

UNIVERSITÉ DE TECHNOLOGIES DE TROYES

-

MASTÈRE SPÉCIALISÉ EXPERT BIG ANALYTICS ET MÉTRIQUES

Projet Neo4j : Application marmiton

Auteurs :

Jean-Baptiste BERRY
Arthur FRÉMOND
Camille ANDRÉ

Intervenant :

Jérôme BATON

31 janvier 2018



Table des matières

1	Mise en place de la base de données	2
1.1	Collecte des recettes de marmiton.org	2
1.2	Mise en place de la base de donnée orientée graph	2
2	Développement de l'application	6
2.1	Py2Neo et développement web	6
2.2	Pour aller plus loin : le machine learning	9
2.2.1	Déroulement	9
2.2.2	Modèle de deep learning	9
2.3	Modèles choisis	9
2.4	Rendu Final	11
2.4.1	Utilisation sans Machine Learning	11
2.4.2	Utilisation avec Machine Learning	12

Partie 1

Mise en place de la base de données

1.1 Collecte des recettes de marmiton.org

Pour la collecte des informations sur les recettes de cuisine via le site `marmiton.org`, nous avons dans un premier temps développé un scraper permettant de récupérer des *.json* de ces recettes. Le détail du scraper est consultable dans le notebook Jupyter¹ dédié et joint à ce mémoire.

Après récupération de toutes les recettes, nous obtenons un fichier *.json* dont chaque élément se présente comme ceci :

```
1 {"@context": "http://schema.org", "@type": "Recipe",
2  "aggregateRating": {"@type": "AggregateRating", "bestRating": 5,
3  "ratingValue": 4, "reviewCount": 13, "worstRating": 0},
4  "author": "Am\u00e9ly", "cookTime": "PT25M",
5  "datePublished": "2006-10-23T11:43:00+02:00",
6  "description": "haricot plat, tomate cerise, ail, huile d'olive",
7  "image": "https://image.afcdn.com/recipe/20130719.jpg",
8  "name": "Haricots plats (cocos) aux tomates cerises",
9  "prepTime": "PT10M",
10 "recipeIngredient": ["1 haricot plat", "15 tomate cerise", "3 gousse
11  ail", "2 cuill\u00e8re \u00e0 soupe huile d'olive"],
12 "recipeInstructions": "Laver et \u00e9cosser les haricots plats les
13  mettre de c\u00f4t\u00e9. Couper en morceaux les gousses d'ail et
14  les faire revenir dans une cuill\u00e8re \u00e0 soupe d'huile d'olive.
15  Ne pas les faire griller. Ajouter les haricots plats et mettre sur feux
16  doux (important) pendant 15 mn en remuant de temps en temps.
17  Quand les haricots sont ramollis (pas trop), ajouter les tomates
18  cerises pour le reste de la cuisson.", "recipeYield": "4 personnes",
19  "totalTime": "PT35M"}
```

On pourra noter un petit souci d'encodage qui sera réglé pendant le traintement sous python. C'est donc ce fichier *.json* que nous importerons sous Neo4j et qui définira notre base de données.

1.2 Mise en place de la base de donnée orientée graph

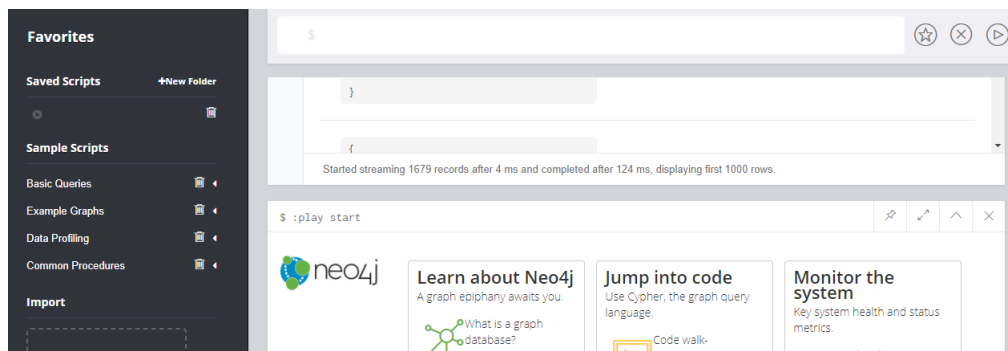
La première étape est d'installer NEO4J en local, pour y avoir accès dans votre navigateur (<http://localhost:7474/browser/>). On pourra trouver la dernière version ici : <https://neo4j.com/download/other-releases/#releases>. Dans un souci de praticité, on nommera le fichier racine `NEO4_JHOME`. Ensuite, avec la console windows, on se place dans l'emplacement du dossier et on exécute la commande `bin\neo4j console`.

