FoodCop - Food Composition Parser
Un parser de liste d'ingrédients

Tristan Salord (IRIT),
Marie-Benoit Magrini (INRAE, Agir), Guillaume Cabanac (IRIT, IUF)

Séminaire AliMining, 29 septembre 2025

Enjeux & Constats

- Analyser massivement la **composition** des produits agroalimentaires mis sur le marché :

  - *pour informer les politiques publiques alimentaires et de santé*,
  - *les consommateurs,*
  - *nourrir la recherche*

« (...) *there is lack of information in the scientific literature on type of ingredients used in packaged foods and their prevalence of use, especially in the US. This may be related to the lack of publicly available datasets containing information on these ingredients and in general, the ever-changing commercial food supply.* »
Ahuja et al., 2021, p.2

Problèmes

- Pas de règles uniformes et standardisées d'affichages des listes d'ingrédients :
  - Europe : Règlement n°1169/2011 (INCO) :

Article 18

Liste des ingrédients

1. La liste des ingrédients est assortie d'un intitulé ou précédée d'une mention appropriée «ingrédients» ou comportant ce terme. Elle comprend tous les ingrédients de la denrée alimentaire, dans l'ordre décroissant de leur importance pondérale au moment de leur mise en œuvre dans la fabrication de la denrée.

  - USA : Code de la régulation Fédérale (CFR) 101.4 « Food ; designation of ingredients » :

§ 101.4 Food; designation of ingredients.

(a)

(1) Ingredients required to be declared on the label or labeling of a food, including foods that comply with standards of identity, except those ingredients exempted by § 101.100, shall be listed by common or usual name in descending order of predominance by weight on either the principal display panel or the information panel in accordance with the provisions of § 101.2, except that ingredients in dietary supplements that are listed in the nutrition label in accordance with § 101.36 need not be repeated in the ingredient list. Paragraph (g) of this section describes the ingredient list on dietary supplement products.

(2) The descending order of predominance requirements of paragraph (a)(1) of this section do not apply to ingredients present in amounts of 2 percent or less by weight when a listing of these ingredients is placed at the end of the ingredient statement following an appropriate quantifying statement, e.g., "Contains __ percent or less of _____" or "Less than __ percent of _____." The blank percentage within the quantifying statement shall be filled in with a threshold level of 2 percent, or, if desired, 1.5 percent, 1.0 percent, or 0.5 percent, as appropriate. No ingredient to which the quantifying phrase applies may be present in an amount greater than the stated threshold.

Problèmes

- Traduction : **pas de grammaire/syntaxe commune** aux industriels pour l'affichage des listes d'ingrédients = Mise en échec des « parsers » usuels

  - Pas/peu d'outil existant :
    - ✓https://github.com/irockel/ingredients_parser
    - ✓https://github.com/q-m/food-ingredient-parser-ruby
    - ✓ ODALIM – Simple Decoding Ingredients
  - ... Mais outils :
    - non testés/benchmarkés,
    - parfois « destructifs »,
    - Orientés (par une langue, par un choix de type de données, ...)

FOODCOP

- Un **parser** de liste d'ingrédients **non-destructif** qui repose sur 4 étapes de traitement:

✓Réduction de la variabilité syntaxique à une forme commune - des listes imbriquées – par un jeu d'expressions régulières ;

```
def simpleformater(rawing):
        '''First normalisation of raw ingredient lists based on a simple set of regular expression. '''

        formatedchain = rawing
        rules=((('\[','('),('\{','('),('\]',')'),('\}',')'),(';',','))

        for rule in rules:
            formatedchain = re.sub(rule[0],rule[1],formatedchain)

        for k in extractedadds:
            if k in formatedchain:
                formatedchain = formatedchain.replace(k,extractedadds[k])

        cleaned = dictremover(extractedadds,formatedchain)
        return cleaned
```

```
def orphanremover(listofelements: list):
    '''Generic function that return a balanced bracketed list
    from an unbalanced one. Here Function track only bad closing bracket.
    It was designed for the addscategoryextract function'''
    newlist=listofelements
    opening=[]
    badpos=[]
    for rk,el in enumerate(newlist):
        if el == '(':
            opening.append(rk)
        elif el == ')':
            if len(opening) == 0:
                badpos.append(rk)
            else:
                opening.pop()

    for ix,rk in enumerate(badpos):
        newlist.pop(rk-ix)
    #return badpos
    return newlist
```

