

PROJET 6

AMÉLIOREZ LE PRODUIT IA DE VOTRE START-UP

- EN DÉTECTANT LES SUJETS D'INSATISFACTION DANS DES COMMENTAIRES
- EN LABELISANT AUTOMATIQUEMENT DES PHOTOS POSTÉES

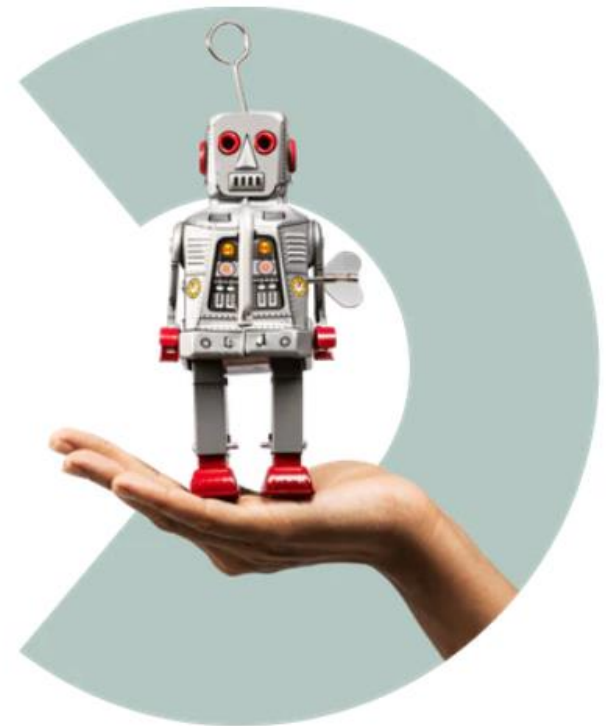
#NLP #LDA #LSA #TSNE #COMPUTER VISION #TRANSFER
LEARNING #VGG16

#NLTK #WORDCLOUD #GENSIM #OPENCV #KERAS
#GRAPHQL

Ingénieur IA

Développez et intégrez des algorithmes de Deep Learning au sein d'un produit IA

Ouddane Nabil



SOMMAIRE

Projet 6

A. INTRODUCTION

1. Contexte
2. Objectifs
3. Ressources complémentaires

B. NLP

C. VISUALISATION

D. API YELP



INTRODUCTION

1. Contexte

- ENJEU global:
 - améliorer la plateforme avec une **nouvelle fonctionnalité de collaboration**
 - Détecter les **sujets d'insatisfaction** présents dans les commentaires postés sur la plateforme.
 - **Labelliser automatiquement les photos** postées sur la plateforme
- ENJEU Compétences DU P5:
 - NLP / Computer Vision / API YELP
- DONNEES SOURCES
 - le jeu de données :
 - <https://www.yelp.com/dataset>



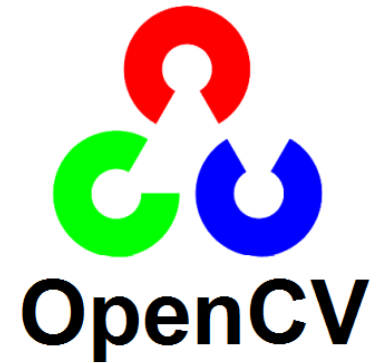
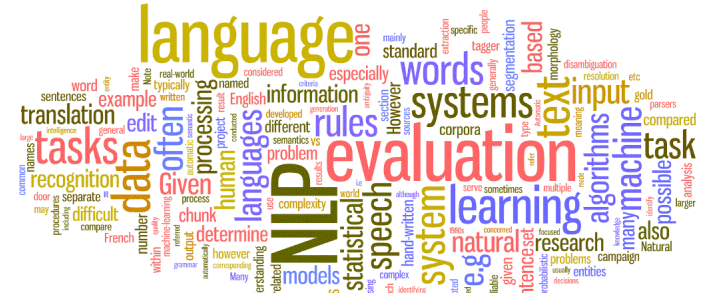
2. Objectifs

- SCRIPT:

- analyser les commentaires pour **détecter les différents sujets d'insatisfaction**
- analyser les photos pour **déterminer les catégories des photos**
- **collecter un échantillon** (environ 200 restaurants) de données via l'API Yelp

- SOUTENANCE :

- Délivrer cette présentation
- Délivrer un script python jupyter autoporteur



3. Ressources complémentaires

- Chargement de gros volumes de données:
 - <https://www.codementor.io/guidotournois/4-strategies-to-deal-with-large-datasets-using-pandas-qdw3an95k>
 - TEXTE
 - Python NLP: https://www.youtube.com/watch?v=SCs8N_-t3cE
 - Preprocessig de texte: <https://datascientest.com/introduction-au-nlp-natural-language-processing>
 - Preprocessig de texte & classification: <https://www.actuia.com/contribution/victorbigand/tutoriel-tal-pour-les-debutants-classification-de-texte/>
 - NLP: <https://towardsdatascience.com/natural-language-processing-nlp-for-machine-learning-d44498845d5b>
 - NLTK: <https://www.nltk.org/>
 - NLTK: <https://code.tutsplus.com/fr/tutorials/introducing-the-natural-language-toolkit-nltk--cms-28620>
 - Cours: <https://openclassrooms.com/fr/courses/6532301-introduction-to-natural-language-processing>
 - Cours: <https://nlp-ensae.github.io/>
 - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html
 - Computer Vizualisation
 - Detecter la meteo: <https://app.livestorm.co/openclassrooms-1/pretraitement-dimages-detecter-automatiquement-la-meteo?type=detailed>
 - https://s3-eu-west-1.amazonaws.com/course.oc-static.com/projects/Webinars/Data/AI_Avril_2021/Weather_first_analysis_V1.0.ipynb
 - Transfer Learning: <https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/>
 - <https://docs.microsoft.com/fr-fr/learn/modules/introduction-to-deep-learning/>
-

