Institut National des Langues et Civilisations Orientales

Programmation itérative et récursive

Projet de fin de semestre

Annotation de recettes de cuisine et calcul de leur complexité en temps et en espace

PODER Solveig - M2 IM
REY Camille - M2 IM

Année universitaire 2020/2021

Sommaire

I. Introduction

L'objectif de notre projet était d'écrire un programme permettant de calculer automatiquement la complexité en temps et en espace d'une recette de cuisine. Cette complexité devait être calculée à la manière de la complexité d'un algorithme, les données d'entrée étant les ingrédients, l'espace mémoire les récipients utilisés.

Pour cette tâche, nous disposions d'un corpus de 23 071 recettes au format XML. Chaque recette de ce corpus est structurée de la manière suivante : titre, type, niveau, coût, ingrédients (chaque ingrédient est ensuite entouré d'une balise ***p***), et enfin le texte détaillant les étapes de préparation de la recette. Ce texte n'était pas balisé, or il était nécessaire de savoir quelles opérations étaient effectuées sur quels ingrédients. La première étape de notre travail a donc consisté à annoter ce texte et à relier chaque ingrédient annoté à l'opération qui lui était associée (en général le verbe qui régit le syntagme désignant l'ingrédient). Concernant l'annotation des ingrédients, nous avons évalué la précision et le rappel de notre fonction à l'aide de la liste des ingrédients en début de recettes. Nous avons pour cela essayé de gérer le problème des coréférences. En effet, par exemple, plusieurs ingrédients peuvent être regroupés sous une seule appellation générique (par exemple, « 4 fraises » et « 5 pommes » dans la liste des ingrédients peuvent être désignés par l'unique syntagme « les fruits » dans les étapes de préparation). Notre solution se base sur un lexique et, bien qu'imparfaite, a produit des résultats plutôt satisfaisants. Cependant, il est important d'avoir à l'esprit que les scores de rappel peuvent être surévalués puisqu'un même ingrédient, n'apparaissant qu'une fois dans la liste, peut apparaître à de multiples reprises dans le texte de la préparation et rien ne nous permet de connaître ce nombre d'apparitions à partir de la liste d'ingrédients seule.

Pour calculer la complexité, nous avions besoin de connaître les quantités des ingrédients pour en déduire les proportions. Or, étant donné que le texte de la recette ne précise que très rarement la quantité des ingrédients qu'il mentionne, notre seconde tâche a été de relier les ingrédients mentionnés dans le texte à la liste des ingrédients donnée au début de la recette pour associer à chaque opération la bonne quantité d'ingrédient. L'association entre la liste d'ingrédients et les ingrédients annotés dans la recette avait déjà été effectuée lors de l'évaluation. Cette étape n'a donc pas occasionné de nouvelles difficultés.

Pour nous aider dans ces premières tâches, nous avons constitué des lexiques (lexique d'ingrédients classés par catégories et lexique d'opérations culinaires) et utilisé un étiqueteur morphosyntaxique (SpaCy [6]). Cependant la tâche de l'étiqueteur était compliquée par les nombreuses coquilles que nous avons pu constater dans le corpus (espaces manquants, lettres en trop, fautes d'orthographe...), ce qui a pu générer des erreurs d'étiquetage et parfois un peu limiter la qualité de nos résultats par la suite.

Pour le calcul de la complexité en temps, des valeurs ont été associées aux opérations culinaires dans le lexique et un tableau de correspondance a également été créé pour les quantités d'ingrédients.

Concernant le calcul de la complexité en espace, la principale difficulté à laquelle il nous a fallu trouver une solution réside dans le fait que la plupart des récipients ne sont pas explicitement mentionnés dans le texte de la recette mais seulement implicitement induits par la nature des ingrédients (de la crème sera stockée dans un pot avant d'être utilisée) ou le sens des verbes désignant les opérations effectués sur les ingrédients (mélanger nécessite un contenant comme par

exemple un saladier). Nous avons donc complété le lexique des opérations culinaires avec les informations de nécessité ou non nécessité d'un récipient, et nous avons convenu que nous pouvions savoir quels ingrédients nécessitaient un récipient pour être stockés d'après leur catégorie (précisée dans notre lexique).

Une fois la complexité en temps et en espace calculé pour l'ensemble des recettes du corpus, nous avons analysé la corrélation entre cette complexité, d'une part, et, d'autre part, le niveau de difficulté de la recette. Il s'agissait de voir si cette corrélation était forte ou si la difficulté subjective d'une recette en appelait à d'autres critères. Les résultats n'ont pas été probants comme vous pourrez le constater dans la partie "Evaluation et analyses".

Pour terminer, nous avons calculé la complexité en temps de notre fonction d'annotation des ingrédients et opérations culinaires et avons pu constaté qu'il s'agissait d'un complexité linéaire, conditionnée par la taille de nos données.

II. État de l'art

L'extraction automatique d'ingrédients dans les recettes a fait l'objet de plusieurs travaux que nous avons mentionnés dans la bibliographie. Nous nous sommes particulièrement intéressées à l'article de T. Hamon et N. Grabar, Extraction of ingredient names from recipes by combining linguistic annotations and CRF selection [1]. Celui de R. Agarwal et K. Miller, Information Extraction from Recipes [2], dont le corpus est comme le nôtre semi-structuré en XML, nous a également fourni quelques pistes pour notre travail.

Hamon et Grabar traitent comme nous des données en français, ce qui est particulièrement intéressant pour nous, et leur travail est divisé en deux étapes. La première étape est un travail de reconnaissance d'entités nommées et utilise pour cela de nombreuses ressources lexicales, non seulement pour les ingrédients mais aussi pour d'autres entités propres aux recettes de cuisine (ustensiles, actions, quantités, durées...) que les auteurs ont ensuite projetées sur leur corpus à l'aide d'un outil (le module Perl Alvis::TermTagger). Les sites web ayant servi à constituer leurs ressources sont mentionnés dans l'article et nous avons pu nous en servir pour constituer notre lexique d'ingrédients que nous avons scrappé sur le site http://les.calories.free.fr/. Les auteurs ont ensuite utilisé un système à base de règles puis un classifieur automatique de type CRF pour distinguer les ingrédients les plus importants.

Agarwal et Miller travaillent quant à eux sur un corpus semi-structuré en XML, dont la structure est très similaire à celle de notre corpus. Leur démarche consiste à générer automatiquement une représentation de recettes interprétable par une machine en réduisant chaque étape de préparation à un prédicat (action) à un à deux arguments (ingrédient/patient, ustensile/agent), ce qui représente précisément ce dont nous avons besoin pour nos calculs de complexité. Leur travail est également réparti en deux étapes : une étape de reconnaissance d'entités nommées utilisant un modèle MEMM (Maximum Entropy Markov Model) puis un travail d'étiquetage en rôles sémantiques effectué à l'aide d'un algorithme à base de règles. Cette dernière étape s'appuie sur des arbres syntaxiques, ce qui nous semble en effet une solution très intéressante pour relier les actions aux ingrédients et aux ustensiles. Il est cependant fort dommage que les auteurs n'expliquent

pas le fonctionnement du module qu'ils utilisent pour régler les problèmes de coréférences que nous avons évoqués en introduction.

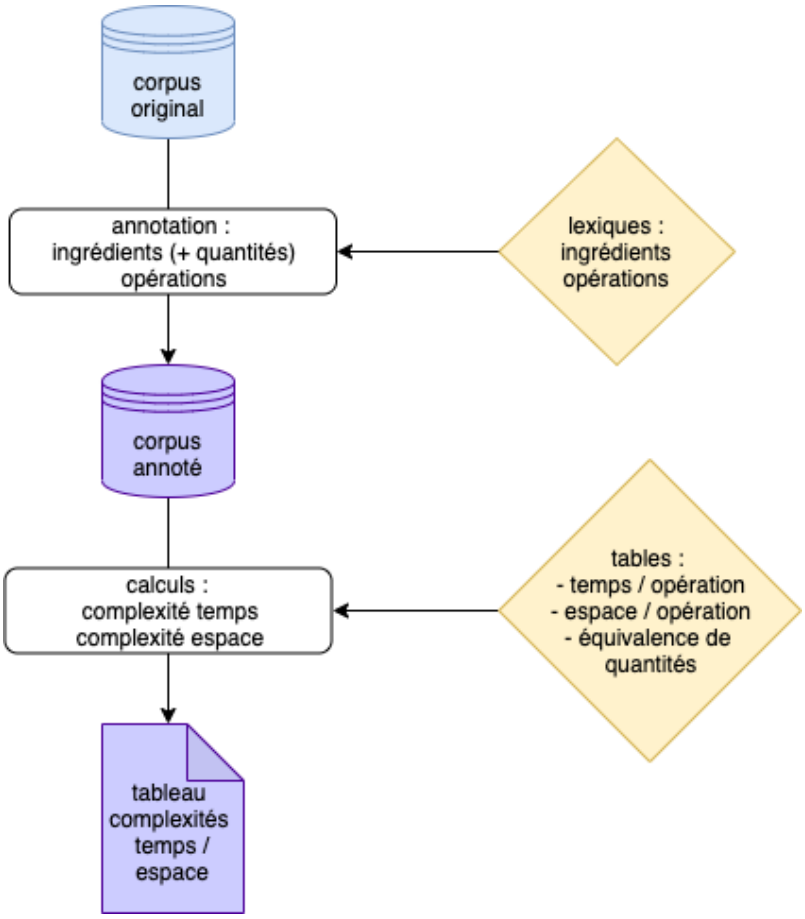
Concernant le calcul de complexité des recettes, qui est le sujet principal de notre projet, les travaux sont plus difficiles à trouver mais il nous faut mentionner l'article de Yun-Jung Lee, Seong-Min Yoon et Hwan-Gue Cho, Complexity and Similarity of Recipes based on Entropy Measurement [3]. Les auteurs y mesurent la complexité des recettes ainsi que la similarité entre recettes (il existe plusieurs autres travaux sur ce sujet, qui n'est pas celui qui nous intéresse ici), en utilisant l'entropie. Pour effectuer ce travail, ils commencent eux aussi par une extraction des ingrédients (en utilisant la liste des ingrédients et en la nettoyant de manière semi-automatique) et des actions effectuées sur eux (peu de précisions sont apportées sur cette étape). L'entropie permet de calculer le degré d'information apporté par ces éléments (si un ingrédient est utilisé dans une majorité de recettes, comme par exemple le sel, sa présence n'apporte que peu d'information) et sert à calculer deux types de complexité : une complexité de préparation basée sur la somme des entropies des ingrédients de la recette (la recette est complexe si les ingrédients sont nombreux et/ou peu communs) et une complexité de procédure basée sur la somme des entropies des verbes d'action (« cooking verbs »). Une visualisation du travail est apportée sous forme de graphes. Bien que ne correspondant pas à la méthode que nous avons utilisé pour ce projet, cet article est très intéressant et son approche nous a semblé pertinente.

