16]. It offers free accounts with an unlimited number of private repositories (which can have up to five users).

SourceTree

SourceTree is a Git client that helps you manage your work and your repositories. It simplifies the use of Git , but without losing the power of this VCS.

2.2 Graph construction

The kernel of the application, the graph, is composed by names of ingredients as the nodes, and similarity relationships between those ingredients as the edges.

The most important workload of the graph construction and almost of the whole project is the data collection fase. There are four steps.

2.2.1 Ingredients list

First of all, I needed a list of ingredients, a large one. So we (Àlex and me) started to search for the internet until we found a GitHub repository with an ontology of many ingredients and more things. Here[4] you can see the ontology.

The next step was to merge all the files in the ontology to obtain one single file with all the ingredients. Then I had to purge the file with regular expressions to have only a list of ingredients. There was much extra text.

2.2.2 Substitutes search

Once I had my list of ingredients, I needed to search information about the possibles substitutes of each one. We found a web page (GourmetSleuth[5]) with many articles describing many ingredients, including their possible substitutes.

Here is where Python comes into play. In order to extract the information of the GourmetSleuth web page I wrote a script to read each article (web scraping) of each ingredient in the list. I used the result of this to create a first prototype of the graph, adding edges to it with every substitute found of each ingredient. The problem here was that not every ingredient in the list were found by the script; maybe because the page had no information about it or because it had some difference in the name. Moreover, it is also possible that the substitute found were not in the list of ingredients.

The appearance of the graph at this point was like shown in figure 1

[4] https://github.com/JoshRosen/cmps140_creative_cooking_assistant/tree/master/ontology
[5] http://www.gourmetsleuth.com/ingredients

