Performance_measurement

Pierre Massé

May 12, 2020

1 Mesure de la performance du modèle

L'objet de ce notebook est d'illustrer la méthodologie de mesure de la performance du modèle.

1.1 Préambule technique

[1]:
```
# setting up sys.path for relative imports
from pathlib import Path
import sys
project_root = str(Path(sys.path[0]).parents[1].absolute())
if project_root not in sys.path:
    sys.path.append(project_root)
```

[2]:
```
# imports and customization of diplay
import os
from functools import partial
import numpy as np
import pandas as pd
pd.options.display.min_rows = 6
pd.options.display.width=108
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.pipeline import Pipeline
from matplotlib import pyplot as plt

from src.pimest import ContentGetter
from src.pimest import PathGetter
from src.pimest import PDFContentParser
from src.pimest import BlockSplitter
from src.pimest import SimilaritySelector
from src.pimest import custom_accuracy
from src.pimest import text_sim_score
from src.pimest import text_similarity
from src.pimest import build_text_processor
```

1.2 Acquisition des données

On récupère les données manuellement étiquetées et on les intègre dans un dataframe

[3]:
```
ground_truth_df = pd.read_csv(Path('..') / '..' / 'ground_truth' / 'manually_labelled_ground_truth.csv',
                              sep=';',
                              encoding='latin-1',
                              index_col='uid')
ground_truth_uids = list(ground_truth_df.index)

acqui_pipe = Pipeline([('PathGetter', PathGetter(ground_truth_uids=ground_truth_uids,
                                                 train_set_path=Path('..') / '..' / 'ground_truth',
                                                 ground_truth_path=Path('..') / '..' / 'ground_truth',
                                                 )),
                       ('ContentGetter', ContentGetter(missing_file='to_nan')),
                       ('ContentParser', PDFContentParser(none_content='to_empty')),
```

```
                        ],
                        verbose=True)

texts_df = acqui_pipe.fit_transform(ground_truth_df)
texts_df
```

```
[Pipeline] … (step 1 of 3) Processing PathGetter, total=   0.1s
[Pipeline] … (step 2 of 3) Processing ContentGetter, total=   0.7s
Launching 8 processes.
[Pipeline] … (step 3 of 3) Processing ContentParser, total=  37.3s
```

[3]:

uid	designation	ingredients	path	content	text
a0492df6-9c76-4303-8813-65ec5ccbfa70	Concentré liquide Asian en bouteille 980 ml CHEF	Eau, maltodextrine, sel, arômes, sucre, arôme …	../../ground_truth/a0492df6-9c76-4303-8813-65e…	b'%PDF-1.5\r\n%\xb5\xb5\xb5\xb5\r\n1 0 obj\r\n…	Concentré Liquide Asian CHEF® \n\nBouteille de…
d183e914-db2f-4e2f-863a-a3b2d054c0b8	Pain burger curry 80 g CREATIV BURGER	Farine de blé T65, eau, levure, vinaigre de ci…	../../ground_truth/d183e914-db2f-4e2f-863a-a3b…	b'%PDF-1.5\r%\xe2\xe3\xcf\xd3\r\n4 0 obj\r<</L…	
ab48a1ed-7a3d-4686-bb6d-ab4f367cada8	Macaroni en sachet 500 g PANZANI	- 100% Semoule de BLE dur de qualité supérieur…	../../ground_truth/ab48a1ed-7a3d-4686-bb6d-ab4…	b'%PDF-1.4\n%\xc7\xec\x8f\xa2\n5 0 obj\n<</Len…	Direction Qualité \n\n \n\n \n\nPATES ALIMENTA…
…	…	…	…	…	…
e67341d8-350f-46f4-9154-4dbbb8035621	PRÉPARATION POUR CRÈME BRÛLÉE BIO 6L	Sucre roux de canne*° (64%), amidon de maïs*, …	../../ground_truth/e67341d8-350f-46f4-9154-4db…	b'%PDF-1.7\r\n%\xb5\xb5\xb5\xb5\r\n1 0 obj\r\n…	FICHE TECHNIQUE \n\nCREME BRÛLÉE 6L \n\nREF : …
a8f6f672-20ac-4ff8-a8f2-3bc4306c8df3	Céréales instantanées en poudre saveur caramel…	Farine 87,1 % (Blé (GLUTEN), Blé hydrolysé (GL…	../../ground_truth/a8f6f672-20ac-4ff8-a8f2-3bc…	b'%PDF-1.5\r\n%\xb5\xb5\xb5\xb5\r\n1 0 obj\r\n…	81 rue de Sans Souci - CS13754 - 69576 Limones…
0faad739-ea8c-4f03-b62e-51ee592a0546	FARINE DE BLÉ TYPE 45, 10KG	Farine de blé T45	../../ground_truth/0faad739-ea8c-4f03-b62e-51e…	b'%PDF-1.5\r\n%\xb5\xb5\xb5\xb5\r\n1 0 obj\r\n…	\n1050/10502066400 \n\n10502055300/1050202520…

[500 rows x 5 columns]

On splitte les textes en blocs de manière basique.