III. Méthodologie

1. Méthodologie globale

Nous avons élaboré deux programmes principaux afin de répondre à la consigne du devoir. Tout d'abord, nous avons mis au point un programme chargé d'annoter le corps des recettes de cuisine. Le but de ce programme est d'identifier les ingrédients, les opérations culinaires, et leurs relations, ainsi que d'ajouter les informations de quantité pour les ingrédients. L'identification des ingrédients et opérations culinaires repose sur des lexiques, et non sur la liste d'ingrédients pour chaque recette, puisque cette liste doit servir pour l'évaluation. Nous avons donc dû commencer par générer un lexique d'ingrédients et un lexique d'opérations culinaires. Les informations de quantité pour chaque ingrédient sont en revanche extraites à partir de la liste d'ingrédients.

Un deuxième programme est chargé de calculer les complexités en temps et en espace à partir du corpus enrichi en informations (la sortie du programme précédent). La complexité en temps est exprimée en nombre d'opérations : à chaque opération culinaire identifiée est associé un nombre d'opérations élémentaires, qui dépend de la complexité de l'opération (opération simple ou complexe), et de la quantité d'ingrédients à laquelle elle est associée. La complexité finale d'une recette correspond à la somme de ces nombres. La complexité en espace est exprimée en nombre de récipients nécessaires pour stocker les ingrédients ou préparations au cours de l'exécution de la recette. Cette complexité prend en compte les récipients utilisés pour stocker les ingrédients de base nécessaires à la recette, et les récipients liés aux différentes opérations culinaires. Le programme de calcul des complexités en temps et en espace nécessite donc plusieurs tables que nous avons dû définir : une table de nombre d'opérations élémentaires par opération culinaire, une table de



complexité en espace des opérations culinaires, et une table de conversion des quantités d'ingrédients.

La figure 1 ci-dessous résume les différentes étapes de notre travail :

Figure 1 : méthodologie globale du travail

2. Création des lexiques

2.1 Lexique des ingrédients

Nous avons choisi d'identifier les ingrédients sur la base d'un lexique. Il aurait été plus simple d'utiliser directement la liste d'ingrédients fournie dans chaque recette, mais cela ne nous semblait pas judicieux puisque c'est cette liste même qui doit servir de référence pour l'évaluation de notre fonction d'identification d'ingrédients. De plus, il nous semblait plus intéressant de tenter de développer un programme capable de reconnaître les ingrédients dans un texte sans avoir besoin de métadonnées telles que la liste d'ingrédients.

Nous avons commencé par observer quelques recettes du corpus, afin d'anticiper les difficultés liées à l'identification d'ingrédients et de choisir une stratégie pour la génération du

lexique. La principale difficulté de la reconnaissance d'ingrédients concerne la coréférence : un même ingrédient peut être désigné par différentes unités linguistiques. Par exemple, il est possible d'avoir « brie » indiqué comme ingrédient de base d'une recette, puis « fromage » pour le désigner dans le corps de la même recette. Il est également possible que plusieurs ingrédients soient regroupés en un seul syntagme. Par exemple, une recette ayant «tomates, carottes, courgettes » dans ses ingrédients peut ne pas les faire apparaître directement dans le corps de sa recette, mais y faire référence dans une phrase telle que « coupez les légumes ». Dans les exemples précédents, les ingrédients sont désignés par des hyperonymes, c'est-à-dire des super-catégories, plus générales. Nous avons donc décidé d'organiser les ingrédients de notre lexique par catégories, afin d'exploiter ces catégories dans l'identification d'ingrédients plus tard. Ces catégories serviront également pour les calculs de complexité en espace.

Pour créer une première version de lexique d'ingrédients regroupés par catégories, nous avons d'abord extrait des données depuis le site web http://www.les-calories.fr/ . Cette plateforme propose des informations sur les valeurs nutritionnelles des aliments, qui sont déjà groupés en catégories. Nous avons extrait les informations d'aliments et de catégories via des techniques basiques de web scraping, puis généré un premier lexique au format JSON. Ce premier lexique était une bonne structure, mais avait besoin d'être enrichi pour constituer une base solide pour l'identification d'ingrédients.

Nous avons décidé d'exploiter les listes d'ingrédients des recettes de notre corpus pour enrichir le lexique d'ingrédients. Nous avons d'abord séparé le corpus en deux parties. 22 722 recettes ont été utilisées pour enrichir le lexique d'ingrédients. Les 349 recettes restantes, sélectionnées aléatoirement, ont été mises de côté pour servir plus tard à l'évaluation de notre fonction de reconnaissance des ingrédients. Nous avons utilisé un parser XML (BeautifulSoup [7]) ainsi que des expressions régulières afin d'extraire au mieux tous les ingrédients indiqués explicitement dans les listes d'ingrédients (balises <p>) du sous-corpus de 22 722 recettes. La sortie obtenue contenait environ 3000 syntagmes, dont beaucoup de bruit. Nous avons nous-mêmes filtré manuellement cette sortie afin de ne retenir que les syntagmes pertinents, et les avons rangés dans les différentes catégories de notre lexique de base. Nous avons également rajouté quelques catégories supplémentaires. Certaines catégories peuvent avoir plusieurs intitulés, séparés par un espace, par exemple « condiment assaisonnement épice ».

Certains noms d'ingrédients peuvent poser des problèmes d'ambiguïté, et nous avons dû décider de les garder ou non, en réfléchissant au choix qui engendrerait statistiquement le moins d'erreur. L'ingrédient « moule », par exemple, n'a pas été conservé. Après quelques tests, nous avons constaté que le token « moule » apparaissait plus souvent comme forme du mot « (un) moule », l'ustensile de cuisine, que comme forme du mot « (une) moule », le fruit de mer. Malheureusement, nous n'avons pas trouvé de moyen avec SpaCy d'identifier séparément les deux formes « moule ». Notre choix privilégie légèrement la précision au rappel : notre programme échouera à reconnaître l'ingrédient « moule », mais n'annotera pas à tort l'ustensile « moule » comme ingrédient.

Le lexique d'ingrédients final contient 994 ingrédients regroupés en 17 catégories.

La figure 2 illustre les différentes étapes pour la génération du lexique d'ingrédients. La figure 3 présente des extraits du lexique généré.

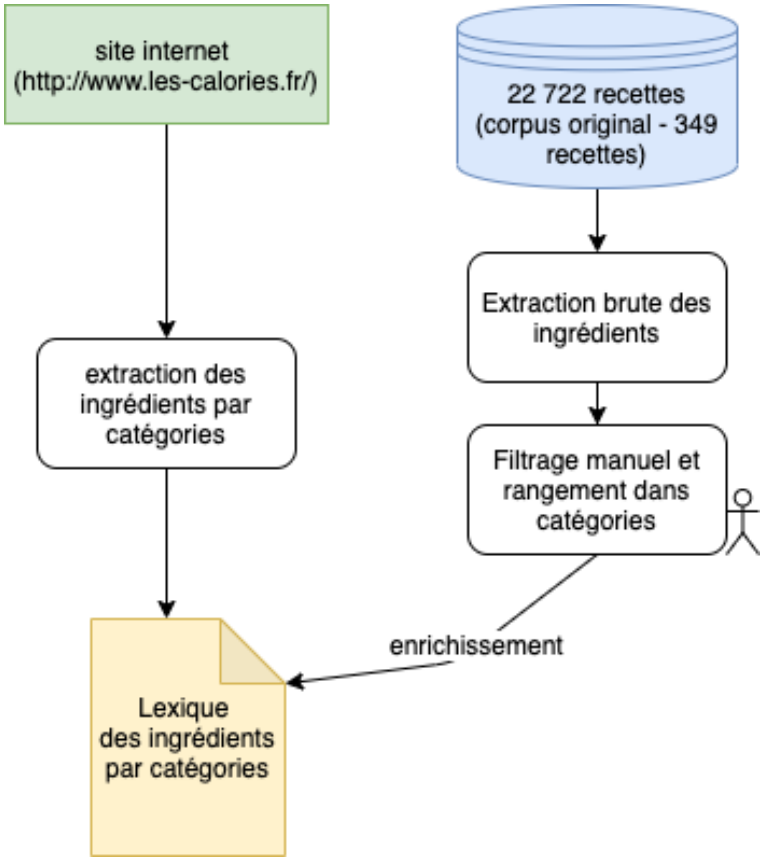


Figure 2 : étapes de la génération du lexique d'ingrédients

```
"fruit": [
    "abricot",
    "airelle",
    "amande",
    "ananas",
    "banane",
```

```
"fromage": [
    "babybel",
    "beaufort",
    "bleu",
    "bonbel",
    "boursault",
```

```
"condiment assaisonnement épice": [
    "basilic",
    "estragon",
    "aneth",
    "ail",
    "gousse d'ail",
```

Figure 3 : extraits du lexique d'ingrédients

2.2 Lexique des opérations culinaires

Pour pouvoir annoter les opérations culinaires dans les recettes, nous avons également décidé d'utiliser un lexique. Nous avons généré ce lexique d'opérations dans un deuxième temps, à partir du corpus original dans son intégralité, et en exploitant le lexique d'ingrédients généré à l'étape précédente.