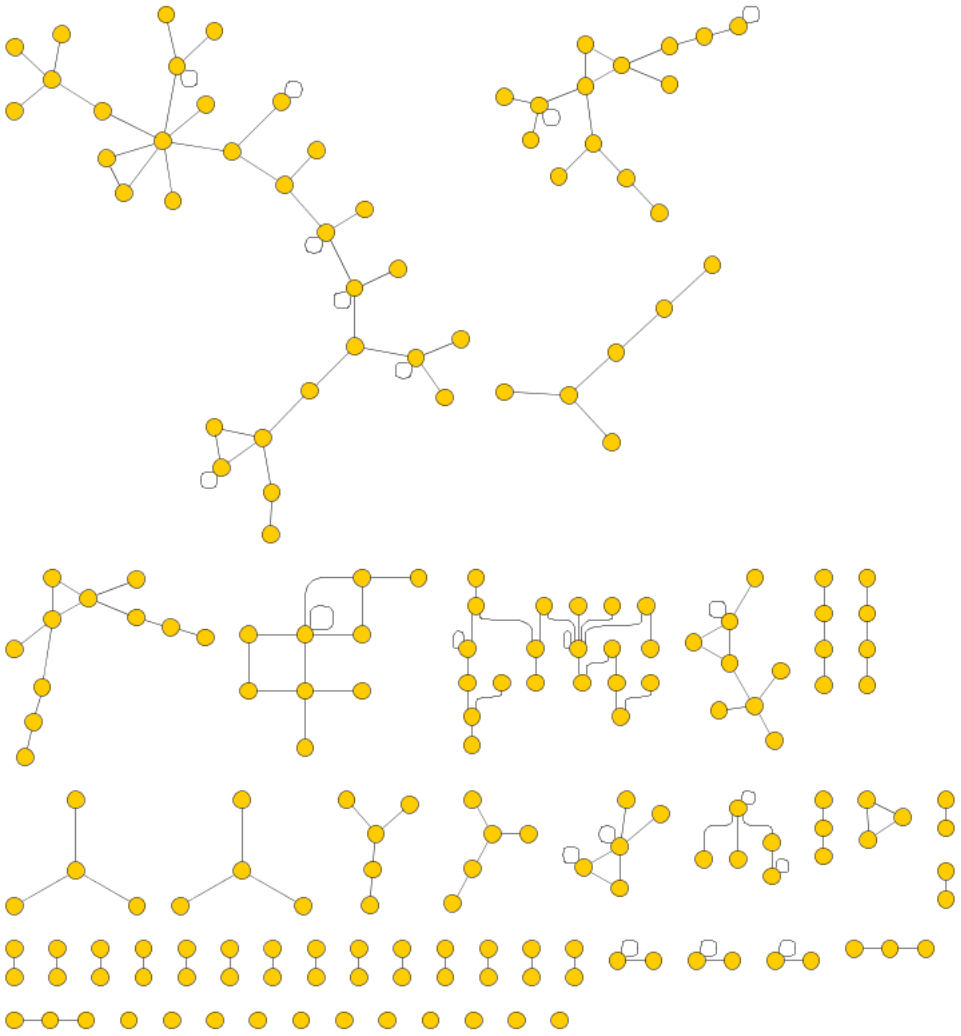

Figure 1: First prototype graph

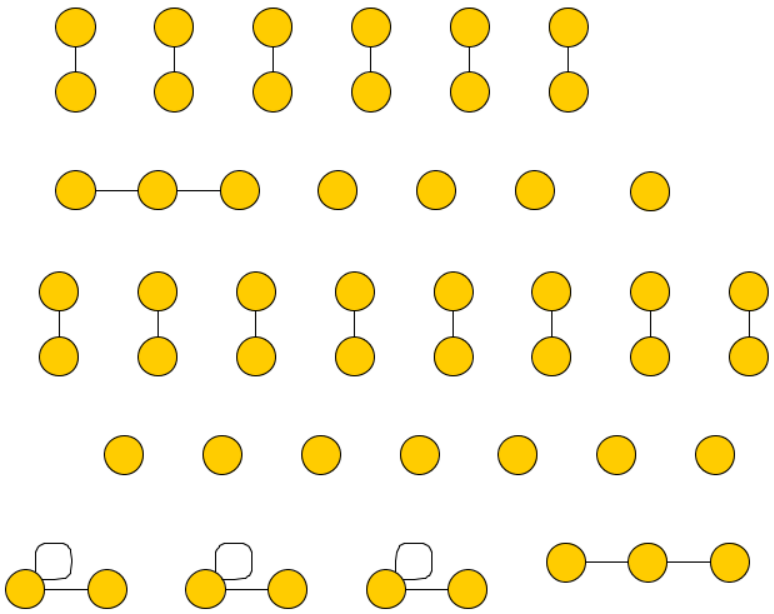

Figure 2: Islands detail

As you can see in figures 1 and 2, there are many islands where ingredients are connected to only a single other ingredient or to themselves. This means that this graph is poorly connected and there are very few nodes. So we need more information.

2.2.3 Extra information

Searching again for a web page to find substitutes to ingredients, so we can complete the prototype graph, we found a useful one: MyRecipes[6]. In this case, we have a table with several ingredients and its substitutes. Once again I wrote a script in Python to scrap the page and extract all the table rows. This time I did not create a graph, instead I saved the information in a text file.

2.2.4 Hand work and final graph

It is known that the web scraping is a technique that collect so much "garbage information". It is so difficult to extract only what you need. Because of that is very common to do a post processing with the collected data. And this is exactly what I did.

[6] http://www.myrecipes.com/how-to/ingredient-substitutions

First of all, I cleaned the text file using regular expressions, search and replace function, column editor, and reading the file to catch any strange character or word without sense. Once I had the file properly formatted (I decided to use the form "ingredient:susbstitute1,substitute2,..."), the next step was to merge the graph edges and the file in order to add all the information in the graph that was not in the file. So, one by one, I was adding ingredients until I completed the work. The result was a number of ingredients around 500, where every single ingredient had at least one substitute, and none of them was themselves.

Having the formatted and completed text file, I wrote another Python script to read the file and create a second graph version. This time, the result was more satisfying, as you can see in figure 3.