[4]:
```
def splitter(text):
    return(text.split('\n\n'))

split_transfo = BlockSplitter(splitter_func=splitter)
splitted_df = split_transfo.fit_transform(texts_df)
splitted_df
```

```
Launching 8 processes.
```

[4]:

uid	designation	ingredients	path	content	text	blocks
a0492df6-9c76-4303-8813-65ec5ccbfa70	Concentré liquide Asian en bouteille 980 ml CHEF	Eau, maltodextrine, sel, arômes, sucre, arôme …	../../ground_truth/a0492df6-9c76-4303-8813-65e…	b'%PDF-1.5\r\n%\xb5\xb5\xb5\xb5\r\n1 0 obj\r\n…	Concentré Liquide Asian CHEF® \n\nBouteille de…	[Concentré Liquide Asian CHEF® , Bouteille de …
d183e914-db2f-4e2f-863a-a3b2d054c0b8	Pain burger curry 80 g CREATIV BURGER	Farine de blé T65, eau, levure, vinaigre de ci…	../../ground_truth/d183e914-db2f-4e2f-863a-a3b…	b'%PDF-1.5\r%\xe2\xe3\xcf\xd3\r\n4 0 obj\r<</L…		[
ab48a1ed-7a3d-4686-bb6d-ab4f367cada8	Macaroni en sachet 500 g PANZANI	- 100% Semoule de BLE dur de qualité supérieur…	../../ground_truth/ab48a1ed-7a3d-4686-bb6d-ab4…	b'%PDF-1.4\n%\xc7\xec\x8f\xa2\n5 0 obj\n<</Len…	Direction Qualité \n\n \n\n \n\nPATES ALIMENTA…	[Direction Qualité , , , PATES ALIMENTAIRES …
…	…	…	…	…	…	…
e67341d8-350f-46f4-9154-4dbbb8035621	PRÉPARATION POUR CRÈME BRÛLÉE BIO 6L	Sucre roux de canne*° (64%), amidon de maïs*, …	../../ground_truth/e67341d8-350f-46f4-9154-4db…	b'%PDF-1.7\r\n%\xb5\xb5\xb5\xb5\r\n1 0 obj\r\n…	FICHE TECHNIQUE \n\nCREME BRÛLÉE 6L \n\nREF : …	
a8f6f672-20ac-4ff8-a8f2-3bc4306c8df3	Céréales instantanées en poudre saveur caramel…	Farine 87,1 % (Blé (GLUTEN), Blé hydrolysé (GL…	../../ground_truth/a8f6f672-20ac-4ff8-a8f2-3bc…	b'%PDF-1.5\r\n%\xb5\xb5\xb5\xb5\r\n1 0 obj\r\n…	81 rue de Sans Souci - CS13754 - 69576 Limones…	
0faad739-ea8c-4f03-b62e-51ee592a0546	FARINE DE BLÉ TYPE 45, 10KG	Farine de blé T45	../../ground_truth/0faad739-ea8c-4f03-b62e-51e…	b'%PDF-1.5\r\n%\xb5\xb5\xb5\xb5\r\n1 0 obj\r\n…	\n1050/10502066400 \n\n10502055300/1050202520…	

```
e67341d8-350f-46f4-9154-4dbbb8035621  [FICHE TECHNIQUE , CREME BRÛLÉE 6L , REF : NAP…
a8f6f672-20ac-4ff8-a8f2-3bc4306c8df3  [81 rue de Sans Souci - CS13754 - 69576 Limone…
0faad739-ea8c-4f03-b62e-51ee592a0546  [ \n1050/10502066400 , 10502055300/10502025200…

[500 rows x 6 columns]
```

1.3 Train/Test split, entraînement et tranformation

On effectue classiquement les étapes de train/test split, on entraîne le modèle sur le set d'entraînement et on le lance sur le set de test.

[5]:
```
train, test = train_test_split(splitted_df, train_size=400, random_state=42)
model = SimilaritySelector(similarity='projection')
model.fit(train['blocks'], train['ingredients'])
predicted = pd.Series(model.predict(test['blocks']),
                      index=test.index,
                      name='predicted'
                     )
predicted = pd.concat([test['ingredients'], predicted], axis=1)
predicted
```

[5]:
```
                                                                ingredients  \
uid
2892dd68-e3a6-474c-b543-3ebfd3490658        Café instantané, café torrefié moulu (3%).
a57c1561-b88e-4694-8bd8-55623f2afa17                                Lentilles blondes
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5       Poire 99,9%, antioxydant: acide ascorbique.
…                                                                                 …
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3                  Jus d'orange à base de concentré
c33aa83e-a502-4339-a8e0-c56db2e59e69  Farine de BLÉ, sucre, huile de colza,, cacao m…
54f40033-f9cf-411c-81a5-11974f6715aa  Piment rouge fort équeuté* (85%), cumin, ail m…

                                                                           predicted
uid
2892dd68-e3a6-474c-b543-3ebfd3490658  - NESTLÉ a un système de management de la qual…
a57c1561-b88e-4694-8bd8-55623f2afa17  Cette fiche technique n'a pas de valeur contra…
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5  Ce produit est une purée de fruits obtenue à p…
…                                                                                 …
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3   \n \nVALEURS NUTRITIONNELLES pour 100mL / NUT…
c33aa83e-a502-4339-a8e0-c56db2e59e69  Ingrédients : Farine de BLÉ, sucre, huile de c…
54f40033-f9cf-411c-81a5-11974f6715aa  A) Ingrédients : \n \nPiment  rouge  fort  équ…

[100 rows x 2 columns]
```

1.4 Mesure de la performance : Précision

1.4.1 Approche naïve

Dans cette première version, on calculera une précision brute, où seuls les strings parfaitement identiques sont considérés comme ok.

[6]:
```
predicted['result'] = (predicted['ingredients'].fillna('') == predicted['predicted'].fillna(''))
predicted['result'].value_counts()
```

[6]:
```
False    99
True      1
Name: result, dtype: int64
```

On a une précision très faible, 1%. L'unique liste d'ingrédients du set de test correctement prédite est la suivante :

[7]:
```
print(predicted[predicted['result']].iloc[0, 0])
```

```
Sirop de glucose, sucre, eau, stabilisants (E440i, E440ii, E415), acidifiants (E330, E450i), conversateur
(E202).
```

1.4.2 Cross-validation de l'approche naïve

Pour avoir une vision plus précise de la performance du modèle, on peut effectuer une cross-validation sur le set d'entraînement.