Nous avons d'abord tokenisé et lemmatisé le texte de la recette à l'aide de la librairie SpaCy. Nous avons ensuite identifié les tokens « ingrédients » par correspondance avec le lexique d'ingrédients (le détail du processus d'identification utilisé est présenté dans la section 3.1). Nous avons ensuite émis l'hypothèse que les opérations culinaires étaient pour la grande majorité des verbes qui gouvernent des syntagmes ingrédients dans les recettes. Nous avons donc cherché à extraire ces verbes. Pour chaque ingrédient trouvé, nous avons extrait le verbe le gouvernant, ou à défaut la racine de la phrase grâce à l'analyse en dépendance fournie par SpaCy (ce processus est expliqué plus en détail dans la section 3.1). Nous avons ainsi récolté environ 3500 tokens lemmatisés identifiés comme verbes gouvernants ou racines par SpaCy. Ce grand nombre de résultats s'explique par différents phénomènes. Tout d'abord, l'orthographe varie d'un utilisateur à l'autre : utilisation d'accents ou non, fautes d'orthographe… Ces variations sont interprétées comme des verbes différents par SpaCy, multipliant ainsi le nombre de tokens de la sortie. De plus, la lemmatisation effectuée par SpaCy est limitée, certaines formes de verbe ne sont pas toujours mises à l'infinitif après lemmatisation. On observe donc plusieurs formes pour certains verbes dans la sortie, par exemple « émincer » et « émincez ». Enfin, l'analyse syntaxique en dépendance et le POS-tagging de SpaCy sont également limités, et peuvent fournir des résultats erronés. Il y avait donc beaucoup de bruit parmi les 3500 tokens en sortie, notamment des noms ou des symboles tels que le tiret « - » identifiés à tort comme des verbes ou racines de phrase.

Nous avons une fois de plus effectué un filtrage humain sur cette sortie, afin de conserver tous les tokens qui correspondaient sémantiquement à des opérations culinaires. Nous avons donc conservé les variantes avec ou sans accents, et les fautes d'orthographe. Nous avons également conservé les différentes formes de verbes (et pas uniquement l'infinitif) puisque nous ne savons pas dans quelles circonstances ni à quelle fréquence les erreurs de lemmatisation SpaCy se produisent, et que notre but était d'identifier un maximum d'opérations culinaires dans les textes de recette.

Le lexique d'opération final contient 847 tokens correspondant à des verbes d'opérations culinaires. La figure 4 ci-dessous illustre les étapes de la génération de ce lexique.

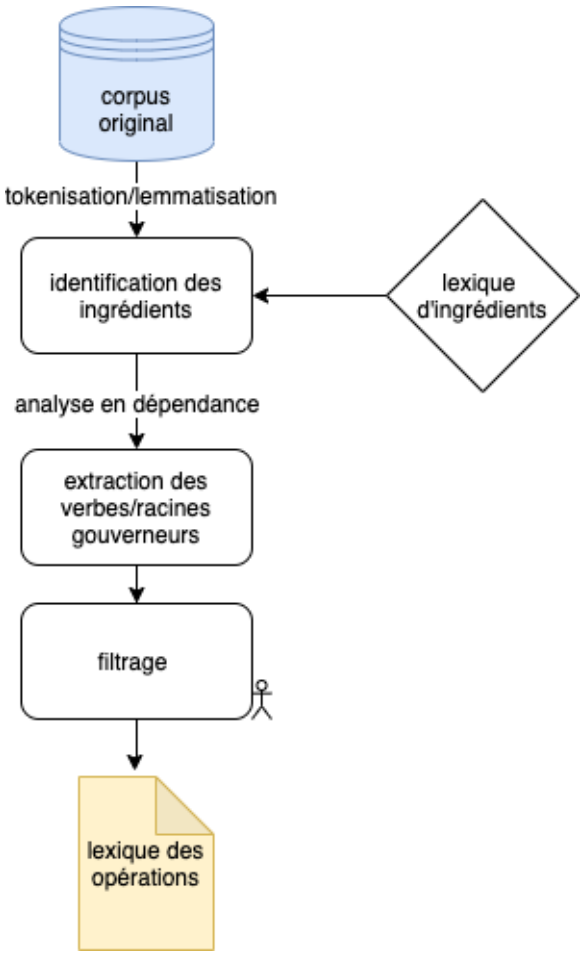


Figure 4 : étapes de la génération du lexique d'opérations

3. Annotation des recettes

Une fois les lexiques prêts, nous avons élaboré le programme d'annotation du texte de préparation d'une recette. Pour respecter l'intitulé du cours « Programmation itérative et récursive » dans le cadre duquel s'inscrit ce projet, nous avons utilisé un mélange de programmation récursive et itérative pour coder nos programmes principaux. Nous avons également utilisé un peu de programmation objet. Les librairies utilisées sont citées dans la bibliographie.

Pour une recette du corpus original au format XML, le programme d'annotation extrait le texte de la préparation (contenu de la balise <preparation>) et le prétraite en supprimant certains caractères et en ajoutant des espaces avant ou après d'autres pour faciliter la tokenisation. Le texte nettoyé est ensuite tokenisé et lemmatisé à l'aide de SpaCy. Chaque token lemmatisé est ensuite analysé individuellement. Nos annotations se font donc **à l'échelle d'un token**.

Les étapes du processus d'annotation des recettes sont illustrées dans la figure 5 ci-dessous.