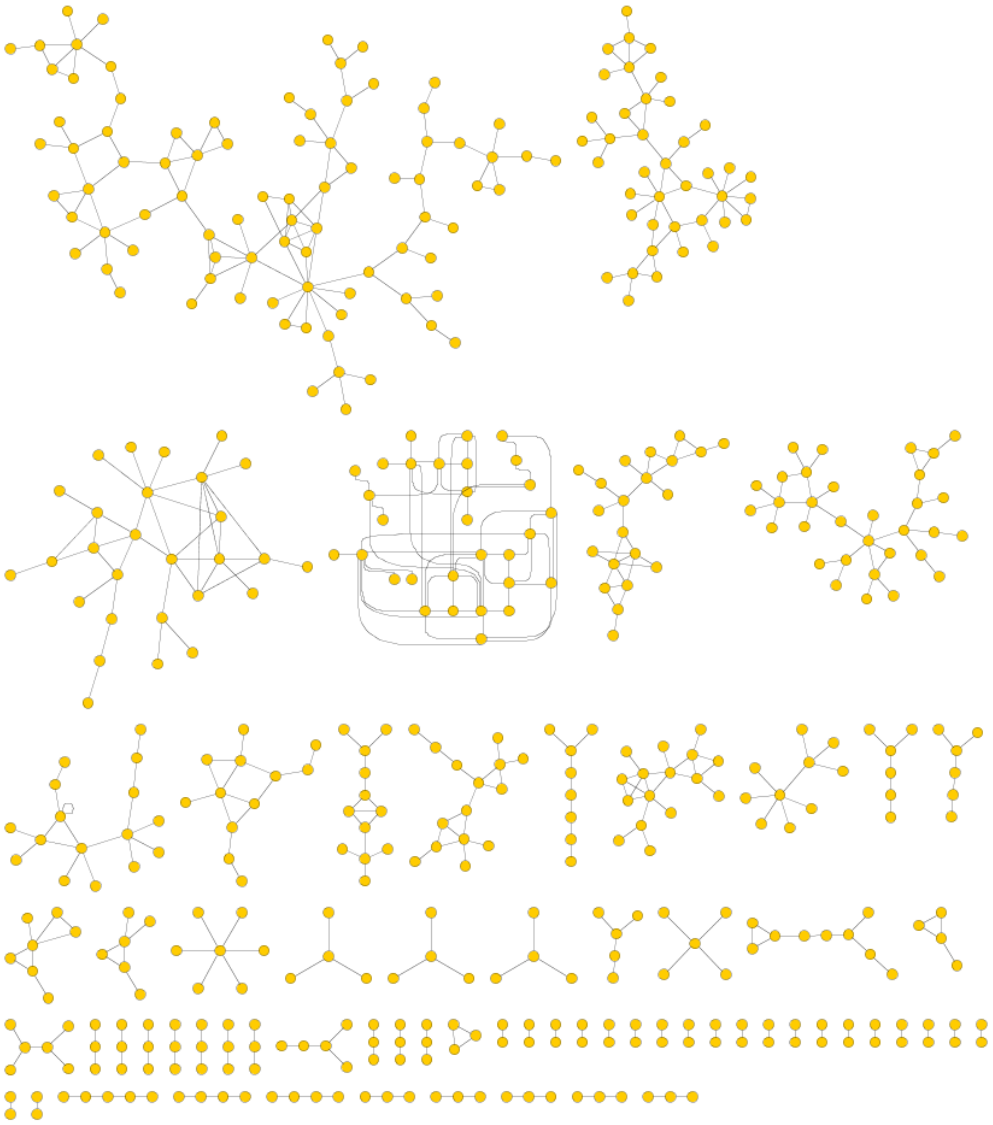

Figure 3: Enhanced graph

In this graph, we have more nodes, better connection, zero alone nodes and interesting groups as seen in figure 4.

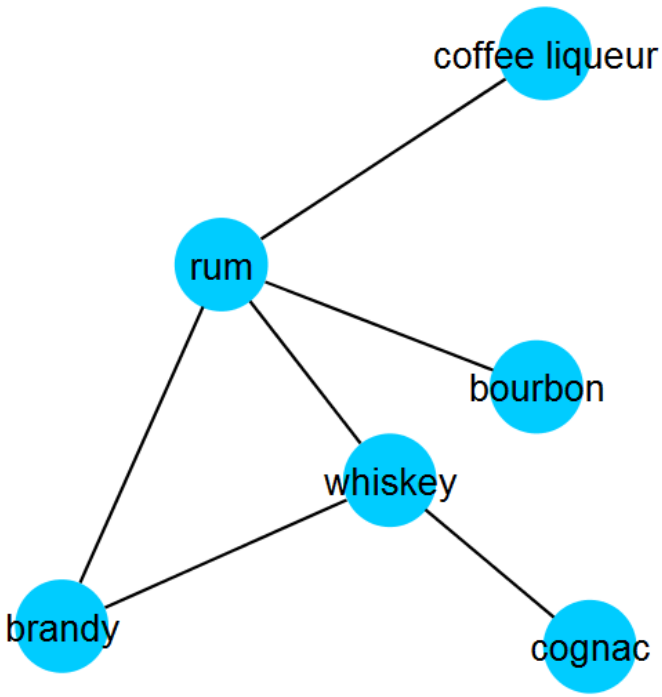

Figure 4: Enhanced graph detail

Each group represents types of food, like flours, pasta, cereals or spices, among others. It is normal that these groups are formed and not to have a fully connected graph, that does not make sense.