✓Normalisation orthographique par fuzzy-matching ;

```
def fuzzaddchecker(lexedinglist):
    ''' Function check if in lexed ingredient list element are additives categories
    returns a boolean'''

    presence = False
    for rk,exp in enumerate(lexedinglist):
        cond1=rk+1<len(lexedinglist) and lexedinglist[rk+1] in ['(',')']
        if re.search(':',exp):
            splitexp=exp.split(':')
            if fuzzysearch(splitexp[0],additivescatlist) != 0:
                presence = True
                break
        elif cond1:
            search=fuzzysearch(exp,additivescatlist)
            if search != 0:
                presence=True
                break
    return presence
```

```
def complexfuzzaddchecker(lexedinglist):
    ''' Useful variant of fuzzaddchecker. Used for complexe ingredient list.
    returns a boolean'''
    presence= False
    for part in lexedinglist:
        if fuzzaddchecker(part) is True:
            presence = True
            break
    return presence
```

FOODCOP

✓Extraction/sauvegarde des informations contextuelles aux ingrédients ;

```python
def commentextract(rawinglist):

    ''' Function extracts comments from raw ingredient list.
    Comment markings are normalised (*).
    Function returns two lists:
    + a cleaned ingredient list from comments,
    + list of the extracted comments.'''

    ### nested function called to normalised comments markings by
    # alternate signs

    def alternatereplacement(referencepoint: int, lookedchain: str, commentlist: list) -> str:
        alternatemarks=['°','#','\d=']
        #comlist = commentlist
        cleanedchain = lookedchain
        if referencepoint == 0:
            referencepoint += 1
        for sign in alternatemarks:
            for comment in commentlist:
                if re.search(sign, comment):
                    referencepoint += 1
                    prefix='*'*referencepoint+' '
                    commentlist.append((prefix+re.sub(sign,'',comment)).strip())
                    commentlist.pop(commentlist.index(comment))
                    cleanedchain=cleanedchain.replace(sign,(' '+prefix))

        return cleanedchain

    comments=[]
    uncomment=[]
```

```python
def propextract(lexedinglist: list) -> list:
    ''' Function to extract proportions from ingredient list.
    Function returns a list containing:
    + lexed ingredient list cleaned from any proportion
    + extracted proportions
    Proportion are marked by a $.'''

    complexprop=re.compile('(\d{1,2}(\.\d{1,2}){0,1}(kg|g|mg|µg-RE)(\/(100g|100g formula|100g prepared product|per\s{0,1}100g|100ml
    ppm=re.compile('(>|<){0,1}\d{1,4}(\.\d{1,4}|-\d{1,4}){0,1}\s{0,1}ppm')
    propsign='$'
    ''' Première situation: cas des pourcentages'''

    firstpass=[]
    secondpass=[]
    prop=[]
    count=1
    for ix,el in enumerate(lexedinglist):
        '''cherche si pourcentage présent'''
        if re.search('(\d{1,3}%)|\d{1,3}\.\d{1,3}%',el):
            match=re.search('(\d{1,3}%)|\d{1,3}\.\d{1,3}%',el).group()
            if ix == 0 and not el.startswith(match):
                newel=el.replace(match,propsign*count)
                prop.append(match)
                firstpass.append(newel)
                count+=1
            elif ix-1 >= 0 and lexedinglist[ix - 1] != '(':
                newel=el.replace(match,propsign*count)
                prop.append(match)
                firstpass.append(newel)
                count +=1
            elif ix-1 >= 0 and lexedinglist[ix - 1] == '(':
```

✓Réagencement sous forme de dictionnaire ;

```
{'ProdId':
        {'ProdId Nb.1':
            {'Ingredient Name': 'XXX', ' Ingredient level of appearance ': 'XXX',
          'Proportion of the ingredient': 'XXX', 'Comment related to the
          ingredient': 'XXX', 'Additive Category': 'XXX'},
        'ProdId. Nb.2':
            {…},
        …}
}
```