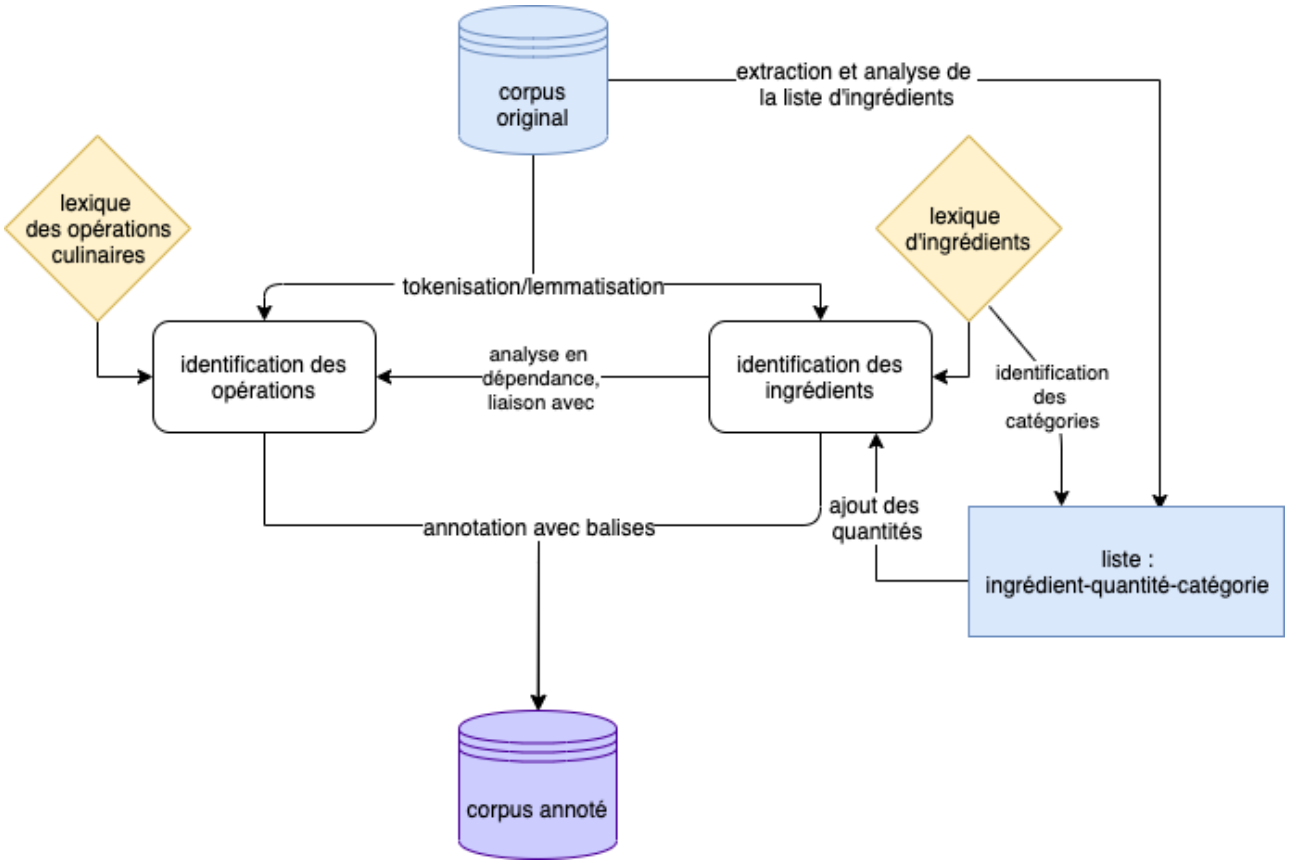


Figure 5 : étapes de l'annotation du corpus

3.1 Annotation des ingrédients et opérations

Tout d'abord, on cherche à savoir si le token analysé correspond à un ingrédient, ou bien à une catégorie d'ingrédients d'après notre lexique d'ingrédients. Cependant, dans notre lexique d'ingrédients original, certains ingrédients sont composés de plusieurs tokens. Pour permettre de meilleures correspondances, nous avons donc généré un lexique « simplifié », pour lequel uniquement le lemme du token « tête » (gouverneur) de chaque syntagme ingrédient est conservé. Ainsi, « chou chinois » est simplifié en « chou » et « beurre allégé » est simplifié en « beurre ». Nous avons conscience que cette simplification efface les différences entre certains ingrédients et peut créer des ambiguïtés (« beurre salé » et « beurre doux » sont regroupés en « beurre », « crème glacée » et « crème fraîche » sont regroupés en « crème »…), mais elle nous semblait nécessaire pour permettre l'identification d'ingrédients qui s'opère à l'échelle d'un seul token dans notre programme. De plus, les utilisateurs utilisent rarement la dénomination entière d'un ingrédient ailleurs que dans la liste des ingrédients. Par exemple, un utilisateur peut préciser «50g d'huile d'olive » et « 3 poivrons rouges » dans la liste des ingrédients, et indiquer dans les instructions « faites revenir les poivrons dans l'huile ». Pour chaque token lemmatisé de la recette, on cherche donc dans le lexique simplifié si ce token correspond à un ingrédient, ou bien à un intitulé de catégorie. Si c'est le cas, le token est considéré comme un ingrédient, et est annoté grâce à une balise <ingredient>.

Cette analyse à l'échelle d'un seul token implique de prendre quelques précautions supplémentaires pour éviter certaines erreurs. Par exemple dans « huile d'olive », « sucre glace » ou « cuillère à soupe », il ne faut pas que « olive », « glace » ou « soupe » soient annotés en tant

qu'ingrédients. Nous avons donc défini des règles simples, reposant sur les tokens précédents, afin d'éviter d'annoter à tort les deuxièmes tokens dans ce type de structures.

Une fois un ingrédient identifié, on cherche le verbe le gouvernant en se basant sur l'analyse syntaxique en dépendance fournie par SpaCy. Pour une phrase en entrée, SpaCy fournit une analyse syntaxique en arbre de dépendance qui représente les relations hiérarchiques des éléments de la phrase. Un exemple est donné en figure 6 pour la phrase « Today I go to school ». Cette analyse en dépendance permet d'obtenir pour chaque token le token qui le gouverne, et la relation de dépendance qui les lie. Le token « racine » (root) de la phrase est le seul token qui n'a pas de gouverneur. Pour trouver le verbe qui gouverne un ingrédient, nous avons donc utilisé une fonction récursive qui remonte de gouverneur en gouverneur dans la phrase jusqu'à trouver un verbe (grâce au POS-tagging SpaCy), ou bien la racine de la phrase. Si le lemme de ce verbe fait partie du lexique d'opérations culinaires, alors on considère qu'il s'agit de l'opération culinaire à laquelle se rapporte l'ingrédient, et le lemme de ce verbe est ajouté à l'annotation de l'ingrédient sous la forme d'un attribut « action » dans la balise <ingredient>. Le token correspondant à cette opération est alors également annoté à l'aide d'une balise <operation>, et l'ingrédient est ajouté à l'attribut « ingredients » de cette balise. Si aucune opération culinaire n'est trouvée pour l'ingrédient analysé, alors l'attribut « action » n'est pas ajouté. Un exemple illustratif concret est donné en figure 7.

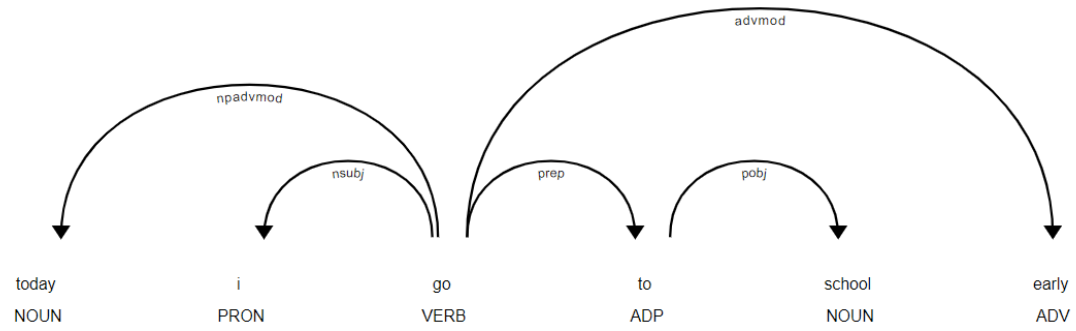


Figure 6 : analyse en dépendance SpaCy

Toutes les opérations culinaires ne sont pas forcément explicitement liées à des ingrédients. Par exemple, la phrase « Bien remuer avant de servir » comporte l'opération culinaire « remuer », qui ne pourra pas être déduite à partir de l'identification d'un ingrédient (selon le processus présenté dans le paragraphe précédent). Il reste donc nécessaire de vérifier pour chaque token lemmatisé s'il s'agit d'une opération culinaire d'après notre lexique, et de l'annoter avec une balise <operation> si c'est le cas.

```
phrase :
"coupez les oignons et les tomates. Mixez à l'aide d'un robot"
annotation :
"<operation ingredients="oignons;tomates"> coupez </operation> les
<ingredient action="couper"> oignons </ingredient> et les
<ingredient action="couper"> tomates </ingredient> . <operation> Mixez
</operation> à l' aide d' un robot"
```

Figure 7 : exemple d'annotation de phrase (avant ajout des quantités)

3.2 Ajout des informations de quantités

La fonction d'identification des ingrédients et des opérations présentée précédemment n'utilise pas la liste d'ingrédients fournie dans chaque recette. Cependant, la détermination des quantités ne peut pas se faire uniquement à partir du texte de la préparation. Il est en effet très rare que les quantités soient explicitées dans les instructions de la recette. Par exemple, si la liste des ingrédients indique « 3 tomates », il sera rarement précisé « coupez les **3** tomates » dans les instructions, et on trouvera plutôt « coupez les tomates ». Il est donc nécessaire de consulter la liste des ingrédients afin d'essayer de déterminer les quantités pour chaque ingrédient identifié à l'étape précédente.

La première étape consiste à extraire les informations de la liste d'ingrédients. Dans la majorité des listes d'ingrédients du corpus, un ingrédient est contenu dans une balise , par exemple « ** ** 3 pavés de saumon ** ** ». Ce n'est cependant pas toujours le cas. On peut trouver des cas où plusieurs ingrédients sont groupés à l'intérieur d'une même balise , par exemple : « ** ** sel, poivre, curcuma ** ** » ou bien « ** ** Pour la pâte : -100g de beurre -100g de sucre ** ** ». Il peut également arriver qu'une balise ne contienne aucun ingrédient, par exemple :

« ** ** Pour la pâte ** **
** ** 100g de beurre ** **
** ** 100g de sucre ** ** »

Nous avons utilisé les expressions régulières pour prendre en compte ces cas et isoler au mieux les ingrédients et leur quantité depuis ces listes. Nous avons ensuite tâché de séparer la quantité et l'ingrédient. La quantité peut être exprimée à travers un nombre seul, ou à travers un nombre accompagné d'une unité. Cette unité peut être exprimée avec une multitude d'expressions différentes, plus ou moins précises. Elle peut faire appel à des systèmes de mesure concrets, tels que les grammes ou les litres, ainsi que les autres unités qui en découlent (milligrammes, centilitres…). Elle peut aussi être exprimée par des contenants tels que « pot », « sachet », « boîte », « cuillère à soupe », par des sous-parties d'aliments telles que « branche » ou « feuille », par des coupes telles que « filet » ou « pavé », ou encore par des expressions plus vagues telles que « un peu de », « une pincée de ». Afin d'identifier toutes ces formes de quantité, nous avons tokenisé et lemmatisé le texte de chaque ingrédient, puis utilisé une fois de plus des expressions régulières, en tâchant d'être le plus exhaustif possible.

Une fois l'expression représentant la quantité identifiée (si elle existe), nous avons cherché à extraire le token principal représentant l'ingrédient dans le syntagme restant. Nous avons à nouveau décidé pour cela d'extraire la tête du syntagme. Une difficulté posée par SpaCy est que les adjectifs de forme « participe passé » étant identifiés comme des verbes, ils sont souvent considérés à tort comme tête de syntagme. Par exemple, pour « chou frisé » ou « saumon fumé », « frisé » et « fumé » seront considérés comme tête de syntagme par SpaCy. Pour éviter ce genre d'erreur, nous avons supprimé les tokens POS-taggés en « verbes » des syntagmes avant d'en chercher la tête. Il s'est aussi avéré que certains adjectifs tels que « frais » ou « sec » sont quasi-systématiquement considérés comme tête de syntagme par SpaCy, nous les avons donc également supprimés.