There is a file in the code folder of the project named "level_2_graph.py". This script reads this final graph and creates a new one weighted and more connected than the first version. Basically, for every neighbour of every node, it creates a weighted edge between the node and its neighbours with weight 1. For the neighbours of the neighbours of every node, it creates a weighted edge between the node and the neighbours of its neighbours with weight 2. For example, imagine I1" means it is similar to I1', which is already similar to I1, so I1 and I1" are less similar than I1 and I1' are, but there is a degree of similarity. This graph is not currently being used because it requires a functionality of the web app that I have not implemented, but which I will explain later.

2.3 Web page development

The next task to do was to exploit the graph. As I said earlier, I decided to do a simple web page using the explained technologies. So, let's see the features of this application.

2.3.1 Persistence

We have the graph, the main information source for this application. Now we need to store it somewhere. As you know, the persistence layer of this project is MongoDB. With a simple Python script I save the graph into the "ingredients" collection.

Throughout the application, Python connects to MongoDB thanks to pymongo to retrieve the desired data. The structure of the database schema is simple; we only have 2 collections: ingredients and recipes. The first one contains the graph (being possible to use other graphs if wanted). The second one stores the recipes. These documents have been entered manually to avoid wasting time and to test the application.

Just for the record, Àlex found an API useful to retrieve thousands of recipes (BigOven API[7]). I wrote a python file to request recipes and store it on MongoDB. It worked fine, but the recipes needed a thorough processing to match the ingredients of those recipes with my list of ingredients. So I decided not to use it.

2.3.2 Find a substitute

This feature reads the graph and shows to the user all the ingredients stored. Then the user can select one of them and search for its substitutes (figure 5). When the user clicks the Search button, then the page shows the list of substitutes of the selected ingredient. Each of them is selectable to perform a new search from it (figure 6).

This is the basic operation of exploitation of the graph. This part of the web page took practically longer than the others because it included everything related to HTML and CSS, plus bottle.

[7] http://api.bigoven.com/

Figure 5: Home page of the web app

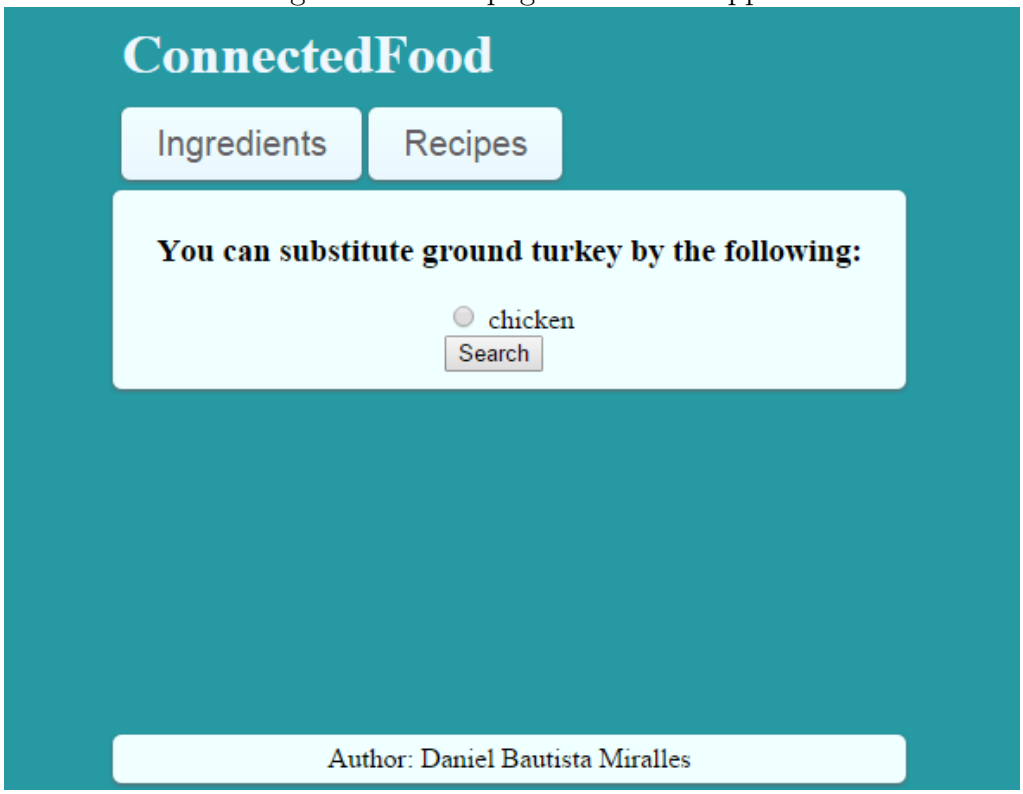

Figure 6: The result of the search

2.3.3 Find a recipe

If the user clicks on the Recipes button of the menu, the application shows a page similar to the ingredients list, but with recipes this time (figure 7). When the user selects a recipe from the drop down list, a new page is load with the information of the recipe (figure 8). Here we can see the name of the recipe, the list of its ingredients and a similar recipe suggested by the application.