1. `Scraper_Jupyter.ipynb`

Ainsi, nous avons :

```
C:\Users\BigData\NEO4J_HOME>bin\neo4j console
AVERTISSEMENT : This command does not appear to be running with administrative rights. Some commands may fail e.g.
Start/Stop
2018-01-29 21:17:33.199+0000 WARN Unknown config option: dbms.security.procedures.white_list
2018-01-29 21:17:33.287+0000 INFO ===== Neo4j 3.3.2 =====
2018-01-29 21:17:33.446+0000 INFO Starting...
2018-01-29 21:17:40.976+0000 INFO Bolt enabled on 127.0.0.1:7687.
2018-01-29 21:17:57.824+0000 INFO Started.
2018-01-29 21:18:01.198+0000 INFO Remote interface available at http://localhost:7474/
```

Et on peut donc se connecter à Neo4j à l'adresse suivante : <http://localhost:7474/>. On ne touche pas au *Bolt* par défaut ni au nom d'utilisateur et mot de passe qui sont également définis par défaut comme *neo4j*. Puis, nous arrivons sur l'interface utilisateur de neo4J :



On va à présent effectuer une modification dans *neo4j.conf* et plus précisément dans le dossier *NEO4J_HOME/conf*. De cette manière, nous pouvons utiliser APOC (Awesome Procedure on Cypher) qui nous permet d'importer nos données au format *.json* dans Neo4j :

```
# This setting constrains all 'LOAD CSV' import files to be under the 'import' direc
# allow files to be loaded from anywhere in the filesystem; this introduces possible
# 'LOAD CSV' section of the manual for details.
dbms.directories.import=import

apoc.export.file.enabled=true
apoc.import.file.enabled=true
apoc.import.file.use_neo4j_config=true
apoc.trigger.enabled=true
apoc.ttl.enabled=true
dbms.security.procedures.unrestricted=apoc.*
dbms.security.procedures.white_list=apoc.load.*
dbms.security.allow_csv_import_from_file_urls=true

# Whether requests to Neo4j are authenticated.
# To disable authentication, uncomment this line
#dbms.security.auth_enabled=false

# Enable this to be able to upgrade a store from an older version.
```

Ensuite, nous téléchargeons le *.jar* d'APOC correspondant à notre version de neo4j qui est déposé dans *NEO4J_HOME/plugins*, comme par exemple, *apoc-3.3.0.1.jar* pour la version 3.3.2 de neo4j. Ensuite, nous pouvons redémarrer neo4j et accéder à l'APOC. Nous pouvons également faire : `CALL apoc.help("apoc")` afin de s'assurer que tout s'est déroulé correctement. Puis, nous allons créer notre base de données avec la commande suivante :

Listing 1.1 – Import de la base de donnée dans neo4j.

```
1 WITH apoc.text.urlencode('marmiton.json') AS url
2 CALL apoc.load.json('file:///'+url) YIELD value AS recette
3 MERGE (r:Recette{name: recette.name}) ON CREATE
4 SET r.Instructions = recette.recipeInstructions,
```

```

5 r.Quantite = recette.recipeIngredient
6 FOREACH (value IN SPLIT(toLower(recette.description),',',' ') |
7 MERGE (e:Element{ingredient: value})
8 CREATE (e)-[:COMPOSE]->(r))

```

Une fois les noeuds et les liens implémentés dans neo4j, nous obtenons une base de données contenant toutes les recettes. Ci-dessous, un élément de la base de données *recette* tel que présenté dans neo4j.

```

{
  "name": "Haricots plats (cocos) aux tomates cerises",
  "instructions": "Laver et écosser les haricots plats les mettre de côté. Couper en morceaux les gousses d'ail et les faire revenir dans une cuillère à soupe d'huile d'olive. Ne pas les faire griller. Ajouter les haricots plats et mettre sur feux doux (important) pendant 15 mn en remuant de temps en temps. Quand les haricots sont ramollis (pas trop), ajouter les tomates cerises pour le reste de la cuisson.",
  "quantite": [
    "1 haricot plat",
    "15 tomate cerise",
    "3 gousse ail",
    "2 cuillère à soupe huile d'olive"
  ]
}

```

FIGURE 1.1 – Extrait d’une recette sous neo4j avec ses instructions et la quantité d’ingrédients nécessaires.