On commence par définir une fonction de scoring, qui pourra être appelée par la fonction standard de cross-validation de scikit-learn. Comme précédemment, il s'agit d'une fonction d'accuracy basique :

[8]:
```
def accuracy_scorer(estim, X, y):
    y_pred = estim.predict(X)
    return((y_pred == y).mean())
```

On retrouve évidemment le même score que précédemment lorsqu'on utilise cette fonction sur le set de test :

[9]:
```
accuracy_scorer(model, test.reset_index()['blocks'], test.reset_index()['ingredients'])
```

[9]:
```
0.01
```

Si on lance la cross-validation avec les paramètres par défaut (cv=5), on obtient le résultat suivant :

[10]:
```
X = splitted_df.reset_index()['blocks'].copy()
y = splitted_df.reset_index()['ingredients'].copy()

cross_val = cross_validate(model,
                           X=X,
                           y=y,
                           scoring=accuracy_scorer,
                          )
print(f'Strict accuracy yields a result of {np.mean(cross_val["test_score"]):.2%} +/-{np.
↪std(cross_val["test_score"]):.2%}')
print(cross_val['test_score'])
```

```
Strict accuracy yields a result of 2.20% +/-0.75%
[0.03 0.03 0.02 0.01 0.02]
```

On voit que sur chacun des 5 folds (validation sur 400 produits), l'accuracy varie entre 1 et 3%.

Si on trace l'accuracy et la standard deviation pour plusieurs valeurs de cv, on obtient les résultats suivants :

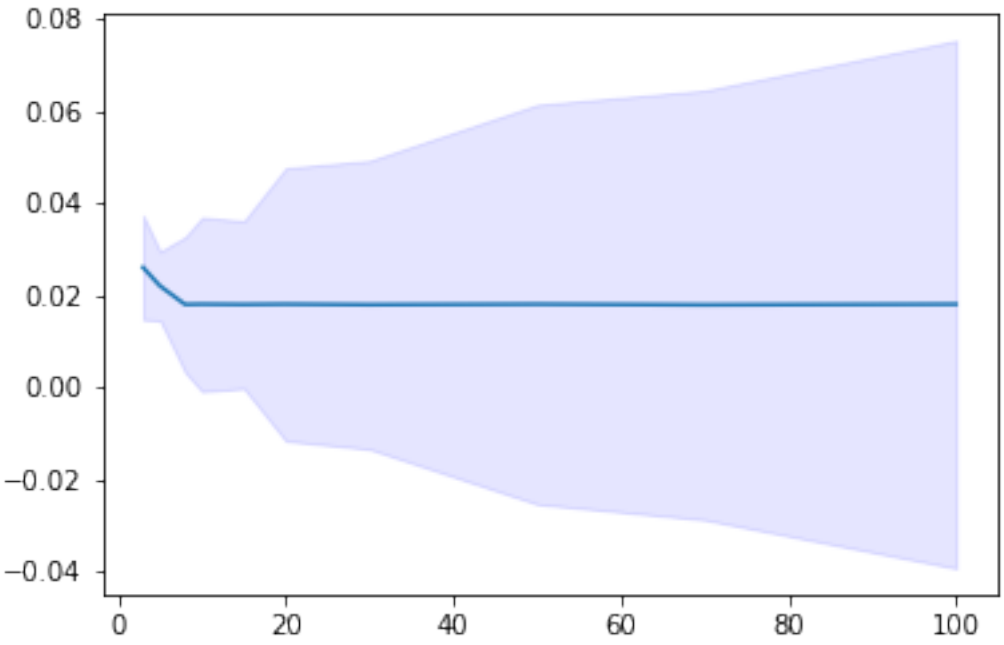
[11]:
```
x = [3, 5, 8, 10, 15, 20, 30, 50, 70, 100]
mean = np.array([])
std = np.array([])
for n_cv in x:
    cross_val = cross_validate(model,
                               X=splitted_df['blocks'],
                               y=splitted_df['ingredients'],
                               scoring=accuracy_scorer,
                               cv=n_cv,
                              )
    mean = np.append(mean, [np.mean(cross_val['test_score'])], axis=0)
    std = np.append(std, [np.std(cross_val['test_score'])], axis=0)

print('mean:', mean, '\nstandard dev:', std)
```

```
mean: [0.02598418 0.022      0.01795315 0.018      0.01794415 0.018
 0.01789216 0.018      0.01785714 0.018     ]
standard dev: [0.01126571 0.00748331 0.01470225 0.01886796 0.01816919 0.0295973
 0.03127858 0.04331282 0.04656573 0.05723635]
```

[12]:
```
fig, ax = plt.subplots()
ax.plot(x, mean)
ax.fill_between(x, (mean - std), (mean + std), color='b', alpha=.1)
```

[12]:
```
<matplotlib.collections.PolyCollection at 0x7f5005cccd90>
```



Il apparaît que l'accuracy se situe aux alentours de 2%, avec un écart type important si on le compare à cette accuracy.

[13]:
```
cross_val = cross_validate(model,
                           X=splitted_df['blocks'],
                           y=splitted_df['ingredients'],
                           scoring=accuracy_scorer,
                           cv=10,
                          )
print(f'Strict accuracy yields a result of {np.mean(cross_val["test_score"]):.2%} +/-{np.
↪std(cross_val["test_score"]):.2%}')
print(cross_val['test_score'])
```

```
Strict accuracy yields a result of 1.80% +/-1.89%
[0.04 0.   0.   0.06 0.02 0.02 0.   0.02 0.   0.02]
```

1.4.3 Ajout d'une étape de text-postprocessing

On utilise la fonction `custom_accuracy` définie dans le module pimest pour calculer l'accuracy avec du text processing. Elle prend en paramètre les mêmes attributs que le `CountVectorizer` de scikit-learn, en plus d'un attribut "tokenize" qui va tokenizer le résultat (pour prise en compte des whitespace et de la ponctuation).