The list of ingredients, in the database, is embedded in the recipe document, but every ingredient of this list exists in the graph. This way, when the application receive the request to show a recipe, it searches the ingredients of that recipe in the graph.

Figure 7: The recipes page

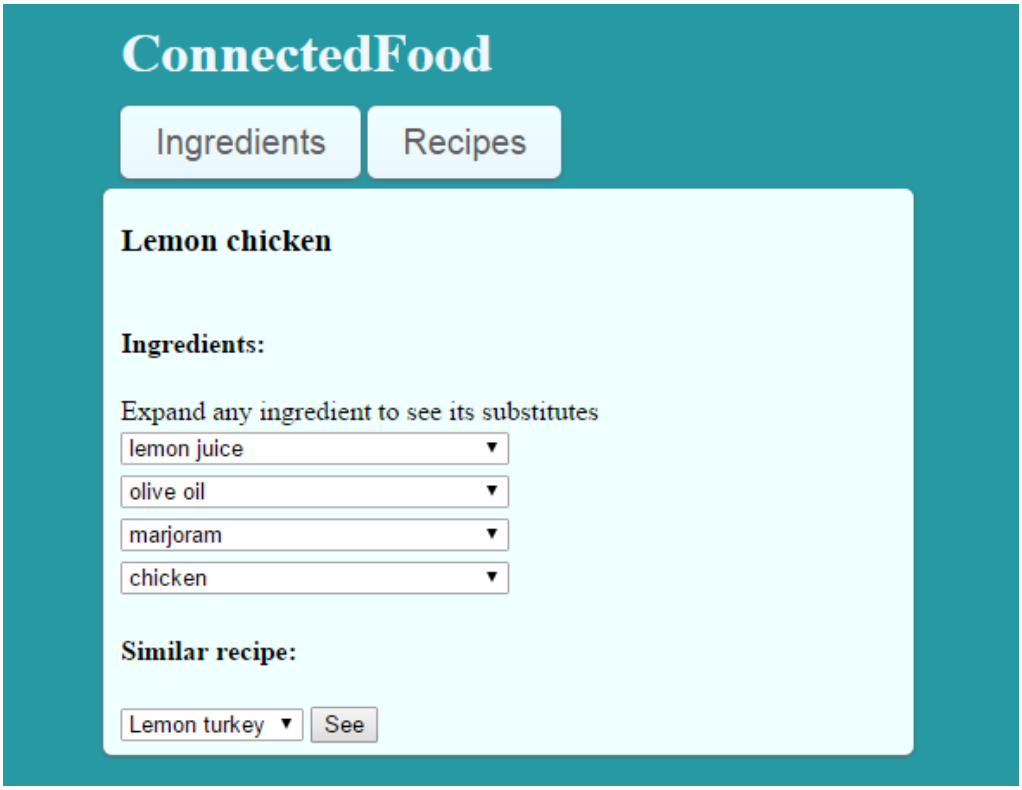

Figure 8: The information of the recipe

2.3.4 Change ingredients

Each one of the ingredients in the recipe information page is a drop down list where you can see the similar ingredients pre-calculated by the server. The way to do it is to search every ingredient from the recipe document obtained from the database in the graph, and retrieve its neighbours. The utility of this feature is to give ideas to the user of what to use instead of a concrete ingredient, if don't have it. You can see an example in figure 9.

