Italian Cuisine and Regional Clusters

A Big Data System from data collection to end-user recommendations

Stefano Pardini; Mattia Florio (Group 8)
Big Data Technologies, 2019-2020
MSc in Data Science - Department of Mathematics, University of Trento

Abstract: This project has the ultimate goal of creating a map of Italian regional recipes and, on through this, return suggestions to the user on regional recipes that can be prepared on the basis of provided ingredients. A pipeline has been structured, starting from the data collection and ingestion, passing through the data preparation and EDA, ending with a Web platform (and an API) that we structured as the serving layer. The aim is therefore to exploit the information provided by the collected recipes and add value, through a prediction of regionality and a personalized response.

Keywords: Italian Cuisine, Regional Recipes, Personalized Cuisine, Apache Kafka, Apache Nutch, Apache Solr, Redis, Neo4j Graph Database, Python

1. Domain Overview

1.1. Our Big Data 5 Vs

The premise that needs to be made is to analyze the context in which we operated, trying to summarize this in the terms of the Big Data 5 Vs. After doing a research, with the aim of selecting the most suitable sources for our project, we decided to collect data from 3 Italian culinary websites:

- giallozafferano.it (most popular in Italy);
- cucchiaio.it (most reliable culinary source);
- ricetteregionali.net (focused on regionality).

Now, briefly, this is our 5 Vs schema:

1. **Volume**: our 3 datasets overall amounted to ∼40 Mb, so numbers that are definitely not due to a big data context. There were no relevant numbers at the storage level, for that reason we decided to use different sources to obtain more entries (∼13500 recipes in total);
2. **Velocity**: our data were in Batch;
3. **Variety**: unstructured as well as semi-structured data;
4. **Veracity**: the sources chosen for data collection are more than reliable websites in the Italian landscape; apart from that, however, there is a non-optimal accuracy of the data, sometimes with syntactic errors, and which subsequently required a data transformation phase;
5. **Value**: the value we have tried to add is expressed in the final phase of our pipeline, that is, the possibility given to the user to provide a series of ingredients and getting back a response with a list of recipes and regional predictions as well.

Before proceeding into the core of our work, we need to focus briefly on legal aspects.

1.2. Legal Aspects

We are aware that in the scraping/crawling phase, it would have been necessary to seek permission from the copyrights' holders, in order to process their data. That's not the case. The data used is copyrighted, so subject to control over reproduction, distribution, and licensing. Given the mainly academic purpose of our work, we have not delved into this aspect of authorisation. The extracted data is neither Personal nor Sensitive, and therefore not subject to GDPR. Knowing that there is a legal basis in data processing at the time when there is copyright holder consent, formally there is not.