Dans un souci de clarté et en limitant l’affichage à 3 noeuds, nous obtenons le graph suivant :

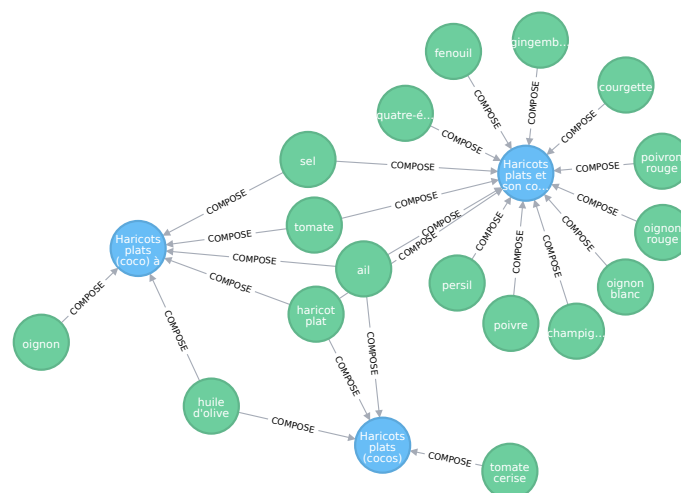


FIGURE 1.2 – Extrait de la base de données Neo4j créée.

Chaque noeud est représenté par un nom de recette, auquel est lié les noeuds des ingrédients nécessaires à sa préparation. Par ailleurs, les noeuds correspondant aux recettes sont composés de 3 propriétés :

Partie 2

Développement de l'application

2.1 Py2Neo et développement web

Pour pouvoir maintenant effectuer des requêtes cypher depuis python, nous utilisons le package `py2neo`.

```
1 from py2neo import authenticate, Graph
2 authenticate("localhost:7474", "user", "pass")
3 sgraph = Graph("http://localhost:7474/db/data/")
```

Dans un premier temps, on initialise les paramètres d'authentification pour ensuite pouvoir se connecter à la base de données orientée graph ainsi authentifiée. Pour les requêtes que nous effectuerons via l'application, nous préparons des fonctions que nous pourrons utiliser plus tard. Ainsi, nous créons une fonction `recette()`, prenant en argument une liste d'ingrédients qui sera rentrée par l'utilisateur sur l'application dans les champs mis à disposition, et qui renvoie toutes les recettes correspondant à ces ingrédients.

Pour le développement web sous python, nous avons utilisé *Flask* (framework open-source). Nous avons choisi d'ordonner notre dossier du projet de la manière suivante :

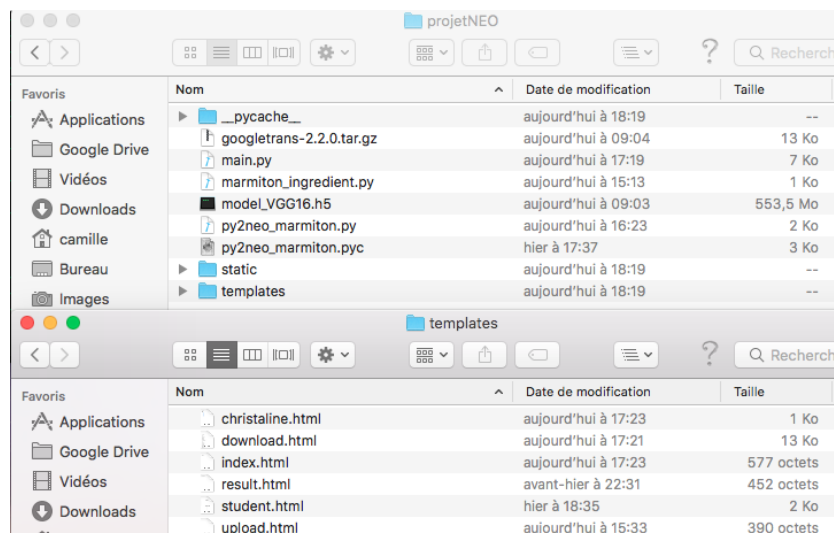


FIGURE 2.1 – Organisation des fichiers de l'application.