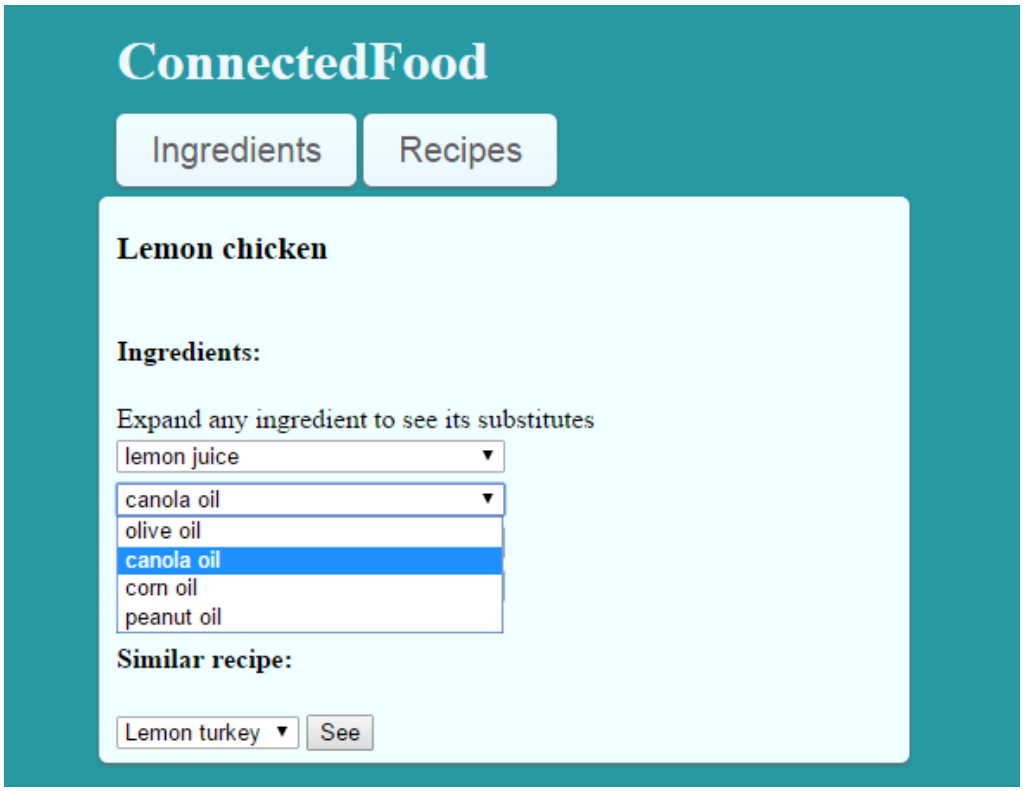

Figure 9: Ingredient change

2.3.5 Similar recipe

Similar to the above, before the application shows the recipe information page, the server calculates a similar recipe of the selected one in order to show it to the user in the Similar recipe section, as you can check in figure 10.

The algorithm that calculates the similarity between two recipes compares the ingredients of each one and gives points to the recipe depending on whether is the same ingredient, is a similar one or if is different. The recipe with the best score is selected to be shown.