[14]:
```
custom_accuracy(model,
                test['blocks'].fillna(''),
                test['ingredients'].fillna(''),
                tokenize=True,
                strip_accents='unicode',
                lowercase=True,
               )
```

[14]: 0.14

L'accuracy est maintenant estimée à 14% (vs. 1%) sur le set de test, après entraînenemt sur le set d'entraînement.

On peut manuellement inspecter les blocks identique, en reproduisant le comportement de la fonction d'accuracy :

[15]:
```
def text_processor(text, **kwargs):
    unused_model = CountVectorizer(**kwargs)
```

```
    prepro = unused_model.build_preprocessor()
    token = unused_model.build_tokenizer()
    return(' '.join(token(prepro(text))))

partial_processor = partial(text_processor, strip_accents='unicode', lowercase=True)
```

[16]:
```
prediction = model.predict(test['blocks'].fillna('')).rename('predicted')
processed_prediction = prediction.apply(partial_processor)
processed_prediction.head(3)
```

[16]:
```
uid
2892dd68-e3a6-474c-b543-3ebfd3490658    nestle un systeme de management de la qualite …
a57c1561-b88e-4694-8bd8-55623f2afa17    cette fiche technique pas de valeur contractue…
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5    ce produit est une puree de fruits obtenue par…
Name: predicted, dtype: object
```

[17]:
```
processed_ground_truth = test['ingredients'].fillna('').apply(partial_processor)
processed_ground_truth.head(3)
```

[17]:
```
uid
2892dd68-e3a6-474c-b543-3ebfd3490658      cafe instantane cafe torrefie moulu
a57c1561-b88e-4694-8bd8-55623f2afa17                          lentilles blondes
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5    poire 99 antioxydant acide ascorbique
Name: ingredients, dtype: object
```

[18]:
```
corrects = test.join(prediction).loc[processed_prediction == processed_ground_truth , ['ingredients',␣
↪'predicted']]
corrects
```

[18]:
```
                                                                                ingredients  \
uid
345591f4-d887-4ddc-bb40-21337fa9269d  Gésier de dinde émincé 50%, graisse de canard …
13980d31-9002-457d-8d49-b451f08f473c  Edulcorants sorbitol, isomalt, sirop de maltit…
c3b6b4df-e586-4f10-8e58-15fbf0816acb  mini poivrons jaunes, eau, sucre, sel, affermi…
0481d91b-9653-42e7-b525-9dc9b87b06f2  Farine de BLE, huile de colza non hydrogénée, …
484ac00a-a670-46a9-a9c4-5114174d9e3b  Pommes de terre 59,5 % - Céleris 40 % - Amidon…
49b11281-34ea-44b0-a11c-4ae21d4c58e3                                                NaN
d59d96cb-0230-4090-8220-78ce8496fd91  Amidon de maïs* - Lait écrémé* - Sel - Fécule …
b8cbe6f9-71d4-4e51-a169-1c163d49a561  Farine de FROMENT, poudre de LACTOSERUM, sucre…
a0492df6-9c76-4303-8813-65ec5ccbfa70  Eau, maltodextrine, sel, arômes, sucre, arôme …
09e45b38-4da1-4eb5-888a-3ebd437a2291  OEUFS, farine de BLE, sucre, amidon de BLE, st…
4f83306f-66de-4545-9b12-7790b57b61ae  Sirop de glucose, sucre, eau, stabilisants (E4…
5cee689e-6fb1-493c-b232-1d8fb1f88a57  Flageolets verts. Jus : eau, sel, affermissant…
63968dc3-6e7c-4056-bd53-820c6cc925be  Carottes, eau, sucre, sel, vinaigre d'alcool, …
dc536305-82fd-4afe-a472-5056ca0e21ea  Légumes 43,2 % (pomme de terre, oignon, carott…

                                                                                predicted
uid
345591f4-d887-4ddc-bb40-21337fa9269d  Gésier de dinde émincé 50%,  graisse de canard…
13980d31-9002-457d-8d49-b451f08f473c  Edulcorants sorbitol, isomalt, sirop de maltit…
c3b6b4df-e586-4f10-8e58-15fbf0816acb  mini poivrons jaunes, eau, sucre, sel, affermi…
0481d91b-9653-42e7-b525-9dc9b87b06f2  Farine de BLE, huile de colza non hydrogénée, …
484ac00a-a670-46a9-a9c4-5114174d9e3b  Pommes de terre 59,5 % - Céleris 40 % - Amidon…
49b11281-34ea-44b0-a11c-4ae21d4c58e3

d59d96cb-0230-4090-8220-78ce8496fd91  Amidon de maïs* - Lait écrémé* - Sel - Fécule …
b8cbe6f9-71d4-4e51-a169-1c163d49a561  Farine de FROMENT, poudre de LACTOSERUM, sucre…
a0492df6-9c76-4303-8813-65ec5ccbfa70  Eau, maltodextrine, sel, arômes, sucre, arôme …
09e45b38-4da1-4eb5-888a-3ebd437a2291  OEUFS, farine de BLE, sucre, amidon de BLE, st…
4f83306f-66de-4545-9b12-7790b57b61ae  Sirop de glucose, sucre, eau, stabilisants (E4…
5cee689e-6fb1-493c-b232-1d8fb1f88a57  Flageolets verts. Jus : eau, sel, affermissant…
63968dc3-6e7c-4056-bd53-820c6cc925be  Carottes, eau, sucre, sel, vinaigre d’alcool, …
dc536305-82fd-4afe-a472-5056ca0e21ea  Légumes 43,2 % (pomme de terre, oignon, carott…
```