2. System Infrastructure

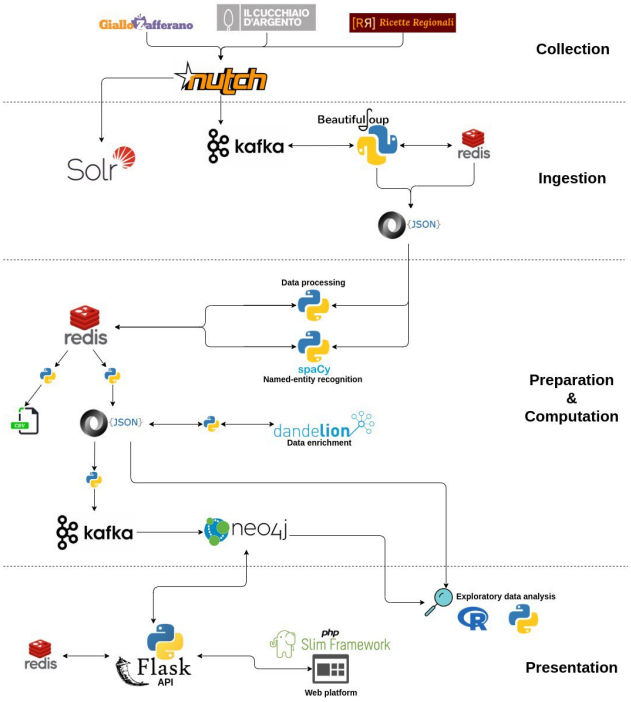

Figure 1: Big Data Pipeline

Before starting, we built up a remote shared environment with the aim to coordinate and share our work, to execute some stages of the pipeline and also hosting some services. The environment is composed of 3 virtual machines (equipped with Debian 10 Buster) hosted on a small Proxmox Virtual Environment cluster (4 physical machines).

2.1. Data Collection

Apache Nutch

First of all we needed data and so, for the crawling phase, we opted to use Apache Nutch, a stable, production ready Web crawler, highly extensible and scalable: today it is often called "the gold standard of web crawling". Its large adoption, the wide availability of plugins and the high flexibility in terms of integration with other services (e.g. Apache Kafka, etc.), are the main reasons we chose Apache Nutch for our project.

- Decoupling. Basically Apache Nutch performs the crawling activity only; it means that the scraping activity can be delegated to a separated stage of the pipeline (exactly as we did).
- Integrability. Apache Nutch comes with a wide set of plugins to communicate with external services (e.g. writes crawled data to a specific indexing backends such as Apache Solr, Apache Kafka, etc).
- Extensibility. Plugins allow anyone to extend the functionality of Nutch simply by writing their own implementation of a given interface. We wrote our own plugins (in Java) in order to extract the raw HTML content of each crawled page, also supporting its publishing on a Kafka topic.

Apache Solr

Solr is an open source full text search framework. With Solr we can search pages acquired by Nutch. We did not use all its potential; in particular, our main use cases were the monitoring of crawling activities, the full-text search for diagnostic and data exploratory purposes. Apache Nutch supports Solr out-the-box, simplifying Nutch-Solr integration.

Beautiful Soup

It is a popular Python package for parsing HTML and XML documents. All the scraping activities have been performed thanks to this library.

2.2. Data Ingestion

Source	Recipes	Ingredients
giallozafferano.it	5034	1525
cucchiaio.it	6904	21852
ricetteregionali.net	861	2198

Figure 2: Ingested Data

Apache Kafka

Apache Kafka is an open-source stream-processing software. We adopted this solution also due to its integrability with Apache Nutch. The Web crawling activity is published (indexed) by Nutch to a Kafka topic, allowing the next pipeline stage to subscribe to the stream of records in order to consume the raw data.

Although Apache Kafka is mostly for streaming data, we decided to adopt it for its performance, as well as for scalability. We used it as temporary storage system as well.

Using all 3 Debian Linux virtual machines available in our Proxmox environment, we experi-

mented with the deployment of a Kafka cluster. It was quite hard in terms of configuration, but we finally ended-up with a real fault-tolerance distributed streaming platform. We switched off the retention period deadline (the default value is 1 week), stretching the durability into persistency. It allowed us to use Kafka as an intermediate layer of persistence also to "move" the data between the various stages of the pipeline. In order to graphically explore the content of the Kafka cluster we used the Web UI "Kafdrop".

2.3. Data Preparation

Redis

Redis is an open source, in-memory data structure store, used as a database, cache and message broker. We used it mainly during the construction of our NER model for the ingredients normalization. In particular, we used the key-value store mechanism as an efficient way to keep track of the "chain" transformation of the ingredients during their normalization stages.

Machine Learning Python Libraries, Dandelion APIs, third-party tools

During the Data Preparation step, which consist of the extract, transform, load (ETL) operation to cleanse, conform, shape, transform, and catalog the data blobs and streams, we decided to develop these stages of our pipeline directly in Python, thanks to its versatility and the wide availability of libraries and tools.