Il peut arriver qu'un même ingrédient soit plusieurs fois présents dans la liste d'ingrédients, notamment quand il sert à deux préparations différentes. Dans ces cas-là, nous avons défini la

quantité de l'ingrédient comme la somme (séparé par l'opérateur « + ») des quantités indiquées dans la liste d'ingrédients. Par exemple, pour une liste qui indiquerait :

« ** ** Pour le coulis : ** **
** ** 100g de fraises ** **
…..
** ** Pour décorer : ** **
** ** Quelques fraises ** ** »

la quantité extraite pour l'ingrédient « fraise » serait « 100 gramme + quelque ». Nous avons conscience que cette solution pourra fausser certaines opérations dans le calcul de complexité en temps, mais nous avons considéré que ne conserver qu'une seule des quantités indiquées fausserait encore davantage les calculs.

Enfin, pour chaque ingrédient extrait de la liste, nous avons cherché la ou les catégorie(s) à laquelle il appartient, s'il existe dans notre lexique d'ingrédient. Ainsi, la catégorie de « saumon » sera « poisson », et les catégories de « salade » seront « légume » et « crudité ». Pour les ingrédients qui ne figurent pas dans notre lexique, la catégorie « non-définie » est attribuée par défaut.

Les informations extraites depuis les listes d'ingrédients sont donc des listes de triplets « ingrédient / quantité / catégorie(s) ». La figure 8 représente une liste d'ingrédients et les informations qui en seront extraites.

```
<ingredients>
  <p>3 tranches de saumon fumé</p>
  <p>500g de courgettes</p>
  <p>2 boîtes de thon</p>
  <p>sel, poivre</p>
</ingredients>
```

ingredient	quantité	catégorie(s)
saumon	3 tranche	« poisson »
courgette	500 gramme	« légume »
thon	2 boîte	« poisson »
sel	(chaîne vide)	« condiment assaisonnement épice »
poivre	(chaîne vide)	« condiment assaisonnement épice »

Figure 8 : exemple d'extractions d'informations depuis une liste d'ingrédients

Ces informations vont servir à ajouter les quantités aux annotations des ingrédients dans le texte de préparation de la recette. Pour chaque ingrédient identifié dans le texte, on cherche s'il est présent dans les ingrédients extraits depuis la liste d'ingrédients. S'il l'est, la quantité extraite pour cet ingrédient est ajoutée à l'annotation de l'ingrédient identifié sous la forme d'un attribut « quantite ». Si ce n'est pas le cas, on cherche si l'ingrédient identifié fait partie des catégories de un ou plusieurs ingrédients extraits de la liste d'ingrédients. Si c'est le cas, les quantités de tous les ingrédients appartenant à la catégorie, séparées par des points-virgules, sont attribuées comme valeur de l'attribut « quantite ». Si l'ingrédient identifié ne correspond à aucun des ingrédients ou catégories d'ingrédients extraits de la liste d'ingrédients, l'attribut « quantite » n'est pas ajouté. La figure 9 présente l'annotation de la phrase « Mélangez les poissons avec les courgettes » en se référant à la liste d'ingrédients en figure 8.

```
<operation ingredients="poissons"> Mélangez </operation> les
<ingredient action="mélanger" quantite="3 tranche;2 boîte"> poissons </ingredient>
avec les <ingredient action="mélanger" quantite="500g"> courgettes </ingredient>
```

Figure 9 : exemple d'annotation finale sur une phrase

Les figures 10 et 11 présentent une recette complète du corpus original en entrée, et sa sortie annotée par notre programme (recette d'identifiant 24823). On constate que l'annotation est globalement bien réalisée, malgré quelques erreurs. Par exemple, le nom « place » est identifié à tort comme l'opération culinaire « placer », et l'ingrédient « huile » est associée à l'opération culinaire « découper ».

```
<?xml version="1.0" encoding="utf-8"?>
<recette id="24823">
    <titre>Poulet au yaourt et paprika</titre>
    <type>Plat principal</type>
    <niveau>Facile</niveau>
    <cout>Bon marché</cout>
    <ingredients>
        <p>1 poulet</p>
        <p>1 yaourt</p>
        <p>4 oignons</p>
        <p>1 gousse d'ail</p>
        <p>1 cuillère à soupe de paprika</p>
        <p>20 cl de bouillon de volaille</p>
        <p>4 cuillères à soupe d'huile</p>
        <p>sel et poivre</p>
    </ingredients>
    <preparation>
        <![CDATA[
    Hachez les oignons.
    Faites revenir le poulet découpé en morceaux dans une cocotte avec l'huile.
    Une fois les morceaux dorés, retirez-les et mettez à leur place les oignons, laissez-les fondre.
    Hors du feu saupoudrez de paprika, salez, poivrez et versez le bouillon.
    Remettez sur le feu, rajoutez les morceaux de poulet et l'ail écrasé.
    Laissez mijoter 1 h à feu doux.
    En fin de cuisson, versez le yaourt battu, mélangez et servez.
  ]]>
    </preparation>
</recette>
```

Figure 10 : recette avant annotation

```xml
<?xml version="1.0" encoding="utf-8"?>
<recette id="24823">
    <titre>Poulet au yaourt et paprika</titre>
    <type>Plat principal</type>
    <niveau>Facile</niveau>
    <cout>Bon marché</cout>
    <ingredients>
        <p>1 poulet</p>
        <p>1 yaourt</p>
        <p>4 oignons</p>
        <p>1 gousse d'ail</p>
        <p>1 cuillère à soupe de paprika</p>
        <p>20 cl de bouillon de volaille</p>
        <p>4 cuillères à soupe d'huile</p>
        <p>sel et poivre</p>
    </ingredients>
    <preparation>
        <operation ingredients="oignons"> hachez </operation> les <ingredient action="hacher" quantite="4">
        oignons </ingredient> . faites <operation ingredients="poulet"> revenir </operation> le
        <ingredient action="revenir" quantite="1"> poulet </ingredient> <operation ingredients="huile"> découpé
        </operation> en morceaux dans une cocotte avec l' <ingredient action="découper" quantite="4 cuillère à soupe">
        huile </ingredient> . une fois les morceaux dorés , <operation> retirez </operation> - les et
        <operation ingredients="oignons"> mettez </operation> à leur <operation> place </operation> les
        <ingredient action="mettre" quantite="4"> oignons </ingredient> , laissez - les <operation> fondre </operation>
        . hors du feu <operation ingredients="paprika"> saupoudrez </operation> de
        <ingredient action="saupoudrer" quantite="1 cuillère à soupe"> paprika </ingredient> , <operation> salez
        </operation> , <operation> poivrez </operation> et <operation ingredients="bouillon"> versez </operation> le
        <ingredient action="verser" quantite="20 cl"> bouillon </ingredient> . <operation> remettez </operation> sur
        le feu , <operation ingredients="poulet"> rajoutez </operation> les morceaux de
        <ingredient action="rajouter" quantite="1"> poulet </ingredient> et l' <ingredient action="écraser"> ail </ingredient>
        <operation ingredients="ail"> écrasé </operation> . laissez <operation> mijoter </operation> 1 h à feu doux . en fin
        de cuisson , <operation ingredients="yaourt"> versez </operation> le <ingredient action="verser" quantite="1"> yaourt
        </ingredient> <operation> battu </operation> , <operation> mélangez </operation> et servez .
    </preparation>
</recette>
```

Figure 11 : recette après annotation

4. Calcul des complexités

Un seul programme est chargé de calculer les complexités en temps et en espace de toutes les recettes du corpus. Une sortie unique est ensuite générée au format csv. Chaque ligne de la sortie contient le nom du fichier annoté analysé, sa complexité en temps, sa complexité en espace, et son niveau de difficulté (niveau allant de 1 à 4, pour « très facile » à « difficile », déjà indiqué dans la recette originale). Les étapes du calcul de la complexité sont illustrées par la figure 12 ci-dessous :

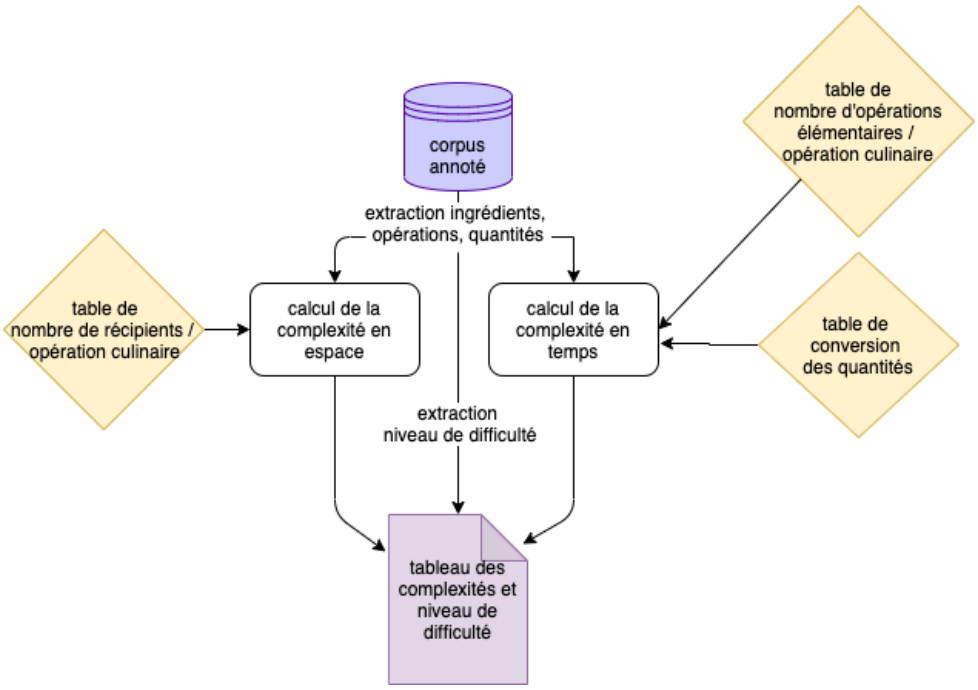


Figure 12 : étapes des calculs de complexité

4.1 Complexité en temps

Le calcul de la complexité en temps s'effectue sur la base du corpus enrichi en annotations d'ingrédients, quantités et opérations culinaires. Ce calcul s'inspire du calcul de complexité en temps d'un algorithme de programmation. On cherche à connaître le nombre d'opérations élémentaires de la recette. Ce nombre d'opérations élémentaires dépend des opérations culinaires à effectuer (équivalent des instruction d'un programme), qui peuvent être plus ou moins complexes, et de la quantité d'ingrédients (équivalent des données fournies à un programme) à laquelle elles s'appliquent.