[19]:
```
with pd.option_context("max_colwidth", 100000):
    tex_str = (
    corrects.replace(r'^\s*$', np.nan, regex=True)
            .to_latex(index=False,
                      index_names=False,
                      column_format='p{7cm}p{7cm}',
                      na_rep='<rien>',
                      longtable=False,
                      header=["Liste d'ingrédients cible", "Liste d'ingrédients prédite"],
                      # label='tbl:GT_postprocessed_corrects',
                      # caption="Prédictions identifiées comme correctes après postprocessing",
                  )
             .replace(r'\textbackslash n', r' \newline ')
             .replace(r'\\', r'\\ \hline')
    )

with open(Path('..') / 'tbls' / 'GT_postprocessed_corrects.tex', 'w') as file:
    file.write(tex_str)
```

1.4.4 Cross-validation de l'approche avec text processing

On fait tourner une cross-validation sur l'ensemble du dataset. On définit d'abord la fonction qui va permettre de calculer le score avec l'ensemble des fonctionnalités de text processing : - retrait des accents - remplacement des whitespaces par des espaces simples - retrait de la ponctuation - mise en minuscule

[20]:
```
processed_accuracy = partial(custom_accuracy,
                             tokenize=True,
                             strip_accents='unicode',
                             lowercase=True,
                            )
cross_val = cross_validate(model,
                           X=splitted_df['blocks'].fillna(''),
                           y=splitted_df['ingredients'].fillna(''),
                           scoring=processed_accuracy,
                          )
print(f'Processed accuracy yields a result of {np.mean(cross_val["test_score"]):.2%} +/-{np.
↪std(cross_val["test_score"]):.2%}')
print(cross_val['test_score'])
```

```
Processed accuracy yields a result of 17.00% +/-2.45%
[0.2  0.16 0.17 0.13 0.19]
```

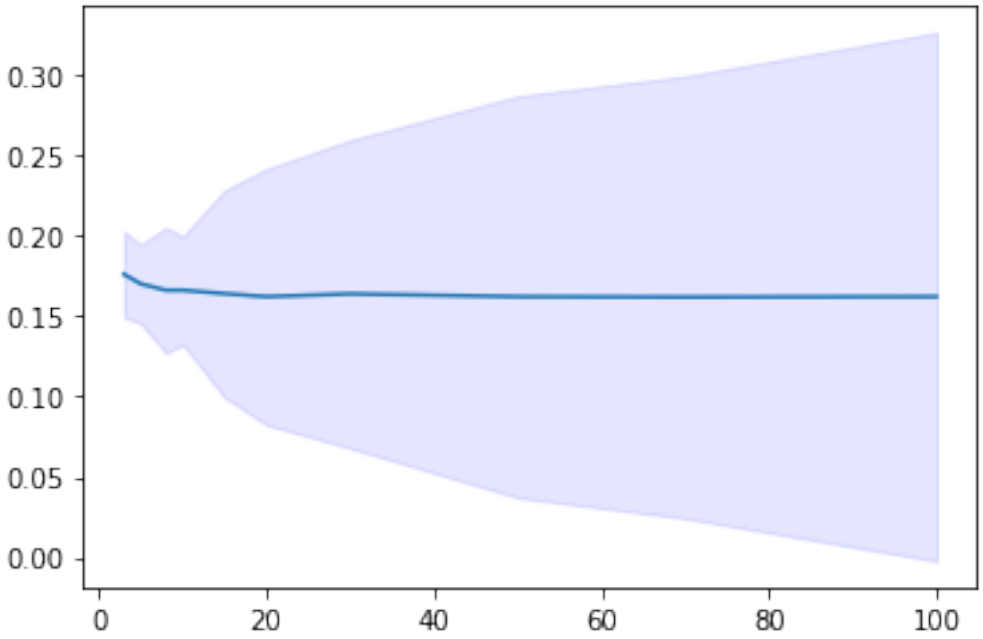
[21]:
```
x = [3, 5, 8, 10, 15, 20, 30, 50, 70, 100]
mean = np.array([])
std = np.array([])
for n_cv in x:
    cross_val = cross_validate(model,
                               X=splitted_df['blocks'].fillna(''),
                               y=splitted_df['ingredients'].fillna(''),
                               scoring=processed_accuracy,
                               cv=n_cv,
                              )
    mean = np.append(mean, [np.mean(cross_val['test_score'])], axis=0)
    std = np.append(std, [np.std(cross_val['test_score'])], axis=0)

print('mean:', mean, '\nstandard dev:', std)
```

```
mean: [0.17593728 0.17       0.16599462 0.166      0.16387403 0.162
 0.16372549 0.162      0.16173469 0.162     ]
standard dev: [0.02658629 0.0244949  0.03903199 0.03352611 0.06445016 0.07947327
 0.09572828 0.12472369 0.13735865 0.16418282]
```

[22]:
```
fig, ax = plt.subplots()
ax.plot(x, mean)
ax.fill_between(x, (mean - std), (mean + std), color='b', alpha=.1)
```

[22]: <matplotlib.collections.PolyCollection at 0x7f5005d50760>



[23]:
```
cross_val = cross_validate(model,
                           X=splitted_df['blocks'].fillna(''),
                           y=splitted_df['ingredients'].fillna(''),
                           scoring=processed_accuracy,
                           cv=10,
                          )
print(f'Processed accuracy yields a result of {np.mean(cross_val["test_score"]):.2%} +/-{np.
 ↪std(cross_val["test_score"]):.2%}')
print(cross_val['test_score'])
```

```
Processed accuracy yields a result of 16.60% +/-3.35%
[0.2  0.18 0.18 0.14 0.12 0.22 0.14 0.12 0.16 0.2 ]
```

1.5 Mesure de la performance : Similarité

1.5.1 Mesures

On peut également mesurer la similarité plutôt qu'uniquement l'accuracy. Cela permet de valoriser les textes qui "ressemblent" aux listes d'ingrédients cibles plutôt que les compter comme des erreurs.