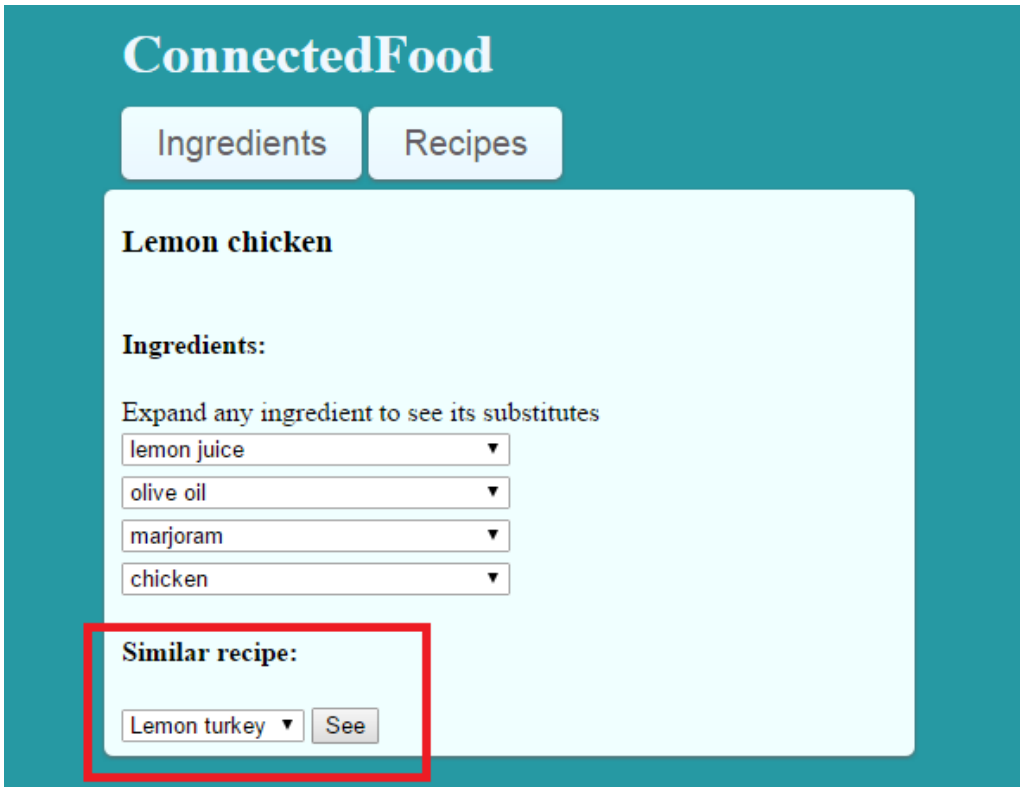

Figure 10: Similar recipe

For exmaple, consider the recipe R1 that we want to compare with the recipes R2 and R3. R1 have the ingredients I1 and I2. R2 have the ingredient I1' (which is similar to I1). And R3 have the ingredients I1, I2', and I3. If we run the algorithm, it will give 1 point to the recipe R2 and 3 points to the recipe R3, so the recipe R3 will be the similar recipe to be shown. I summarize this on the following formulas.

$$I_{R1} = I_{R2} \rightarrow 2p \tag{1}$$

$$I_{R1} \approx I_{R2} \rightarrow 1p \tag{2}$$

$$I_{R1} \neq I_{R2} \rightarrow 0p \tag{3}$$

$$Score_{R2} = \sum_{i}^{0} Score_{I_i} \tag{4}$$

In case 1 the ingredient of R2 is the same ingredient of R1, so R2 will gain 2 points. As the ingredient of R2 of case 2 is similar but not the same that the one of R1, R2 will gain only 1 point. If the ingredient of R2 is not similar neither the same of the ingredient in R1, R2 will gain no points for this comparison (case 3). Formula 4 means that the total sum of the points of every ingredient comparison of the recipe is the final score of that recipe.

Finally, and returning to the recipe page, if the user clicks on the See button, the application loads the recipe page of the similar recipe selected.

2.3.6 Error pages

If someone attempts to load a page without setting the correct parameter (e.g. load the /recipe page without selecting a recipe), the application raises a 405 http error. ConnectedFood catches this error and shows a custom page like the one in figure 11.

Figure 11: 405 error page

Also, if the user writes a path to a page that not exists in the server (e.g. /fakepage), the application raises a 404 error that catches and shows a custom page (figure 12).

Figure 12: 404 error page

There is another error page which catches the 500 errors. This error means that server has thrown an internal error. This could be probably caused by an error on the database. Is not very common. You can see an example in figure 13.

Figure 13: 500 error page

In any case, you can return to the Ingredients page or to the Recipes page by clicking the corresponding menu button.

2.4 How to run the project

2.4.1 Prerequisites

This project uses Python version 2.7.8 plus the Python modules explained previously. I highly recommend to download the Anaconda distribution[8], wich contains most of the modules. To download the rest of the modules I used the pip command.

Regarding the database, it is necessary to install MongoDB (you can download it here [9]) and configure it. If you are using Windows, you can enter here [10] and follow the instructions to set up the MongoDB environment. This project connects to "localhost", at the 27017 port.

2.4.2 Importing the database

Once you have MongoDB installed and configured as well as Python, the only thing you have to do to fill the database with the testing data source, is to run the "create_database.py" Python file. This script reads the resources from the proper project folder and creates the database and collections needed, and saves the data too.

2.4.3 Running the project

Finally, in order to run the application, just run the Python file named "app.py". Then, use a web browser to surf to the "http://localhost:8080/" url.

[8] https://store.continuum.io/cshop/anaconda/
[9] https://www.mongodb.org/downloads
[10] http://docs.mongodb.org/manual/tutorial/install-mongodb-on-windows/

3 Conclusions and future work

At the beginning I did not thought that the graph creation were to be so difficult. I wanted to focus on the user interface and the functionality of the application, but I had to sacrifice that part to provide an acceptable graph. However, today I think that the most important work of the project is, in fact, the graph. The rest of the features could be developed in the future. These functionalities could be the following.

Collaborative

The final application could be collaborative and, for example, the users could propose better relationships between ingredients or new recipes. The text file with the ingredient substitutions that I mentioned above is an easy way to change the graph before running the script to re-generate it from that source.

Smartphone or tablet app

A Rest API could be developed in order to provide the information to the apps. This kind of platforms, the apps for Android or iOS, are the most extended nowadays.

Machine learning

The application could learn to relate ingredients and to recommend recipes to the users according to their activity.

Adding information

The ingredients and the recipes could store more "metadata" like calories or quantities respectively.

Best web app

Right now, the web page is extremely simple. It could be implemented with better technologies like Django or even with another programming language more powerful and scalable, like Java.

Database performance

The additions to the ingredient information or the recipes cause the database to grow. This means that, with the current design of the database, it could lose

effectivity. In other words, it could affect to the database's scalability. So, include a better design or indexes on certain fields may improve performance.