Prenons par exemple une recette dont les instructions seraient simplement « casser les œufs ». Si on définit que l'opération « casser » correspond à une seule opération élémentaire, et qu'il est indiqué d'utiliser 3 œufs dans la liste d'ingrédients, alors la complexité sera égale à 1x3 soit 3. Prenons maintenant une opération plus difficile telle que « dénerver un foie gras ». L'opération « dénerver » est une opération complexe, il faut enlever les nerfs soigneusement un à un. On peut considérer que cette opération culinaire plus longue et complexe correspond à 4 opérations élémentaires. La complexité en temps d'une telle recette sera donc 4x1, soit 4. Modifions légèrement la recette, en changeant la quantité de foie gras. Pour « dénerver un demi foie gras », le travail est deux fois moins important et prendra deux fois moins de temps, la complexité sera donc 4 x 0.5, soit 2. Enfin, pour une recette composée de plusieurs opérations culinaires (simples ou complexes), la complexité totale sera la **somme** des complexités de chaque opération (en fonction de la quantité d'ingrédients). Ainsi, la complexité de la recette « coupez un demi foie gras et cassez 5 œufs » sera : « 4 x 0.5 + 1 x 5 » soit 7. Ce calcul est réalisable sur nos recettes grâce aux annotations d'opérations culinaires, mais nécessite d'avoir défini quelques tables d'informations : le nombre d'opérations élémentaires par opération culinaire (par exemple 4 pour « dénerver » et « 1 » pour casser), et les équivalences de quantités (par exemple 1 pour « boîte », 0.1 pour « centilitre »…).

Pour créer la table du nombre d'opérations élémentaires par opération culinaire, nous avons repris le lexique d'opérations culinaires, puis l'avons converti vers un format tabulaire, afin d'ajouter une colonne correspondant au nombre d'opérations élémentaires. Pour nous économiser du temps de travail, nous n'avons rempli cette colonne que pour les opérations que nous considérions « complexes », pour lesquelles le nombre d'opération élémentaires est supérieur à 1 (les autres opérations auront par défaut 1 comme nombre d'opération élémentaire). Nous avons par exemple considéré que « cuire » représentait 3 opérations élémentaires, car il est en général nécessaire de vérifier plusieurs fois l'avancée de la cuisson, ou de remuer. L'action « décortiquer » représente 4 opérations élémentaires, car le processus consiste en général à retirer plusieurs parties d'un ingrédient, par exemple la tête, la carapace, et les pattes pour une crevette.

Pour convertir les quantités, nous avons créé une table d'équivalence. Comme expliqué précédemment, les quantités sont en général exprimées soit avec un nombre seul, soit avec un nombre accompagné d'une unité. Dans le cas d'un nombre seul, aucune conversion n'est nécessaire. Par exemple, pour « 3 aubergines », la quantité « 3 » peut être réutilisée telle quelle dans le calcul de complexité. En revanche, pour « 500 g d'aubergines », on ne peut utiliser la quantité « 500g » telle quelle dans le calcul, c'est pourquoi une table de conversion des unités est nécessaire. Nous avons défini par exemple que « boîte », « pavé » ou « filet » correspondaient tous à une unité de quantité, tandis que « tranche » correspond à 0.25 unité de quantité, et « branche » à 0.1. Pour les poids, nous avons défini « 100 grammes » comme 1 unité de quantité, tandis que pour les liquides, nous avons

choisi « 10 centilitres ». Par conséquent, l'unité « gramme » vaut 0.01, « kilogramme » vaut 10, « centilitre » vaut 0.1… Une fois convertie, l'unité est multipliée par le nombre qui la précède. Ainsi, la quantité « 500g » sera convertie en 500 x 0.01, soit 5 unités de quantité. « 3 tranches » sera converti en 3 x 0.25, soit 0.75 unité de quantité.

Notre système de conversion est très simplifié, car il ne prend pas en compte le type d'ingrédient traité. Cela peut créer des incohérences notamment au niveau des unités de poids. Nous avons choisi 100 grammes comme équivalent d'une unité de quantité, car cela correspond grossièrement à un yaourt, un oignon, une carotte, une portion de viande pour une personne… Mais imaginons une recette qui indiquerait dans la liste des ingrédients « 1 poulet entier », et une autre recette qui indiquerait « 1,8 kg de poulet ». Dans le premier cas, la quantité après conversion sera « 1 », tandis que dans le deuxième cas, la quantité sera « 18 ». La quantité réelle de poulet dans les deux recettes est pourtant très proche. Cet aspect de notre méthode reste à améliorer pour prendre en compte ce genre de cas.

Grâce à ces tables, on peut calculer la complexité en temps de chaque opération culinaire d'une recette et additionner l'ensemble pour obtenir la complexité totale de la recette. Précisons que pour des opérations culinaires qui ne sont liées à aucun ingrédient, la complexité correspond simplement au nombre d'opérations élémentaires de cette opération d'après notre table. À titre d'exemple, détaillons le calcul de complexité en temps pour une phrase de la recette présentée précédemment en figure 11.

Phrase annotée :
hors du feu **<operation ingredients="paprika">** saupoudrez **</operation>** de **<ingredient action="saupoudrer" quantite="1 cuillère à soupe">** paprika **</ingredient>** , **<operation>** salez **</operation>** , **<operation>** poivrez **</operation>** et **<operation ingredients="bouillon">** versez **</operation>** le **<ingredient action="verser" quantite="20 cl">** bouillon **</ingredient>** .

D'après nos tables :
saupoudrez = 1 opération élémentaire
saler = 1 opération élémentaire
poivrer = 1 opération élémentaire
verser = 1 opération élémentaire
cuillère à soupe = 0.2 unité de quantité —> 1 cuillère à soupe = quantité de 0.2
cl = 0.1 unité de quantité —> 20 cl = quantité de 2

Calcul :
complexité_temps = (saupoudrez x quantité_paprika) + saler + poivrer + (verser x quantité_bouillon)
= (1 x 0.2) + 1 + 1 + (1 x 2)
= 4.2

4.2 Complexité en espace

La complexité en espace correspond au nombre de récipients nécessaires pour stocker les ingrédients au cours de la préparation. Nous partons du principe que certains ingrédients nécessitent un récipient pour les stocker dès le début de la recette. En fonction des opérations culinaires à effectuer au cours de la recette, des récipients supplémentaires peuvent être nécessaires.

Les emballages du commerce ne sont pas considérés comme des récipients. Par exemple, nous ne considérons pas que le tube plastique d'un tube de mayonnaise soit un récipient, ni l'emballage d'une plaquette de beurre. Pour définir quels ingrédients nécessitent un récipient de stockage, nous nous sommes une fois de plus appuyés sur les catégories définies dans notre lexique d'ingrédients. Nous avons ainsi défini que seuls les ingrédients appartenant aux catégories « coquillage crustacés », « poisson », « viande », et « volaille » nécessitent un récipient, car ils ne peuvent à priori pas être posés directement sur le plan de travail (contrairement aux fruits et légumes). On ne prend pas en compte la quantité d'ingrédients, car on considère que tous les ingrédients identiques sont groupés dans un seul récipient (il n'y aura pas un récipient différent pour chaque crevette, par exemple).

Pour définir quelles opérations culinaires nécessitent un récipient supplémentaire, nous avons ajouté une troisième colonne au lexique des opérations culinaires, que nous avons remplie uniquement pour les opérations nécessitant un récipient. Par exemple, nous avons considéré que « mélanger » implique l'utilisation d'un récipient supplémentaire, en général un bol ou un saladier. « enfourner » nécessite un plat ou une plaque de cuisson, « poêler » nécessite une poêle… Nous avons longuement hésité sur le traitement des opérations culinaires qui apparaitraient plusieurs fois dans une recette. Par exemple, si l'opération « mélanger » apparaît plusieurs fois au cours d'une recette, faut-il considérer qu'un récipient supplémentaire est nécessaire pour chaque occurrence de « mélanger », ou bien que toutes les opérations « mélanger » s'effectueront dans un seul et même récipient ? Il n'y a pas de bonne réponse à cette question, aucun des deux traitements n'est idéal. Nous avons tout de même émis l'hypothèse que les opérations similaires seront effectuées dans un seul et même récipient dans la plus grande partie des cas. Ainsi, même si l'opération « mélanger » est présente plusieurs fois dans les instructions d'une recette, un seul récipient sera ajouté dans le calcul de la complexité en espace.