[24]:
```
similarity_kinds = ['levenshtein',
                    'damerau-levenshtein',
                    'jaro',
                    'jaro-winkler',
                   ]

sim_dict = dict()

for similarity in similarity_kinds:
    sim = text_sim_score(model,
                         test['blocks'].fillna(''),
                         test['ingredients'].fillna(''),
                         similarity=similarity,
                        )
    sim_dict[similarity] = f'{sim:.2%}'
```

```
    print(f'Similarity with {similarity} similarity is {sim:.2%}')
```

```
Similarity with levenshtein similarity is 48.86%
Similarity with damerau-levenshtein similarity is 48.86%
Similarity with jaro similarity is 63.56%
Similarity with jaro-winkler similarity is 65.67%
```

Les similarités de Levenshtein et Damerau-Levenshtein donnent des résultats identiques, à presque 50% de similarité moyenne. Celles basées sur Jaro tournent aux alentours de 65%, comme on s'y attendait dans la mesure où elle est très "indulgente" sur les textes longs.

Si on effectue des cross validations sur chacune de ces distances sur le dataset complet, on obtient :

[25]:
```
similarities = {similarity: partial(text_sim_score, similarity=similarity) for similarity in␣
↪similarity_kinds}

cross_vals = dict()

for similarity in similarity_kinds:
    cross_vals[similarity] = cross_validate(model,
                                            splitted_df['blocks'].fillna(''),
                                            splitted_df['ingredients'].fillna(''),
                                            scoring=similarities[similarity],
                                            cv=10,
                                            )
for similarity in similarity_kinds:
    print(f'Model evaluated with {similarity} similarity a result of '
          f'{np.mean(cross_vals[similarity]["test_score"]):.2%} '
          f'+/-{np.std(cross_vals[similarity]["test_score"]):.2%}')
```

```
Model evaluated with levenshtein similarity a result of 49.79% +/-3.73%
Model evaluated with damerau-levenshtein similarity a result of 49.80% +/-3.72%
Model evaluated with jaro similarity a result of 62.78% +/-3.28%
Model evaluated with jaro-winkler similarity a result of 64.40% +/-3.50%
```

On transforme en tableau latex pour insertion dans le rapport.

[26]:
```
result_strings = dict()

for similarity in similarity_kinds:
    result_strings[similarity] = {'train/test set': sim_dict[similarity],
                                  'cross validation': f'{np.mean(cross_vals[similarity]["test_score"]):.2%}␣
↪'
                                                      f'+/-{np.std(cross_vals[similarity]["test_score"]):.
↪2%}'}
result_df = pd.DataFrame(result_strings).T
print(result_strings)
labs = {'levenshtein': 'Levenshtein',
        'damerau-levenshtein': 'Damerau-Levenshtein',
        'jaro': 'Jaro',
        'jaro-winkler': 'Jaro-Winkler',
       }
(result_df.rename(labs)
          .to_latex(Path('..') / 'tbls' / 'similarities_result.tex',
                    column_format='lcc',
                    bold_rows=True,
                   )
)
```

```
{'levenshtein': {'train/test set': '48.86%', 'cross validation': '49.79% +/-3.73%'}, 'damerau-levenshtein':
{'train/test set': '48.86%', 'cross validation': '49.80% +/-3.72%'}, 'jaro': {'train/test set': '63.56%',
'cross validation': '62.78% +/-3.28%'}, 'jaro-winkler': {'train/test set': '65.67%', 'cross validation':
'64.40% +/-3.50%'}}
```

1.5.2 Illustration

On illustre les différents niveaux de similarité sur le set de test après entrainement sur le set d'entrainement.

[33]:
```
# building the dataframe
y_pred = model.predict(test['blocks']).rename('predicted')
comp_df = pd.concat([test['ingredients'].fillna(''), y_pred], axis=1)
processed_df = comp_df.applymap(build_text_processor())

# computing similarities and ranks
sim_funcs = {sim: partial(text_similarity, similarity=sim) for sim in similarity_kinds}
for sim in similarity_kinds:
    processed_df[sim] = processed_df.apply(lambda x: sim_funcs[sim](x['ingredients'], x['predicted']),␣
↪axis=1)
    processed_df[sim + '_rank'] = processed_df[sim].rank(axis=0, method='first', ascending=False)
```

[34]:
```
(processed_df.join(comp_df, lsuffix='_')
             .sort_values(['levenshtein'], ascending=False)
             .loc[(processed_df['ingredients'] != '') & (processed_df['predicted'] != '')]
)
```