Il y a un dossier `static` qui contient notre fichier `css`, les photos utilisées sur le site et une police de *FontSquirrel 3D* pour le titre de l'application sur la page d'accueil. Il y a un dossier `templates` qui contient tous les fichiers `HTML` utilisés autour de ce projet et les images téléchargés dans la

page d'accueil pour le Machine Learning. Notre algorithme de ML ira chercher des photos dans ce dossier. Nous y reviendrons plus tard. Enfin, on retrouve tout en haut du projet nos scripts python, qui vont nous permettre de lancer notre application et d'exécuter nos requêtes sur la base de données neo4j.

A présent, regardons le programme principal, *main.py* :

```
7 #librairies
8 from flask import Flask, render_template, request
9 from flask_uploads import UploadSet, configure_uploads, IMAGES
10 from py2neo_marmiton import recette, etapes
11
```

On définit deux strings / html que l'on va utiliser plus tard :

```
html_str = """
<!doctype html>
<html>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}" type="text/css" charset="utf-8">
<body>
<p>
Dans votre frigo, vous avez les ingrédients suivants : {{result}}.<br/><br/> <br/>
Les plats que nous vous proposons sont :<ul style="list-style-type:square"> {0} </ul>.<br/><br/> <br/>
<p>Veuillez choisir une recette parmi les résultats :
<form method="POST" action="http://localhost:5000/index">
<select name="nom" size="1">
{puces}
</select>
<input type = "submit" value = "submit" />
</form>
</p>
</body>
</html>"""

html_str_bis = """
<!doctype html>
<html>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}" type="text/css" charset="utf-8">
<body>
<p> Vous avez choisi la recette suivante : {{camille}}.<br/><br/> <br/>
Voici les étapes de la recette : <br/> <br/>
<ul style="list-style-type:square">
{0}
</ul>
</p>
</body>
</html>"""
```

On définit notre application de la manière suivante, puis on lui indique le dossier dans lequel les photos téléchargées depuis la page d'accueil seront sauvegardées.

```
53 app = Flask(__name__)
54
55 photos = UploadSet('photos', IMAGES)
56
57 app.config['UPLOADED_PHOTOS_DEST'] = 'static/img'
58 configure_uploads(app, photos)
59
```

Puis, on définit, les différentes pages, paramétrées avec Flask. On commence par la première page :

```
60 @app.route('/')
61 def student():
62     return render_template("student.html")
63
```


Regardons le fichier HTML, `student.html` :

```

1 <html>
2 <meta charset="utf-8">|
3 <head>
4 <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}" type="text/css" charset="utf-8">
5 
6 <div id="main">
7 <h1 align="center">MARMIBON APP</h1>
8 </div>
9 </head>
10 <body>
11 
12 <form action = "http://localhost:5000/result" method = "POST">
13 <div align="center" class="recherche">
14 <h2>Ici, rentrez manuellement vos ingrédients</h2>
15 <p>ingrédient 1<input type = "text" name = "ingredient" /></p>
16 <p>ingrédient 2<input type = "text" name = "ingredient" /></p>
17 <p>ingrédient 3<input type = "text" name = "ingredient" /></p>
18 <p>ingrédient 4<input type = "text" name = "ingredient" /></p>
19 <p>ingrédient 5<input type = "text" name = "ingredient" /></p>
20 <p>ingrédient 6<input type = "text" name = "ingredient" /></p>
21 <p>ingrédient 7<input type = "text" name = "ingredient" /></p>
22 <p>ingrédient 8<input type = "text" name = "ingredient" /></p>
23 <p>ingrédient 9<input type = "text" name = "ingredient" /></p>
24 <p>ingrédient 10<input type = "text" name = "ingredient" /></p>
25 <p><input type = "submit" value = "submit" /></p>
26 </div>
27 </form>
28 <div align="right" class="nouveau">
29 <h2>Ici, téléchargez une photo de votre frigo</h2>
30 <form method="post" enctype="multipart/form-data" action="{{ url_for('upload') }}">
31 <input type="file" name="photo" />
32 <input type="submit" />
33 </form>
34 </div>
35 </body>
36 </html>
37