This section was the most challenging part of our pipeline, as the raw data of our dataset needed some improvements:

- normalization of the extracted ingredients;
- more info about recipes' regionality;
- cleaning and uniformity of other fields, used later in the EDA part and in the final layer as well.

To solve these challenges we had, in order:

- developed a Named-Entity Recognition model, based on the available Python library: **spaCy**;
- used ***dandelion.eu*** APIs for the Entity Extraction of the regionality, using these APIs for giallozafferano.it and cucchiaio.it data (the extraction was done on the "description" node of each recipe);
- done the data cleaning and further normalization phase of other fields relevant to us (Category, Regions, Kcal, Rating, Reviews), through the libraries available in Python, specifically for the calculation of textual similarity (Natural Language Toolkit).

For summary reasons (for documentation and full explanation refer to the Wiki section on Github), here are some data that shows the evolution of this first phase of Data Transformation:

Source	Raw Data	First Cleaning	NER model processing	Post Processing	By-hand cleaning	Normalized (before merging)
giallozafferano.it	1525	null	null	1523	null	1523
cucchiaio.it	21852	19757	5346	5175	4230	4230
ricetteregionali.net	2198	2151	914	null	741	741

Figure 3: Ingredients normalization steps

Source	Regional Tags pre-Dandelion	Regional Tags post-Dandelion	Confirmed Regional Tags	New Regional Tags
giallozafferano.it	479	1286 (+807)	304	503
cucchiaio.it	666	760 (+94)	42	52

Figure 4: Dandelion.eu APIs

During the normalization phase on the ingredients, we decided to keep the 3 datasets separated, because of their specificities and differences. We therefore needed to merge the data sources that required an additional phase of data transformation:

- first cleaning through ingredients frequency and text similarity (we developed a specific metric in order to aggregate specific ingredients to the most general and similar one);
- second cleaning of singular/plural, males/females ingredients;
- manual refinement, working on semantic similarities and ingredient disambiguation;
- computation of the Levenshtein distance between each pair of ingredients, with the aim of evaluating further similarities.

So, we started from **6494** non-normalized ingredients, we got **4920** before merging the three datasets, and we arrived to **986** pure ones, thus consisting of a kind of "ontology", used for EDA, as well as for the estimation of regionality in the final layer.

In this stage of Data Preparation we worked on the ingredients. Even if one problem in our dataset was solved, another one rose up: the

problem of redundancy between Recipes. For this step, we felt it appropriate to take advantage of the peculiarities of **Neo4j**, the Graph Database we chose for the final data storing and for the Exploration Data Analysis phase.

3. Data Computation

3.1. Graph Database: Neo4j

First of all: why the choice of a Graph Database? This type of database is based on an Entity-Relationship model in which, at a computational level, nodes and relationships are evaluated in the same way; for that reason we felt that it fit our needs. With the ultimate goal of recommend the user a series of recipes, an effective and at the same time efficient query response was a feature we required. In addition, other elements leaned us towards this choice. On the one hand the clustering capabilities of nodes, based on different similarity metrics; on the other hand, a good integration with the previous steps of our pipeline, in particular the data ingestion phase carried out with Kafka. Neo4j tools we used:

- APOC library;
- Neo4j Streams - Sink: Kafka → Neo4j;
- Neo4j Graph Data Science Library, for the computation of similarity;
- Cypher query language.

As a first point, to import data into our Graph Database, it was necessary to structure an E-R model. This is the direct graph, which we decided to implement:

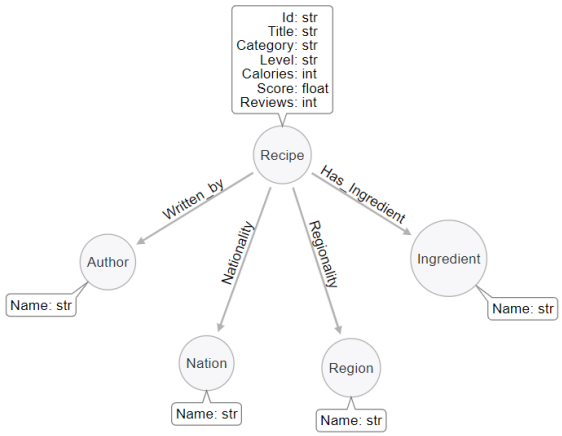

Figure 5: E-R Model in Neo4j

So, to connect Kafka to Neo4j, we used a special tool from the Neo4j streams library: the Neo4j streams sink. Through this, it is possible to ingest the data from Kafka into Neo4j, simply publishing on a topic records encoded in a JSON-friendly format (CUD) that represents Graph Entities (Nodes/Relationships). After ingesting the data, it was necessary to eliminate redundancy between recipes. The computation of the similarity between each pair of recipes, and the subsequent deletion of the repeated data, was done following a triple layer, composed in order of:

1. similarity between titles: we adopted 2 similarity measure (Levenshtein and Jaro-Winkler, both provided by the Graph Data Science Library - GDS)
2. common ingredients: Jaccard Distance Similarity (GDS);
3. matching on membership categories.

After removing repeated recipes, here are some numbers about the cleaned and **final data**:

Nodes	Relationships	Recipes	Ingredients	Regional_Tags	Regional_Recipes
13378	129673	12357	986	2264	1978

3.2. Exploratory Data Analysis

First, here is the map containing the 20 clusters corresponding to the 20 Italian regions, with their respective recipes.

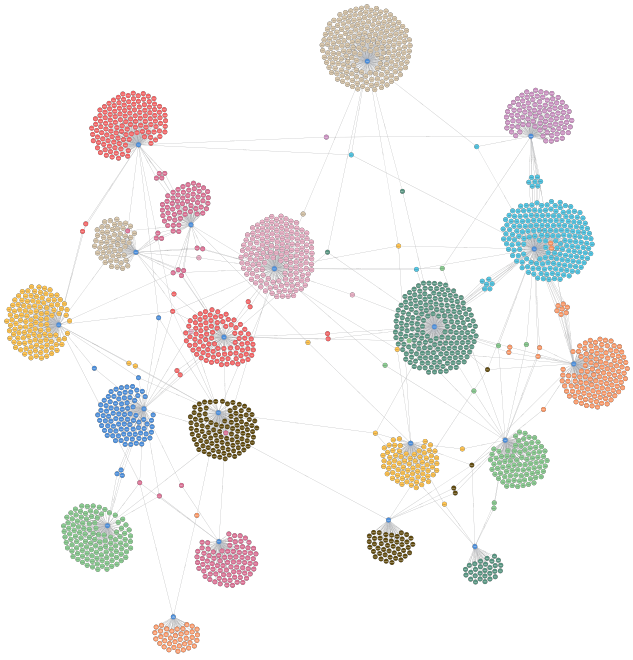

Figure 6: Regional Clusters in Neo4j - zoom Out

There are recipes that share more than one region, and correspond to the nodes scattered in the graph.

If we wanted to know all the typical recipes of Trentino-Alto Adige:

```
MATCH (r:Recipe)-[:REGIONALITY]->
(i:Region{name:"Trentino-Alto Adige"})
RETURN r,i
```

The result, in graph but also available in tabular form, is:

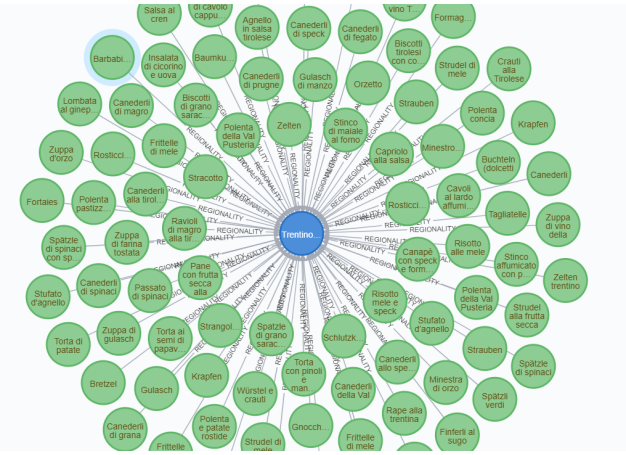

Figure 7: Regional Clusters in Neo4j - zoom In

Before moving towards the final developments of our project, we have therefore implemented a phase of Exploratory Data Analysis. We achieved this goal through **Cypher**, Neo4j's graph query language, and the integration with Data Visualization libraries on Python and R:

- Pandas, Seaborn, Plotly (Python);
- Tidyverse (R).

First of all, this is the distribution of recipes on a regional basis, where it is possible to identify some trends, namely a greater presence of recipes in the 2 extremes of the peninsula: North (Lombardia, Emilia-Romagna, Liguria and Tuscany) and South (Campania, Calabria, Sicily).

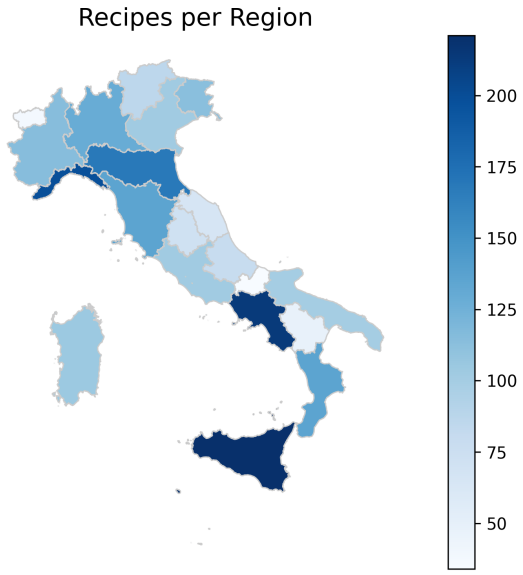

Figure 8: Recipes per Region

Secondly, we wanted to take advantage of the extracted info about the level in which a recipe was cataloged (Easy-Medium-Difficult), and associate it with the number of ingredients per recipe. This graph was produced, overlapping histograms and PDFs of the 3 separate classes.

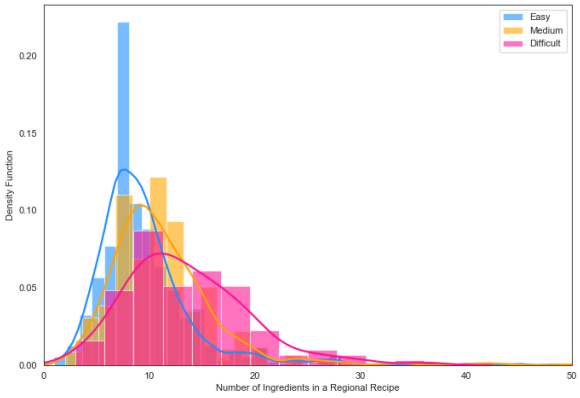

Figure 9: Probability Density Functions of the number of ingredients per recipe, between the 3 level of difficulty.

It emerges that, although the 3 classes are distinguishable, there is a substantial overlap of the 3, and therefore being able to infer what goes against common sense, that a recipe with many ingredients is necessarily difficult. This characteristic is therefore linked to other factors.

To explore the spread of the individual ingredients, and their more or less frequent use in the

Italian regional recipes, we used a wordcloud on the frequencies, always extracted through Cypher.

Figure 10: Ingredients WordCloud

We deliberately omitted ”salt” and ”pepper” because practically present in almost all recipes and not semantically relevant for this type of investigation.

Among the 3 datasets extracted, within that of gialozaffernao.it, there are some additional information very useful for a couple of other analyses.