[34]:

uid	ingredients_	predicted_	levenshtein	damerau-levenshtein	jaro	jaro-winkler	ingredients	predicted
b8cbe6f9-71d4-4e51-a169-1c163d49a561	farine de froment poudre de lactoserum sucre p…	farine de froment poudre de lactoserum sucre p…	1.000000	1.000000	1.000000	1.000000	Farine de FROMENT, poudre de LACTOSERUM, sucre…	Farine de FROMENT, poudre de LACTOSERUM, sucre…
dc536305-82fd-4afe-a472-5056ca0e21ea	legumes 43 pomme de terre oignon carotte tomat…	legumes 43 pomme de terre oignon carotte tomat…	1.000000	1.000000	1.000000	1.000000	Légumes 43,2 % (pomme de terre, oignon, carott…	Légumes 43,2 % (pomme de terre, oignon, carott…
63968dc3-6e7c-4056-bd53-820c6cc925be	carottes eau sucre sel vinaigre alcool acidifi…	carottes eau sucre sel vinaigre alcool acidifi…	1.000000	1.000000	1.000000	1.000000	Carottes, eau, sucre, sel, vinaigre d'alcool, …	Carottes, eau, sucre, sel, vinaigre d’alcool, …
…	…	…	…	…	…	…	…	…
bb77c7b0-9c63-4869-9ab1-823ba1158f53	tilleul 100	dans le cadre des recommandations de sante pub…	0.019185	0.019185	0.259974	0.259974	Tilleul (100%).	Dans le cadre des recommandations de santé pub…
6267b9f8-2529-4bc6-ba4b-26760f0522b3	eau gazeifiee colorant e150d acidifiants acide…	coca cola light mini 150 mlean5449000239808mar…	0.012461	0.012461	0.506069	0.506069	eau gazéifiée\ncolorant : E150d\nacidifiants :…	CocaCola Light mini 8 x 150 mlEAN544900023980…
2ca5dc9e-8058-499a-affe-3ec9c06d55b7	100 arabica	gammemarqueargumentation commercialepreparatio…	0.011411	0.011411	0.372595	0.372595	100% Arabica	

```
2ca5dc9e-8058-499a-affe-3ec9c06d55b7  gammemarqueargumentation commercialepreparatio…

[90 rows x 8 columns]
```

[67]:
```
comp_df
```

[67]:
```
                                                                    ingredients  \
uid
2892dd68-e3a6-474c-b543-3ebfd3490658        Café instantané, café torrefié moulu (3%).
a57c1561-b88e-4694-8bd8-55623f2afa17                                Lentilles blondes
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5       Poire 99,9%, antioxydant: acide ascorbique.
…                                                                                  …
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3                  Jus d'orange à base de concentré
c33aa83e-a502-4339-a8e0-c56db2e59e69  Farine de BLÉ, sucre, huile de colza,, cacao m…
54f40033-f9cf-411c-81a5-11974f6715aa  Piment rouge fort équeuté* (85%), cumin, ail m…

                                                                    predicted
uid
2892dd68-e3a6-474c-b543-3ebfd3490658  - NESTLÉ a un système de management de la qual…
a57c1561-b88e-4694-8bd8-55623f2afa17  Cette fiche technique n’a pas de valeur contra…
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5  Ce produit est une purée de fruits obtenue à p…
…                                                                                  …
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3   \n \nVALEURS NUTRITIONNELLES pour 100mL / NUT…
c33aa83e-a502-4339-a8e0-c56db2e59e69  Ingrédients : Farine de BLÉ, sucre, huile de c…
54f40033-f9cf-411c-81a5-11974f6715aa  A) Ingrédients : \n \nPiment  rouge  fort  équ…

[100 rows x 2 columns]
```

[79]:
```
# outputing to latex
with pd.option_context("max_colwidth", 100000):
    tex_str = (processed_df.sort_values('levenshtein_rank')
                           .loc[(processed_df['ingredients'] != '') &
                                (processed_df['predicted'] != '') &
                                (processed_df['predicted'].apply(len) <=300)]
                           .join(comp_df, lsuffix='_')
                           .iloc[np.r_[0:3, 50:53, -3:0]]
                           .to_latex(columns=['ingredients', 'predicted', 'levenshtein', 'levenshtein_rank',
                                              'damerau-levenshtein', 'damerau-levenshtein_rank', 'jaro',
                                              'jaro_rank', 'jaro-winkler', 'jaro-winkler_rank'],
                                     index=False,
                                     column_format='p{5cm}p{5cm}cccccccc',
                                     formatters={'levenshtein': lambda x: f'{x:.2%}',
                                                 'levenshtein_rank': lambda x: f'{x:1.0f}',
                                                 'damerau-levenshtein': lambda x: f'{x:.2%}',
                                                 'damerau-levenshtein_rank': lambda x: f'{x:1.0f}',
                                                 'jaro': lambda x: f'{x:.2%}',
                                                 'jaro_rank': lambda x: f'{x:1.0f}',
                                                 'jaro-winkler': lambda x: f'{x:.2%}',
                                                 'jaro-winkler_rank': lambda x: f'{x:1.0f}'},
                                     header=["Listes d'ingrédients cibles", "Listes d'ingrédients prédites",
                                             'Lev', 'rang', 'Dam-Lev', 'rang', 'Jaro', 'rang', 'Jaro-Win',␣
↪'rang'],
                                     na_rep = '<rien>',
                                    )
    ).replace(r'\textbackslash n', r' \newline ').replace(r'\\', r'\\ \hline')

with open(Path('..') / 'tbls' / 'similarity_illustration.tex' , 'w') as file:
    file.write(tex_str)
```

[77]:
```
(processed_df.sort_values('levenshtein_rank')
             .loc[(processed_df['ingredients'] != '') & (processed_df['predicted'] != '')
                  & (processed_df['predicted'].apply(len) <=300)]
             .join(comp_df, lsuffix='_')
             .iloc[np.r_[0:3, 50:53, -3:0]]
)
```