```

Pour ce qui est du FrontEnd, nous nous organisons comme suit. On appelle notre CSS grâce à la balise `link` tout en haut. Sa syntaxe est particulière à Flask. On insère une photo et on va définir une fiche d'action sur la page `http://localhost/5000/result` et on va récupérer les ingrédients dans une liste python. Puis, on définit une zone pour télécharger une photo d'ingrédient, l'enregistrer dans le dossier `static/img` où elle pourra être utilisée par un algorithme de Machine Learning pour être labélisée (par exemple, on télécharge une photo de tomate et l'algorithme reconnaît que c'est une tomate). Nous pouvons voir ci-dessous la définition de la page `http://localhost/5000/result` :

```

7 @app.route('/result', methods = ['POST', 'GET'])
8 def result():
9     if request.method == 'POST':
10         result = request.form#on récupère le résultat de la form dans student.html.
11         ListeIngredient = result.getlist('ingredient')#on stocke le résultat dans une liste que l'on nomme ListeIngredient.
12         myList = []
13         for ingredient in ListeIngredient:
14             if len(ingredient) > 0:
15                 myList.append(ingredient)
16                 string_list = " ".join(map(str, myList))#on fait cela pour obtenir quelque chose de plus propre pour l'affichage.
17                 stock = recette(myList)#on applique la fonction recette que l'on a défini dans py2neo_marmiton.py.
18                 li = "<li>{0}</li>"#pour l'affichage des recettes sous forme d'énumération.
19                 select = "<option>{0}</option>"#pour l'affichage des recettes dans une barre déroulante.
20                 subitems = [li.format(a.encode('utf-8')) for a in stock]#pour l'affichage des recettes sous forme d'énumération.
21                 subitems_bis = [select.format(a.encode('utf-8')) for a in stock]#pour l'affichage des recettes dans une barre déroulante.
22                 carac = " ".join(subitems_bis)#on concatène la liste dans une chaîne de caractères
23                 neo = html_str.format(" ".join(subitems), puces = carac)#on substitue ces valeurs dans la string html défini au début.
24                 #"".join(subitems_bis)
25                 #neo = neo.format(" ".join(subitems_bis))
26                 neo = neo.replace("{", "{{")
27                 neo = neo.replace("}", "}}")
28                 #print(neo)
29                 f = open("templates/christaline.html", "w")#on définit un nouveau fichier html, christaline.html
30                 f.write(neo)#on y enregistre la nouvelle string actualisée.
31                 f.close()#on ferme le fichier
32                 #print html_str.format(" ".join(subitems))
33                 return render_template("christaline.html", result = string_list)#on restitue le résultat et on remplace result dans le html par
34                                     #string_list.
35

```

Le script sur Flask pour la page d'index est le suivant (Voir les commentaires du script pour plus de détails) :

```

1 @app.route('/index', methods = ['POST', 'GET'])
2 def index():
3     if request.method == 'POST':
4         result_new = request.form['nom']#on récupère le nom de la recette dans la barre déroulante.
5         instru = etapes(result_new.encode('utf-8'))#on applique la fonction etapes de py2neo_marmiton.py à la recette sélectionnée.
6         li = "<li>{0}</li>"#pour l'affichage des étapes sous forme d'énumération.
7         subitems = [li.format(a.encode('utf-8')) for a in instru]#pour l'affichage des étapes sous forme d'énumération.
8         neo = html_str_bis.format(" ".join(subitems))#on substitue ces valeurs dans la string html définie au début.
9         neo = neo.replace("{", "{{")
10        neo = neo.replace("}", "}}")
11        f = open("templates/index.html", "w")#on définit un nouveau fichier html, christaline.html
12        f.write(neo)#on y enregistre la nouvelle string actualisée.
13        f.close()#on ferme le fichier
14        return render_template("index.html", camille = result_new)#on restitue le résultat et on remplace camille dans le html par
15                                #result_new
16

```

2.2 Pour aller plus loin : le machine learning

2.2.1 Déroulement

De manière avant-gardiste, nous avons souhaité intégrer dans notre application du Machine Learning notamment à travers la mise en place d'un réseau de neurones. Le but est de permettre à l'utilisateur de charger une photo de l'intérieur de son réfrigérateur pour que l'application lui propose des recettes automatiquement.

Pour se faire, nous devons décomposer le processus en plusieurs étapes clés en termes de Machine Learning :

- Localisation des différents ingrédients présents dans l'image de l'intérieur du réfrigérateur.
- Découpage de l'image en multiples images composées chacune d'un unique ingrédient.
- Analyse des images et reconnaissance (labélisation) des ingrédients.