La complexité en espace totale d'une recette est donc calculée en additionnant le nombre d'ingrédients nécessitant un récipient (d'après leur catégorie) et le nombre d'opérations distinctes nécessitant un récipient (d'après la troisième colonne de notre lexique). À titre d'exemple, détaillons le calcul de complexité en espace pour la recette suivante :

```
<ingredients>
<p>10 coquilles saint jacques</p>
<p>15 crevettes</p>
<p>3 poireaux</p>
<p>sel, poivre</p>
</ingredients>
<preparation>
<operation ingredients="saumon;poireaux">Coupez</operation> les
<ingredient action="couper" quantite="3">poireaux</ingredient> .
Faites <operation ingredients="coquilles;crevettes"> revenir les
<ingredient action="revenir" quantite="10">coquilles</ingredient> et les
<ingredient action="revenir" quantite="15">crevettes</ingredient> à la poêle jusqu'à
ce qu'elles soient bien colorées. <operation ingredients="poireaux">Ajoutez
</operation> les <ingredient action="ajouter" quantite="3">poireaux</operation> .
<operation> Salez </operation> , <operation> poivrez </operation>.
<operation> Dresser </operation> le tout dans un plat et servez.
</preparation>
</recette>
```

Les ingrédients de base sont « coquilles saint jacques », « crevettes », « poireaux », « sel » et « poivre ». Les ingrédients « crevettes » et « coquilles saint jacques » appartiennent à la catégorie « coquillage crustacé », dont nous avons défini que les ingrédients nécessitent un récipient pour le stockage. Les ingrédients « poireaux » et « sel , poivre » appartiennent respectivement aux catégories « légume » et « condiment assaisonnement épice », qui ne font pas partie des catégories dont les ingrédients nécessitent un récipient de stockage. Il y a donc besoin de deux récipients pour les ingrédients : un pour les coquilles, et un pour les crevettes.

La liste des opérations culinaires de la recette est : « couper », « revenir », « ajouter », « saler », « poivrer » et « dresser ». Parmi ces opérations, nous avons défini dans notre lexique que « revenir » et « dresser » nécessitent un récipient. Il y a donc besoin de deux récipients pour stocker les ingrédients ou préparations au cours de la recette.

La complexité en espace totale d'une recette correspond à la somme des récipients nécessaires pour le stockage des ingrédients et des récipients nécessaires aux opérations culinaires, elle vaut donc 2 + 2, soit 4 pour cet exemple.

IV. Évaluation et analyses

1. Évaluation de la reconnaissance d'ingrédients

Comme nous l'avons expliqué, notre fonction de reconnaissance d'ingrédients s'appuie sur des lexiques, et non sur la liste d'ingrédients fournie dans chaque recette. Cette liste nous a en effet servi de référence pour évaluer les performances de la fonction de reconnaissance d'ingrédients. Nous avons effectué deux évaluations : une sur le corpus entier, et une sur le sous-corpus de 349 recettes qui n'avaient pas servi à l'enrichissement du lexique d'ingrédients.

Les métriques d'évaluation que nous avons utilisées sont la précision et le rappel. La précision permet d'évaluer le bruit, c'est-à-dire les éléments non-pertinents parmi les résultats. Le rappel permet d'évaluer le silence, c'est à dire les éléments pertinents « oubliés », qui n'ont pas été identifiés.
La précision est définie par la formule : **vrais positifs / (vrais positifs + faux positifs)**
Le rappel est défini par la formule : **vrais positifs / (vrais positifs + faux négatifs)**
où « vrais positifs » désigne les éléments reconnus par le programme qui sont pertinents, i.e. les ingrédients identifiés dans le texte de préparation qui font effectivement partie de la liste d'ingrédients fournie par la recette. L'expression « faux positifs » désigne quant à elle les éléments identifiés à tort par le programme, c'est-à-dire des ingrédients qui ne font pas partie de la liste d'ingrédients. Enfin, les « faux négatifs » sont les éléments pertinents non-reconnus par le programme, i.e. les ingrédients mentionnés dans la liste d'ingrédients mais qui n'apparaissent pas dans les ingrédients identifiés par le programme d'annotation.

Les ingrédients de la liste d'ingrédients servant de référence sont extraits de la même manière que celle décrite dans la section 3.2 pour l'ajout des quantités, les comparaisons se font donc à l'échelle du token tête de syntagme, lemmatisé.

Nous avons tenu compte des « ingrédients-catégories » dans l'évaluation. Par exemple, si la liste d'ingrédients d'une recette indique « tomate, courgette, aubergine » comme ingrédients, et que la fonction d'identification d'ingrédients ne trouve aucun de ces ingrédients, mais trouve en revanche « légumes », alors on considèrera que « tomate, courgette, aubergine » et « légumes » correspondent aux mêmes ingrédients, puisque « légume » est une catégorie qui comprend «tomate », « courgette » et « aubergine ». L'ingrédient « légumes » identifié ne sera donc pas considéré comme un faux positif, et les ingrédients « tomate », « courgette » et « aubergine », bien que non identifiés directement, ne seront pas considérés comme des faux négatifs.

Nous avons également géré le cas particulier de « sel » et « poivre », qui sont souvent indiqués dans la liste des ingrédients, mais « fusionnés » avec le verbe dans le texte de la préparation (« saler » et « poivrer »). Ainsi, dans les cas où « sel » et « poivre » seraient indiqués dans la liste d'ingrédients sans être identifiés par la fonction de reconnaissance d'ingrédients, si « saler » et « poivrer » ont été identifiés comme opérations, alors « sel » et « poivre » ne seront pas considérés comme des faux négatifs.

Illustrons le processus d'évaluation avec l'exemple fictif suivant :

ingrédients attendus (liste d'ingrédients)	ingrédients trouvés (fonction d'identification)
tomate	crudité
concombre	tomate
radis	salade
huile	huile
sel	
poivre	
vinaigre	

Dans cet exemple, le seul faux positif est « salade », puisque cet ingrédient n'apparaît pas dans la liste des ingrédients attendus. Le terme « crudité » n'est pas un faux positif, car il s'agit d'une catégorie à laquelle appartiennent les ingrédients attendus « tomate », « concombre » et « radis ». L'ingrédient « vinaigre » est un faux négatif, car il fait partie des ingrédients attendus, mais il n'apparait pas parmi les ingrédients trouvés. En revanche, « radis » et « concombre » ne sont pas des faux négatifs, car ils correspondent à la catégorie « crudité ». Enfin, « sel » et « poivre » ne seront considérés comme des faux négatifs que si « saler » et « poivrer » n'ont pas été identifiés parmi les opérations culinaires de la recette. La précision pour cet exemple vaudra donc 3/4 soit 0.75, et le rappel vaudra 3/4 ou 3/6 (en fonction de « sel » et « poivre »).

L'évaluation produite sur l'ensemble des 23 071 recettes est enregistrée sous forme d'un tableau csv avec le nom du fichier, la précision et le rappel pour chaque recette. Une ligne est ajoutée à la fin avec les moyennes de précision et de rappel, respectivement 0.60 et 0.69.

L'évaluation produite sur le corpus d'évaluation de 349 recettes n'ayant pas servi à la création du lexique d'ingrédients est également enregistrée au même format, et présente des résultats légèrement inférieurs : 0.58 pour la précision et 0.67 pour le rappel.

Ces résultats nous semblent très satisfaisants, dans la mesure où une fonction vraiment peu performante présenterait des résultats proches de 0 (contrairement aux tâches de classification binaires par exemple, pour lesquelles on prend souvent 0.5 de f-mesure comme référence en dessous de laquelle un classifieur serait moins performant que le hasard). Plusieurs raisons expliquent les résultats d'évaluation que nous observons.

Tout d'abord les ingrédients servant de référence pour l'évaluation dépendent eux-mêmes de fonctions que nous avons créées pour les extraire depuis les listes d'ingrédients. Ces fonctions sont imparfaites, et il n'est donc pas impossible que la liste d'ingrédients « attendus » servant de base à l'évaluation soit elle-même parfois erronée. Aussi, il est très courant que les utilisateurs ne fassent pas explicitement référence dans les instructions à tous les ingrédients de leur liste, ou bien qu'ils les désignent par d'autres syntagmes (phénomène de coréférence). Notre système de catégories est une tentative de solution à ce problème, mais ne suffit évidemment pas à gérer tous les phénomènes de coréférence. La coréférence est sans nul doute la cause d'une grande partie des faux négatifs, et fait baisser le rappel. Les erreurs de lemmatisation de SpaCy peuvent également être à l'origine de mauvaises correspondances entre les ingrédients trouvés et les ingrédients attendus. En observant le

détail du processus d'évaluation, nous avons constaté par exemple des cas où « soja » était lemmatisé tantôt en « soja », tantôt en « sojer », résultant en un faux négatif et un faux positif, alors que l'ingrédient est bien le même. Enfin, il arrive également très souvent que des ingrédients extérieurs à la liste d'ingrédients soient mentionnés dans le corps de la recette. C'est notamment le cas à la fin des recettes, avec des suggestions d'accompagnement. Par exemple, une recette de tartiflette qui se terminerait par « Accompagnez préférablement d'une salade, et si vous souhaitez prendre un dessert, évitez les gâteaux ou yaourts, et privilégiez un fruit frais », engendrerait la détection de nombreux ingrédients qui n'apparaissent pas dans la liste d'ingrédients fournie par la recette, ce qui ferait baisser la précision de manière significative.