Through the extraction of the kcal field, we wanted to look at a graphical comparison of the distributions of the kcal numeric variable for the 20 regions. There is a certain overall alignment of the caloric amount per recipe between the various groups, with a difference in internal distributions.

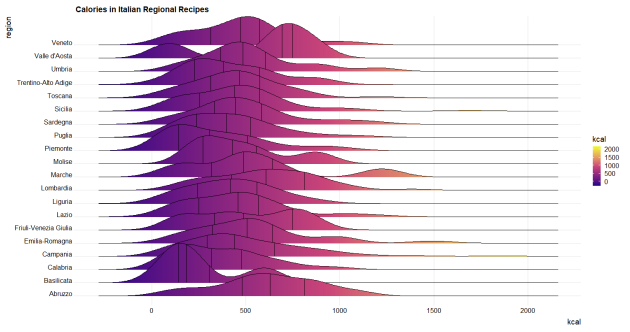

Figure 11: Kcal Distributions among the 20 regions

In the graph we also wanted to show the quartiles, the ranges of which vary greatly between the regions, thus highlighting a different intra-regional distribution.

Finally, with the goal of combining 4 variables (score, reviews, categories, number of recipes per region), we opted for a bubble chart. The numerical data extracted from Neo4j were grouped by category before and then by region. So, on the plot there are 20(regions) x 13(categories) datapoints.

The magnitude of each individual point is proportional to the number of recipes observed in that subset. The colors are distinguished according to the 13 categories of dishes. Observations of the same colour correspond to the same category in different regions. The plot was scaled on a logarithmic basis.

Figure 12: Bubble Chart: reviews vs score in each type of dish

Summarily, we can observe that ”antipasti” and ”salse” have a good approval rating but are rarely reviewed; ”dolci” associates a good rating with a high number of reviews, therefore of users’ interactions.

After this exploration phase on the available data, we asked ourselves some questions, with the central goal of adding information to the original data. The path taken, although in the limited final result that would merit a deepening, is that of a B2C tool, through the creation of a Web platform based on a set of API.

4. Final Layer: *What’s cooking in Italy?*

The project was structured not only to try to create a map, possibly exhaustive of Italian recipes, and for the identification of regional clusters, but also and above all to try to add value compared to the initial data.

So we thought to structure a Web platform capable not only of making the user interact passively - by selecting the appropriate recipe within

our database - but also actively. In this second perspective, an user, by providing a list of ingredients, has the possibility to receive as a response a series of recipes that he can cook.

To live interact with our Web platform, just connect to the link: **http://bigdata2020.byteriot.it/**.

We wanted to give relevance to the regional distinction of Italian gastronomic culture, also offering guidance on the regionality prediction, based on the ingredients provided.

The user is initially given the choice of two scenarios:

1. Empty Fridge? Look at Italian recipes divided by region or type of dish.
2. What can I cook with these ingredients?

The first option obviously does not differ much from the services already presented. It is on the second point that we wanted to work more. First, the user provides a number of ingredients at his disposal. Our platform acts schematically like this:

1. the ingredients are normalized through the NER model we have developed, and the corresponding ingredients of our ontology are returned;
2. the normalized ingredients are delivered to Neo4j, which returns a series of recipes that can be cooked with them.

Let's see some steps of the platform. Opting for more "interactive" part, this is what we see:

Figure 13: The user gives a list of messy ingredients

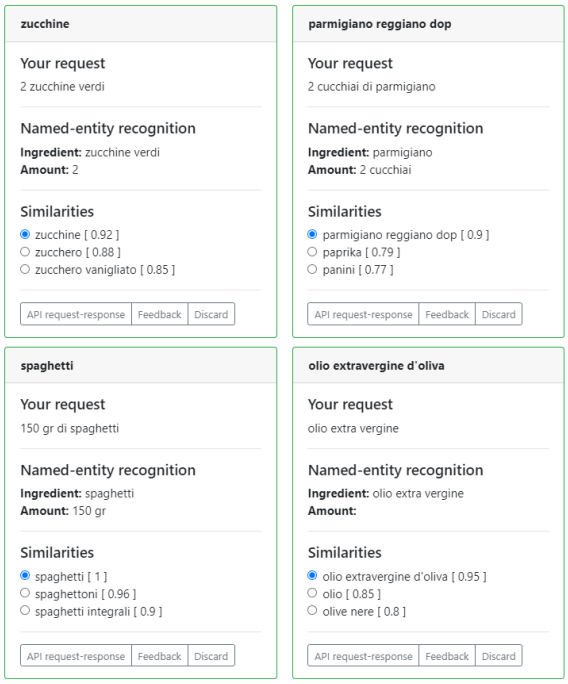

Figure 14: Our NER model response

We implemented the possibility to choice the similarity algorithm (this feature is interesting in particular from the APIs point of view). In the case we wanted clues about the regional Italian cuisine, the response is computed behind the scenes, based on a classification setting, very close to the KNN algorithm, computed through:

- a specific Cypher Query based on Jaccard similarity;
- a map of regional recipes frequencies, close to the "virtual recipe" inserted in by the user.

These are the regional recipes suggestions:

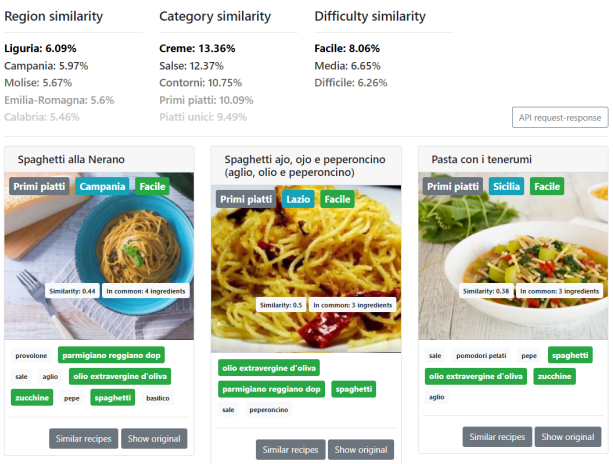

Figure 15: Regional recipes suggestions

So, we can see at the top of the page, classifications' predictions, not only about the regionality. And, if we are more curious about similar recipes linked to a choosen one, this is the response:

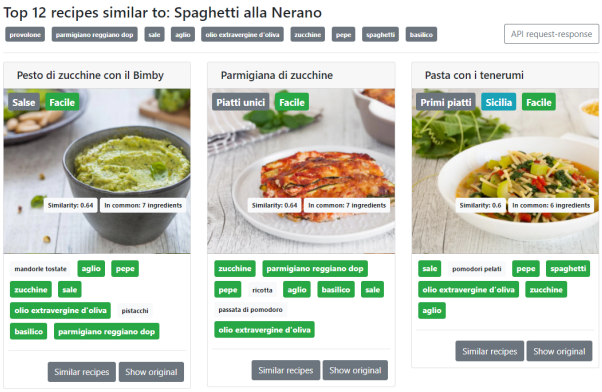

Figure 16: Similar Recipes

5. Conclusions

We tried to center the goal of the project, creating a map of Italian regional cuisine clusters, and we moved also on a parallel track, attempting to use the extracted info in a different way. We are aware that some improvements were possible. For example a possible choice of Apache Spark tools for the preparation-computation part, enabling fast iterative processing on Batch data.

Finally, we developed a lot of different ideas about the final layer, trying to think to different use cases. One of this could be the creation of a system for personalized diets, according to user's nutritional requests, rating/reviews score of the recipes. We think that what we chose is a good compromise between what was expected from the project and what were our expectations of possible, concrete and tangible, uses of it.