Une fois ces étapes faites, les ingrédients peuvent être regroupés dans une liste et transférés sous forme d'une requête à la base de données Neo4j. Cette étape de requêtage est identique à celle implémentée lorsque l'utilisateur spécifie ses ingrédients manuellement dans les champs laissés à sa disposition.

2.2.2 Modèle de deep learning

Les modèles les plus performants en termes de reconnaissance et traitement d'images sont les modèles à convolution (CNN). En effet, les réseaux de neurones convolutifs (CNN) sont principalement utilisés pour classer des images. Ils comportent deux parties bien distinctes :

- Une première étape composée de couches convolutionnelles
- Une deuxième étape composée de couches connectées ordinaires

La première partie d'un CNN, partie convolutive à proprement parler, fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de filtres créant de nouvelles images appelées cartes de convolutions. Certains filtres intermédiaires réduisent la résolution de l'image.

Puis, les cartes de convolutions sont mises à plat et concaténées, constituant l'entrée de la 2ème étape. La deuxième partie est simplement faite de plusieurs couches de perceptrons connectées, exactement comme un réseau de neurones ordinaires. La sortie est une dernière couche comportant un neurone par catégorie pour classer l'image.

Il existe de multiples modèles CNN bâtis sur ce principe. Chacun possède une architecture différente par le nombre de filtres, leurs tailles, etc. Chaque CNN est conçu dans un but bien précis.

2.3 Modèles choisis

Il est possible sur les bases décrites ci-dessus d'entraîner son propre modèle CNN. Cependant plusieurs problèmes se posent :

- Une base de données importante doit être utilisée pour entraîner ces réseaux de neurones.
- Il faut une base de données par ingrédient (ou classe) que le modèle doit apprendre.
- Notre problématique comprend un nombre de classes égal au nombre d'ingrédients que notre modèle est censé connaître, soit un nombre très important.

Des débuts de modèles sont actuellement développés sur la base de la reconnaissance alimentaire. Cependant, rien n'est encore abouti pour le moment. Développer un modèle « from scratch » serait un projet à part entière vu la complexité des éléments à reconnaître. Nous avons donc décidé d'utiliser des modèles pré-entraînés disponibles sur keras, une librairie de deep learning sur Python fonctionnant avec Tensorflow. Ces modèles ont été développés pour la compétition image-net1000 challenge, où les modèles sont testés sur la reconnaissance de 1000 classes différentes. Il est possible de récupérer les poids des modèles entraînés et de les utiliser pour ses propres besoins.

Dans notre cas, nous avons choisi et utilisé le modèle VGG16. Une explication de son implémentation dans notre projet est disponible sous forme d'un fichier¹ jupyter dédié, joint à ce mémoire. Nous avons dû ajouter un traducteur en sortie du réseau de neurones pour traduire la prédiction en anglais du label, en français. En effet, notre base de données Neo4j du site Marminton est en français et toute requête avec des termes anglais aboutit à des résultats nuls. Une étape est manquante : celle de la localisation des différents ingrédients dans la photo de l'intérieur du réfrigérateur. Après recherches, il existe un modèle performant de localisation d'éléments dans une image, le modèle YOLO (You Only Look Once). Cependant ce modèle, ne fonctionne que sur de la vidéo. A nouveau, des modèles sont en cours de développement mais aucune solution n'est utilisable pour le moment.

Face à ces limites, nous avons donc décidé de limiter l'implémentation du deep learning à la reconnaissance d'un unique ingrédient présent sur une seule image, et non sur une image de réfrigérateur composée de multiples ingrédients.

Pour résumer, l'utilisateur peut uploader sur le site une image ayant un unique ingrédient puis l'application est capable de reconnaître l'ingrédient présent sur la photo et enfin propose à l'utilisateur les recettes possibles avec cet ingrédient.