FOODCOP : exemple de sorties

Types of ingredient information – code names	Description
Ingredient Name – 'rawing'	Name of the ingredient as mentioned in the ingredient list.
Ingredient level of appearance – 'level'	Depth at which the ingredient appears in the ingredient list. For example, if the ingredient appears inside parentheses, (ingredient), it has a depth of 1. If the ingredient appears inside parentheses which are themselves inside parentheses, ((ingredient)), ingredient has a depth of 2, and so on.
Proportion of the ingredient – 'prop'	Proportion of the ingredient as mentioned in the ingredient list.
Comment related to the ingredient – 'comment'	Extra-information related to the ingredient. Only concerns information marked by a specific character sign.
Additive Category – 'additives'	General category or function of the additive mentioned in the ingredient list.

Nomenclature des clés des dictionnaires

Exemple de parsing d'une liste d'ingrédients

Raw Input:

```
'''tomato* (tomato pulp*, tomato concentrate*), water, rehydrated red kidney beans* (11%),
courgettes* (8%), carrots* (7%), onions* (6%), peppers* (6%), soy proteins* (5%), rehydrated
chickpeas*, cold extracted extra virgin olive oil*, spices*, sea salt, garlic*, basil*

*ingredients from organic farming'''
```

--- *** ---

Dictionary Output:


```
{'75_1': {'rawing': 'tomato', 'level': 0, 'comment': ['ingredients from organic farming']},
'75_2': {'rawing': 'tomato pulp', 'level': 1, 'comment': ['ingredients from organic
farming']}, '75_3': {'rawing': 'tomato concentrate', 'level': 1, 'comment': ['ingredients from
organic farming']}, '75_4': {'rawing': 'water', 'level': 0}, '75_5': {'rawing': 'rehydrated
red kidney beans', 'level': 0, 'prop': ['11%'], 'comment': ['ingredients from organic
farming']}, '75_6': {'rawing': 'courgettes', 'level': 0, 'prop': ['8%'], 'comment':
['ingredients from organic farming']}, '75_7': {'rawing': 'carrots', 'level': 0, 'prop':
['7%'], 'comment': ['ingredients from organic farming']}, '75_8': {'rawing': 'onions',
'level': 0, 'prop': ['6%'], 'comment': ['ingredients from organic farming']}, '75_9':
{'rawing': 'peppers', 'level': 0, 'prop': ['6%'], 'comment': ['ingredients from organic
farming']}, '75_10': {'rawing': 'soy proteins', 'level': 0, 'prop': ['5%'], 'comment':
['ingredients from organic farming']}, '75_11': {'rawing': 'rehydrated chickpeas', 'level': 0,
'comment': ['ingredients from organic farming']}, '75_12': {'rawing': 'cold extracted extra
virgin olive oil', 'level': 0, 'comment': ['ingredients from organic farming']}, '75_13':
{'rawing': 'spices', 'level': 0, 'comment': ['ingredients from organic farming']}, '75_14':
{'rawing': 'sea salt', 'level': 0}, '75_15': {'rawing': 'garlic', 'level': 0, 'comment':
['ingredients from organic farming']}, '75_16': {'rawing': 'basil', 'level': 0, 'comment':
['ingredients from organic farming']}}
```

Benchmarking

- Comment comparer évaluer la proximité de listes d'ingrédients quand ces listes peuvent être de taille et de complexité variable (Ruback et al., 2018)?