BigOven API

Previously I mentioned the BigOven API. The application could feed the recipes collection with it or with any other useful API like this. Obviously, developing the relevant features to do so would be needed.

Search from list of ingredients

It could be another feature to let the user select one or more ingredients and perform a search of recipes according to if it has those ingredients or a similar ones.

Search excluding ingredients

Like the previous one, it could make a search from a list of ingredients but, this time, avoiding them (e.g. sugar, if the user is diabetic).

Search from properties

If the ingredients had information about their properties, the user could do a search of, for example, the recipes that are lower in fat.

2 level graph

As I explained before, there is a weighted graph which lets the user decide to see the similar ingredient depending on the weight of the edges. Unfortunately, this function is not yet implemented on the web application, because it meant a lot of changes and little time to do them. However, it could definitely be a good feature.

Quite possibly, in the future, I dedicate myself to perform any of these tasks. Or reuse any of the parts of the project.

Although there is a lot of work and lots of hours spent in things that I finally won't use in the current state of the project, the best of all is the amount of new knowledge that I have acquired and that I will be able to certainly benefit on my professional life.

4 Especial thanks

I would like to thank to my girlfriend Kat and all my family, my work companions and friends for their support.

Also thanks to Daniel Villatoro for giving me the original idea, to Oriol Pujol for trying to put me on my way in the project and to Àlex Pardo for finally doing it.

References

[1] Python Software Foundation. *The Python Tutorial* [online]. Last updated on May 23, 2015 [consulted on June 21, 2015]. Introduction. Available here: https://docs.python.org/2.7/tutorial/index.html

[2] MongoDB, Inc.. *PyMongo 3.0.2 Documentation* [online]. [consulted on June 21, 2015]. Overview. Available here: http://api.mongodb.org/python/current/

[3] Marcel Hellkamp. *Bottle: Python Web Framework* [online]. Last updated on June 21, 2015 [consulted on June 21, 2015]. Introduction. Available here: http://bottlepy.org/docs/dev/index.html

[4] NetworkX Developers. *NetworkX documentation* [online]. Last updated on September 20, 2014 [consulted on June 21, 2015]. Overview. Available here: https://networkx.github.io/documentation/latest/overview.html

[5] Python Software Foundation. *requests 2.7.0, Python HTTP for Humans* [online]. [consulted on June 21, 2015]. Introduction. Available here: https://pypi.python.org/pypi/requests

[6] Python Software Foundation. *7.2. re — Regular expression operations* [online]. Last updated on May 23, 2015 [consulted on June 21, 2015]. 7.2.1. Regular Expression Syntax. Available here: https://docs.python.org/2/library/re.html

[7] Marcin Kasperski. *Processing XML and HTML with Python* [online]. Last update on April 25, 2015 [consulted on June 21, 2015]. Introduction. Available here: http://lxml.de/

[8] Wikipedia. *HTML* [online]. Last update on June 21, 2015 [consulted on June 21, 2015]. Development. History. Available here: https://en.wikipedia.org/HTML

[9] Python Software Foundation. *18.2. json — JSON encoder and decoder* [online]. Last updated on May 23, 2015 [consulted on June 26, 2015]. Introduction. Available here: https://docs.python.org/2/library/json.html

[10] Wikipedia. *Cascading Style Sheets* [online]. Last update on June 20, 2015 [consulted on June 21, 2015]. Introduction. Available here:
https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[11] MongoDB, Inc.. *Introduction to MongoDB* [online]. [consulted on June 21, 2015]. What is MongoDB. Available here:
http://docs.mongodb.org/manual/core/introduction/

[12] MongoDB, Inc.. *Introduction to MongoDB* [online]. [consulted on June 21, 2015]. Document Database. Available here:
http://docs.mongodb.org/manual/core/introduction/

[13] Wikipedia. *JSON* [online]. Last update on June 16, 2015 [consulted on June 21, 2015]. Introduction. Available here:
https://en.wikipedia.org/wiki/JSON

[14] Don Ho. *Notepad++* [online]. [consulted on June 21, 2015]. About. Available here:
https://notepad-plus-plus.org/

[15] Bryan Goines. *git –local-branching-on-the-cheap* [online]. Last update on May 19, 2015 [consulted on June 21, 2015]. Introduction. Available here:
https://git-scm.com/

[16] Sarah Maddox, Dan Stevens. *Bitbucket Documentation* [online]. Last update on October 20, 2014 [consulted on June 21, 2015]. Bitbucket Documentation Home. Available here:
https://confluence.atlassian.com/display/BITBUCKET/