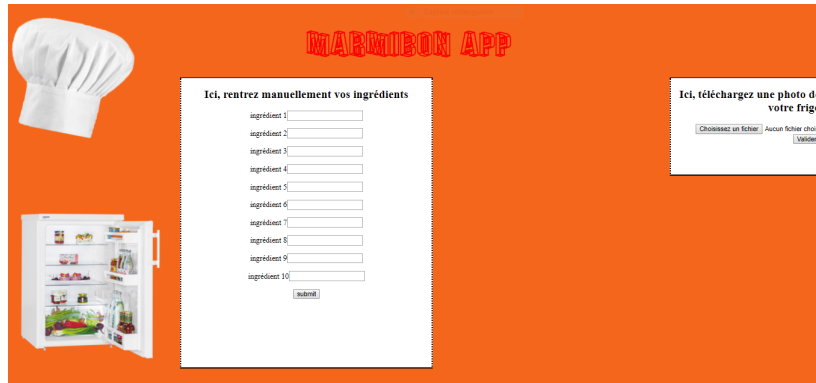
1. Deep_Learning_Jupyter.ipynb

2.4 Rendu Final

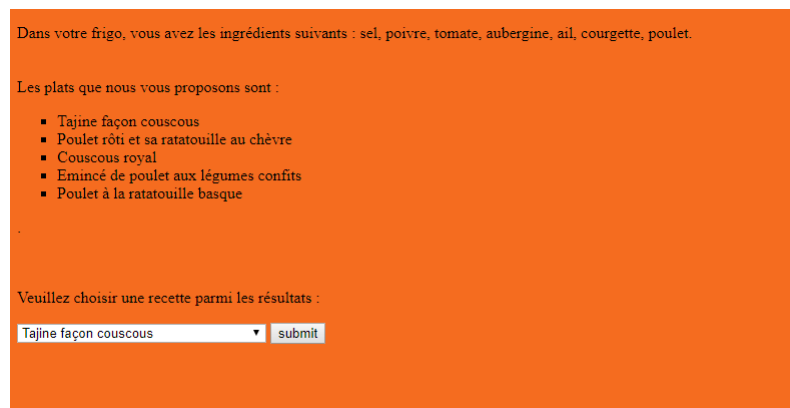
2.4.1 Utilisation sans Machine Learning

Maintenant que l'application est prête, nous obtenons les résultats suivants lorsque nous la lançons.

Pour la page d'accueil :



Par exemple, on rentre les ingrédients sel, poivre, tomate, aubergine, ail, courgette, poulet et on obtient pour la page de résultats :



Puis, on sélectionne la recette que l'on souhaite suivre. Par exemple, on choisit "Couscous royal" et la page index nous renvoie les étapes de préparation associées à cette recette :

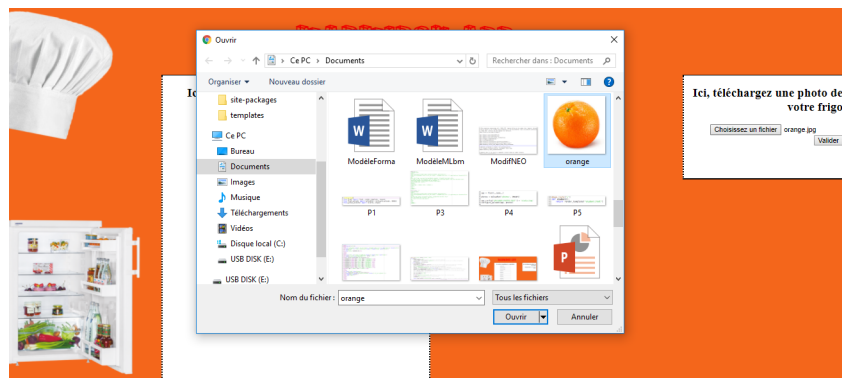
Vous avez choisi la recette suivante : Couscous royal.

Voici les étapes de la recette :

- 1) La veille, mettre les pois chiche à tremper dans de l'eau froide salée
- 2) Mélanger la semoule avec les raisins secs
- 3) Saler et mouiller avec 7 cl d'huile d'olive et 1 l d'eau bouillante
- 4) Mélanger à la fourchette et laisser reposer 15 minutes
- 5) Gratter à nouveau avec une fourchette pour bien décoller les grains
- 6) Couper le poulet et l'agneau en morceaux
- 7) Peler les navets et les carottes, les couper en morceaux
- 8) Couper tous les autres légumes en gros morceaux
- 9) Hachez les oignons
- 10) Faire chauffer la partie basse du couscoussier avec un peu d'huile
- 11) Y faire revenir les morceaux de viande de toutes part
- 12) Ajouter les oignons et laisser suer quelques minutes en remuant
- 13) Saler, poivrer
- 14) Ajouter les carottes, les navets, les pois chiches et les fonds d'artichaut
- 15) Mouiller d'eau froide jusqu'en haut du couscoussier, ajouter les épices, 2 cuillères à soupe de harissa et 1 poignée de coriandre hachée
- 16) Poser le panier vapeur dessus, couvrir le fond d'un linge propre et y déposer la semoule
- 17) Fermer
- 18) Porter le tout à ébullition, puis baisser un peu le feu et laisser cuire 30 minutes
- 19) Retirer le panier de semoule, égrener à la fourchette et rajouter un peu d'huile
- 20) Ajouter les fèves, les tomates, les courgettes et les aubergines dans le couscoussier
- 21) Remettre le panier vapeur et poursuivre la cuisson 15 minutes
- 22) Faire griller les merguez à la poêle, sans matière grasse, pendant 10 minutes
- 23) Retirer la semoule, la mettre dans un grand plat
- 24) Egoutter la viande et les légumes et les disposer dans un autre plat avec les merguez, parsemer de coriandre hachée
- 25) Enfin, verser le bouillon de cuisson dans un saladier
- 26) Présenter un pot de harissa à côté.