[77]:

uid	ingredients_
345591f4-d887-4ddc-bb40-21337fa9269d	gesier de dinde emince 50 graisse de canard 47…
c3b6b4df-e586-4f10-8e58-15fbf0816acb	mini poivrons jaunes eau sucre sel affermissan…
0481d91b-9653-42e7-b525-9dc9b87b06f2	farine de ble huile de colza non hydrogenee oe…
d8b0687e-5e25-4ff9-9384-087f993218bc	eau jus de fruit de la passion sucre epaississ…
536361db-1bbb-4e64-ae53-d970eeac7db2	sucre amidon de mais arome vanille
194419d0-d9f2-4799-81ac-d9e3aa77fd27	pommes de terre eau sel
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3	jus orange base de concentre
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5	poire 99 antioxydant acide ascorbique
04235024-80f3-46c2-bad0-aae0d5fab024	persil

uid	predicted_	levenshtein
345591f4-d887-4ddc-bb40-21337fa9269d	gesier de dinde emince 50 graisse de canard 47…	1.000000
c3b6b4df-e586-4f10-8e58-15fbf0816acb	mini poivrons jaunes eau sucre sel affermissan…	1.000000
0481d91b-9653-42e7-b525-9dc9b87b06f2	farine de ble huile de colza non hydrogenee oe…	1.000000
d8b0687e-5e25-4ff9-9384-087f993218bc	cereales contenant du gluten crustaces et prod…	0.200692
536361db-1bbb-4e64-ae53-d970eeac7db2	ajouter le produit la preparation avec les aut…	0.200000
194419d0-d9f2-4799-81ac-d9e3aa77fd27	anhydride sulfureux et sulfites en concentrati…	0.154762
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3	valeurs nutritionnelles pour 100ml nutrition f…	0.104418
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5	ce produit est une puree de fruits obtenue par…	0.096667
04235024-80f3-46c2-bad0-aae0d5fab024	cereales contenant du gluten savoir ble seigle…	0.045455

uid	damerau-levenshtein	jaro	jaro-winkler	levenshtein_rank
345591f4-d887-4ddc-bb40-21337fa9269d	1.000000	1.000000	1.000000	1.0
c3b6b4df-e586-4f10-8e58-15fbf0816acb	1.000000	1.000000	1.000000	3.0
0481d91b-9653-42e7-b525-9dc9b87b06f2	1.000000	1.000000	1.000000	4.0
d8b0687e-5e25-4ff9-9384-087f993218bc	0.200692	0.553928	0.553928	68.0
536361db-1bbb-4e64-ae53-d970eeac7db2	0.200000	0.491130	0.491130	69.0
194419d0-d9f2-4799-81ac-d9e3aa77fd27	0.154762	0.482876	0.482876	73.0
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3	0.104418	0.423567	0.423567	78.0
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5	0.096667	0.437327	0.437327	80.0
04235024-80f3-46c2-bad0-aae0d5fab024	0.045455	0.507576	0.507576	86.0

uid	damerau-levenshtein_rank	jaro_rank	jaro-winkler_rank
345591f4-d887-4ddc-bb40-21337fa9269d	1.0	1.0	1.0
c3b6b4df-e586-4f10-8e58-15fbf0816acb	3.0	3.0	4.0
0481d91b-9653-42e7-b525-9dc9b87b06f2	4.0	4.0	5.0
d8b0687e-5e25-4ff9-9384-087f993218bc	68.0	64.0	66.0
536361db-1bbb-4e64-ae53-d970eeac7db2	69.0	75.0	75.0
194419d0-d9f2-4799-81ac-d9e3aa77fd27	73.0	76.0	76.0
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3	78.0	83.0	83.0
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5	80.0	82.0	82.0
04235024-80f3-46c2-bad0-aae0d5fab024	86.0	73.0	73.0

uid	ingredients
345591f4-d887-4ddc-bb40-21337fa9269d	Gésier de dinde émincé 50%, graisse de canard …
c3b6b4df-e586-4f10-8e58-15fbf0816acb	mini poivrons jaunes, eau, sucre, sel, affermi…
0481d91b-9653-42e7-b525-9dc9b87b06f2	Farine de BLE, huile de colza non hydrogénée, …
d8b0687e-5e25-4ff9-9384-087f993218bc	Eau ; jus de fruit de la passion ; sucre ; épa…
536361db-1bbb-4e64-ae53-d970eeac7db2	Sucre, amidon de maïs, arôme vanille
194419d0-d9f2-4799-81ac-d9e3aa77fd27	Pommes de terre, eau, sel.
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3	Jus d'orange à base de concentré
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5	Poire 99,9%, antioxydant: acide ascorbique.
04235024-80f3-46c2-bad0-aae0d5fab024	Persil

uid	predicted
345591f4-d887-4ddc-bb40-21337fa9269d	Gésier de dinde émincé 50%, graisse de canard…
c3b6b4df-e586-4f10-8e58-15fbf0816acb	mini poivrons jaunes, eau, sucre, sel, affermi…
0481d91b-9653-42e7-b525-9dc9b87b06f2	Farine de BLE, huile de colza non hydrogénée, …
d8b0687e-5e25-4ff9-9384-087f993218bc	Céréales contenant du gluten (2) \nCrustacés e…
536361db-1bbb-4e64-ae53-d970eeac7db2	ajouter le produit à la préparation avec les a…

```
194419d0-d9f2-4799-81ac-d9e3aa77fd27  Anhydride sulfureux et sulfites en \nconcentra…
ebfc9e73-5d91-4b45-8331-8c8f9bed3bb3   \n \nVALEURS NUTRITIONNELLES pour 100mL / NUT…
3634fb1e-ee79-41d1-8aaa-084c1fae5bd5  Ce produit est une purée de fruits obtenue à p…
04235024-80f3-46c2-bad0-aae0d5fab024  Céréales contenant du gluten (à savoir blé, se…
```