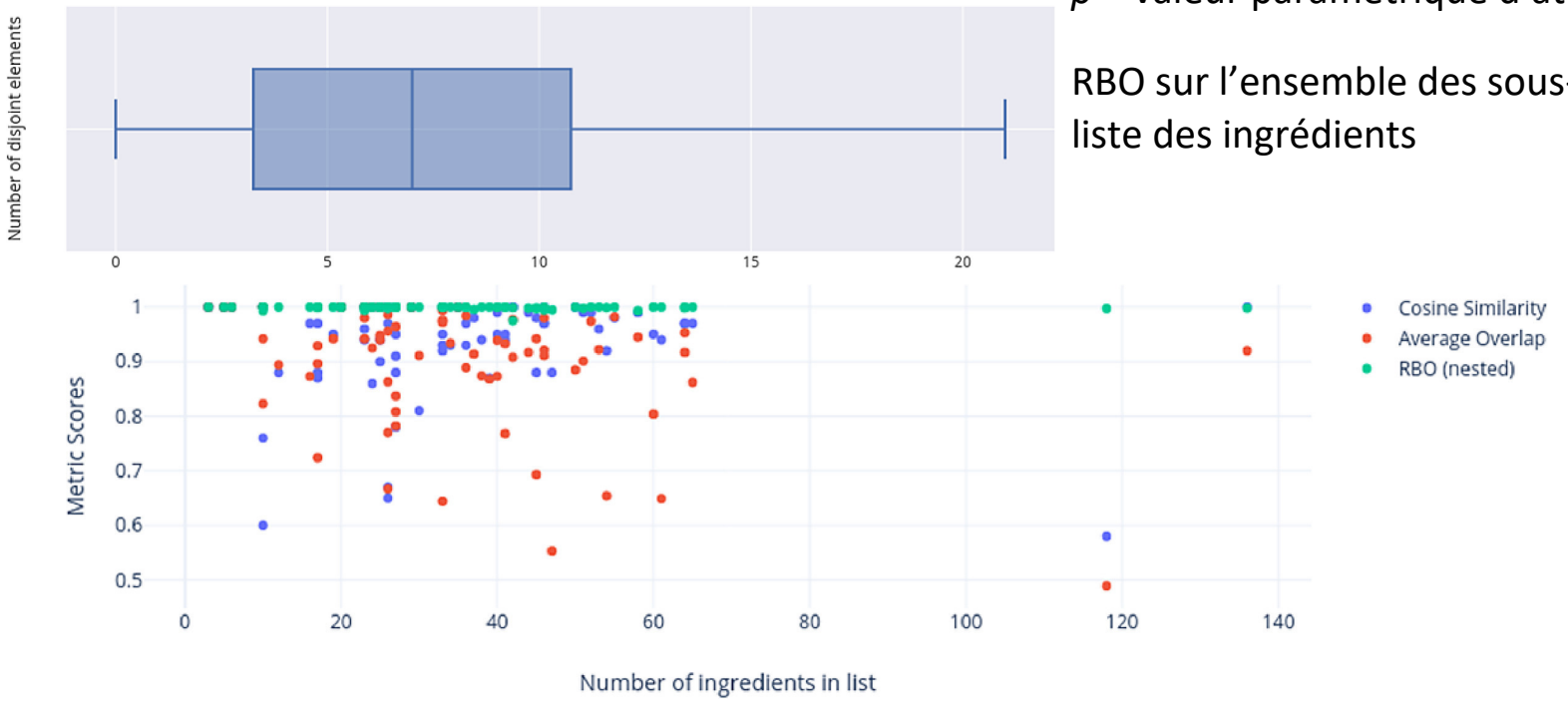
$$AO\,(L1, L2, k) = \frac{1}{k}\sum_{d=1}^{k} A_d$$

Dispersion of ingredient list differences between the two parsers

$$\mathrm{RBO}(S,T,p) = (1-p)\sum_{d=1}^{\infty} p^{d-1} \cdot A_d$$

p = valeur paramétrique d'attention

RBO sur l'ensemble des sous-listes de la liste des ingrédients



Bibliographie

L. Ruback, C. Lucchese, A. Mera Caraballo, G. García, M. Casanova, C. Renso, Computing entity semantic similarity by features ranking, 2018.

Salord, T., Magrini, M.-B., Lullien-Pellerin, V., Cabanac, G., Amiot, M.-J., Barron, C., Boire, A., Micard, V., & Weber, M. (2024). Crop diversity used in branded products with focus on legume species worldwide. npj Science of Food, 8(1), 68. ,https://doi.org/10.1038/s41538-024-00305-7

W. Webber, A. Moffat, J. Zobel, A similarity measure for indefinite rankings, ACM Trans. Inf. Syst. 28 (2010) 1–38,doi:10.1145/1852102.1852106