2. Corrélation entre complexités et niveau de difficulté

A partir des sorties des calculs de complexité en temps et en espace, nous avons cherché une corrélation entre complexité en temps, complexité en espace, et niveau de difficulté (indiqué dans chaque recette). Nous avons pour cela calculé les coefficients de corrélation entre les différentes variables, grâce à la libraire Numpy [9]. Ces coefficients ont été rajoutés à la fin du tableau en sortie du programme de calcul des complexités :

complexité en temps / difficulté	0.19944694056186218
complexité en espace/ difficulté	0.2594726711240869
complexité en temps / complexité en espace	0.27939377539333937

Le coefficient de corrélation peut avoir des valeurs comprises entre -1 et 1. Plus le coefficient est proche de 1, pus les variables sont liées positivement, c'est-à-dire qu'elles évoluent dans la même direction. Inversement, plus le coefficient est proche de -1, plus les variables sont liées négativement, c'est-à-dire qu'elles évoluent en sens contraire. Enfin, plus le coefficient est proche de 0, moins les variables sont liées linéairement. Les 3 coefficients que nous avons calculés ont une valeur positive, ce qui indique que les variables évoluent plutôt dans le même sens. Cependant, les valeurs de ces coefficients sont plus proches de 0 que de 1, nous ne pouvons donc pas dire qu'il existe une corrélation forte entre les variables.

On peut tout de même noter qu'avec nos méthodes de calcul de complexités, il semblerait qu'il existe une plus forte corrélation entre complexité en espace et niveau de difficulté, qu'entre complexité en temps et niveau de difficulté, et une corrélation encore un peu plus importante entre complexité en temps et complexité en espace.

3. Analyse de la complexité en temps d'une fonction du programme

Pour analyser la complexité en temps de notre fonction d'annotation des ingrédients et des opérations culinaires, nous avons eu recours à la librairie Python Time dont la méthode perf_counter permet d'obtenir le temps d'exécution d'une fonction en calculant le temps écoulé entre deux appels. Chaque appel de cette méthode renvoie en effet la valeur d'un compteur de performance (i.e. une horloge avec une résolution maximale afin de calculer des écarts de la plus courte durée possible).

Nous avons ainsi calculé le temps d'exécution de la fonction pour chaque recette du corpus, en prenant en compte la longueur de la recette (en nombre de tokens) et avons généré un graphique avec Matplotlib [8] afin de visualiser le temps d'exécution en fonction de la longueur de la recette passée en paramètre (lorsque plusieurs recettes avaient le même nombre de tokens, nous avons calculé la moyenne des temps d'exécution). Ce graphique est représenté en figure 13.

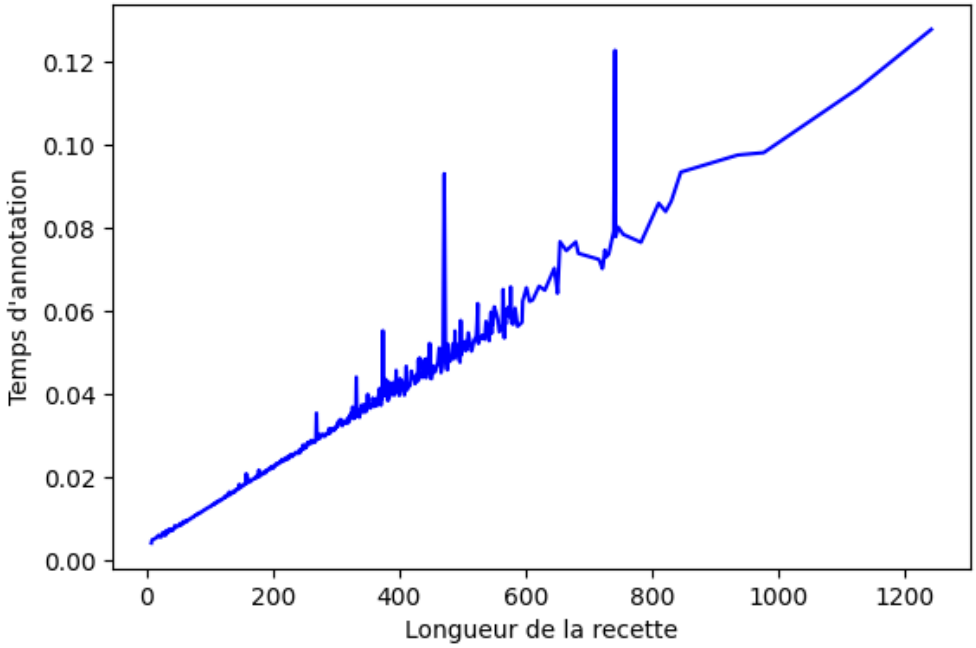


Figure 13 : Graphique du temps d'exécution de la fonction d'annotation en fonction de la longueur de la recette

On constate assez facilement que la complexité de notre fonction est linéaire (on peut l'exprimer avec la fonction y = 0.0001x), c'est-à-dire que le temps de traitement est directement proportionnel à la taille des données. Cela semble assez logique dans la mesure où la fonction annoter_recette appelle la fonction récursive _annoter_recette qui s'appelle elle-même autant de fois qu'il y a de tokens dans la préparation de la recette. La complexité linéaire se note O(n) où n est la taille des données (O étant la notation du grand O de Landau couramment utilisé pour les calculs de complexité d'un algorithme).

V. Conclusion

Malgré l'ampleur de la tâche et le peu de temps dont nous disposions pour l'accomplir, nous avons réussi à obtenir des résultats assez satisfaisants, que ce soit pour la tâche préalable d'annotation du corpus ou pour le calcul de complexité des recettes.

En effet, l'utilisation de lexiques pour la reconnaissance des ingrédients et des opérations culinaires nous a permis d'atteindre des scores de précision et de rappel plutôt honorables. Il faut cependant garder à l'esprit que les scores de rappel peuvent être légèrement surévalués puisque notre calcul du rappel ne se base que sur la présence ou l'absence d'un ingrédient attendu parmi les ingrédients annotés. De ce fait, si un des ingrédients de la liste prise comme référence est mentionné plusieurs fois dans le texte de la recette, d'éventuels faux négatifs ne seront pas détectés du moment qu'une seule de ces occurrences est trouvée. Pour obtenir une évaluation vraiment précise de notre travail, il faudrait avoir une annotation humaine de notre corpus et confronter l'annotation automatique à cette annotation humaine.

Concernant toujours l'annotation du corpus, plusieurs améliorations sont à envisager.

Tout d'abord, nous avons mis au point une solution plutôt concluante pour la coréférence dans le temps qui nous était imparti, mais cette solution demeure très imparfaite et ce sujet mérite d'être creusé : il faudrait notamment pouvoir traiter les anaphoriques (à quel ingrédient fait référence tel pronom personnel ?). D'autre part, nous avons analysé et annoté des tokens mais il serait probablement plus efficace et plus précis de travailler à l'échelle de syntagmes, même si le temps de traitement s'en trouverait allongé. Une étape supplémentaire de normalisation dans le prétraitement pourrait également permettre d'améliorer les résultats en corrigeant certaines fautes grossières ou coquilles facilement identifiables (comme par exemple *asssaisonnement* avec trois *s*).

Enfin, il serait intéressant d'essayer d'autres analyseurs morpho-syntaxiques et peut-être même d'en combiner plusieurs afin d'obtenir les meilleurs résultats possibles. En effet, SpaCy fournit des résultats imparfaits à tous les niveaux (POS-tagging, lemmatisation, analyse en dépendances), or il s'agit de fonctionnalités primordiales pour notre travail, ainsi que pour son évaluation. Ces erreurs ont donc un fort impact sur les performances mais aussi sur la fiabilité des résultats d'évaluation, ce qui est problématique. Certaines erreurs semblent communes à beaucoup d'étiqueteurs et propres au style de rédaction de recettes de cuisine (en effet Rahul Agarwal et Kevin Miller [2], qui utilisent le parseur de Stanford sur de l'anglais, rapportent aussi des problèmes pour étiqueter les nombreux verbes à l'impératif caractéristiques du style prescriptif des recettes) mais SpaCy fait également de nombreuses autres erreurs.

Pour le calcul de complexité en temps, un travail plus précis et minutieux sur les ressources utilisées permettrait d'obtenir des résultats plus cohérents. Nous pensons notamment à la conversion des quantités : nous aurions aimé mettre en place un système qui tienne compte du type d'ingrédient de sorte que nous ne puissions pas avoir, comme dans l'exemple déjà mentionné plus haut, 1.8 kg de poulet qui se transforme en 18 poulets. Nous n'avons malheureusement pas eu le temps nécessaire pour implémenter cela.

Pour conclure, nous avons obtenu des résultats satisfaisants mais nous disposons encore d'une grande marge d'amélioration.

VI. Bibliographie

1. Articles scientifiques

[1] Hamon, T., & Grabar, N. (2013, October). Extraction of ingredient names from recipes by combining linguistic annotations and CRF selection. In *Proceedings of the 5th international workshop on Multimedia for cooking & eating activities* (pp. 63-68).

[2] Agarwal, R., & Miller, K. (2011). Information Extraction from Recipes. *Department of Computer Science, Stanford University-2008*.

[3] Kim, S. D., Lee, Y. J., Cho, H. G., & Yoon, S. M. (2016). Complexity and similarity of recipes based on entropy measurement. *Indian Journal of Science and Technology*, *9*, 26.

[4] Silva, N., Ribeiro, D., & Ferreira, L. (2019, April). Information extraction from unstructured recipe data. In *Proceedings of the 2019 5th International Conference on Computer and Technology Applications* (pp. 165-168).

[5] Storby, J. (2016). Information extraction from text recipes in a web format.

2. Librairies Python

[6] Honnibal, M., & Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.*

[7] Richardson, L. (2007). Beautiful soup documentation. *April*.

[8] Hunter, J. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9(3), 90–95.*

[9] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del R\'io, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy and Warren Weckesser, Hameer Abbasi, & Christoph Gohlke and Travis E. Oliphant (2020). Array programming with NumPy. *Nature, 585(7825), 357–362.*