2.4.2 Utilisation avec Machine Learning

L'utilisateur peut charger une image depuis son bureau grâce à la fenêtre prévue à cet effet à droite de la page d'accueil.



Ici, nous décidons d'uploader une image d'orange. Après avoir téléchargé l'image, la machine détecte l'ingrédient présent et la labélise. Dans notre exemple, le programme détecte bien une orange, et nous propose ainsi toutes les recettes utilisant des oranges.

Le nom de votre photo est : orange_5.jpg

Le nom de votre ingrédient est : orange

Les plats que nous vous proposons sont :

- Risotto aux crevettes et aux oranges
- Magret de canard au café
- Rôti de biche à la sauce aux fruits secs
- Simple gâsier en civet de Noël
- Galette de sarrasin Terre et Mer
- Gambas au safran et à l'orange
- Poulet à l'orange caramélisé
- Tarte Tatin étoilée endives orangées chèvre
- Pilaf de volaille à l'indonésienne
- Rôti de porc frotté aux pignons
- Encocado de crevettes (Équateur)
- Paccheri en sauce veau agrumes
- Rôti de porc aux fruits secs
- Saute de porc aux agrumes
- Travers de porc à la californienne
- Magret de canard épice à l'orange
- Filet de lieu noir en papillote de poireaux aux agrumes
- Cuisses de poulet à l'orange et à la menthe
- Risotto à l'orange
- Daube provençale
- Ragout de porc et de moules (Catalogne)
- Filet mignon rôti aux épices du sud
- Poulet aux agrumes facile
- Canard à l'orange et au miel
- Médallions de veau au citron
- Porc aux agrumes
- Dorade à l'orange
- Poulet entier laqué

De la même manière que dans la partie précédente, nous pouvons sélectionner la recette d'intérêt dans le menu déroulant.

- Tajine de mouton à l'orange
- Rôti de lotte et ses légumes à l'orange
- Brochettes de Canard laqué
- Canard à l'orange à ma façon
- Capuccino de potiron aux épices d'hiver
- Filet mignon sauce soja et carottes glacées
- FILET DE PERCHE AUX AGRUMES
- Croquants d'avocats aux agrumes et crevettes
- Riz au jus d'orange
- Tarte aux légumes façon cheesecake
- Tajine de saumon flambé à l'armagnac
- Brochettes de dinde et de porc
- Rouelle de porc au miel et noix de coco
- Osso-bucco aux saveurs asiatiques façon funambuline
- Daube de boeuf au vin blanc
- Poulet sauté aux fruits

Veuillez choisir une recette parmi les résultats :

Risotto aux crevettes et aux oranges ▼ submit

Et enfin, l'application nous montre les étapes à suivre pour la recette sélectionnée.

Vous avez choisi la recette suivante : Rôti de porc aux fruits secs.

Voici les étapes de la recette :

- 1) Préparer une marinade à partir du jus du citron avec quelques zestes et le jus de l'orange
- 2) Y ajouter les épices et les crevettes
- 3) Laisser reposer environ une heure
- 4) Dans une poêle, faire revenir l'oignon coupe finement et l'ail
- 5) Y ajouter les tomates, coupées en petits cubes puis le lait de coco
- 6) Laisser chauffer 5 min à feu vif
- 7) Y ajouter les crevettes et la marinade, ajuster l'assaisonnement
- 8) Laisser cuire 5 min de plus..
- 9) C'est cuit..
- 10) À servir avec du riz blanc, des bananes revenu à la poêle et un peu d'avocat.