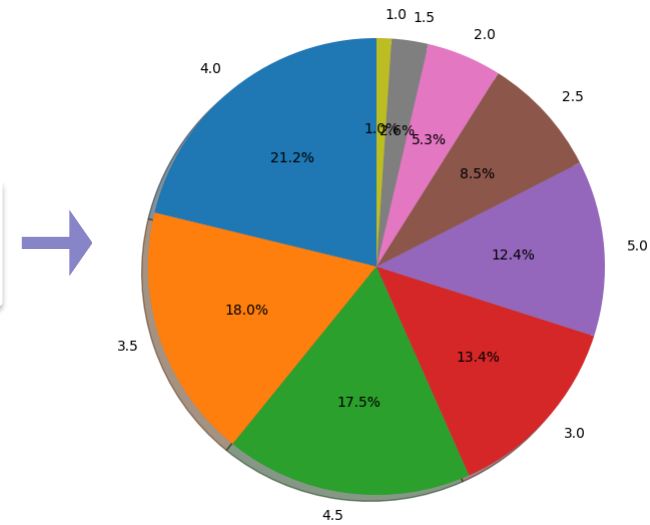
B

**PROJET NLP/LDA:
ANALYSEZ LES
COMMENTAIRES POUR
DÉTECTER LES
DIFFÉRENTS SUJETS
D'INSATISFACTION**

Analyse rapide des fichiers YELP



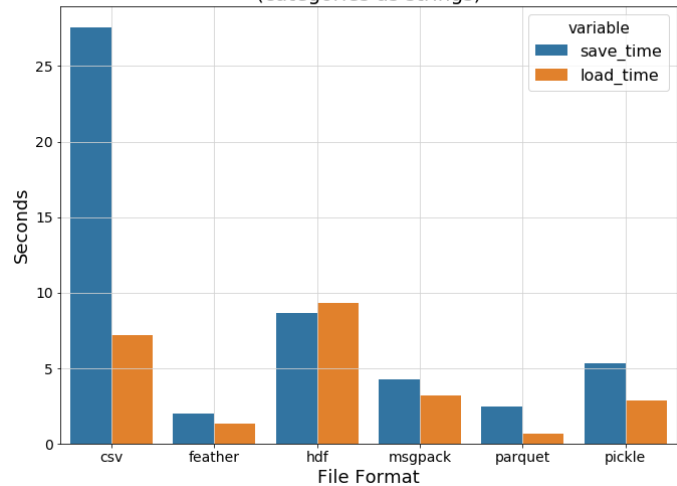
Infos principales:
On a globalement plus de bonnes
reviews que de mauvaises



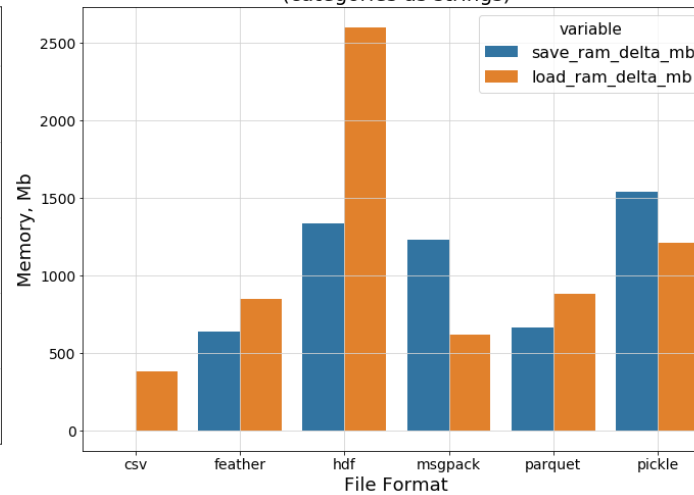
Pourquoi Feather?

<https://towardsdatascience.com/the-best-format-to-save-pandas-data-414dca023e0d>

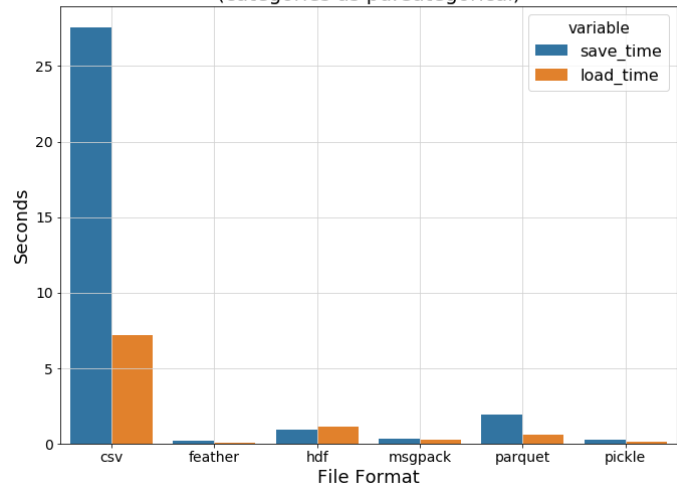
Time to Save/Load a Data Frame
(categories as strings)



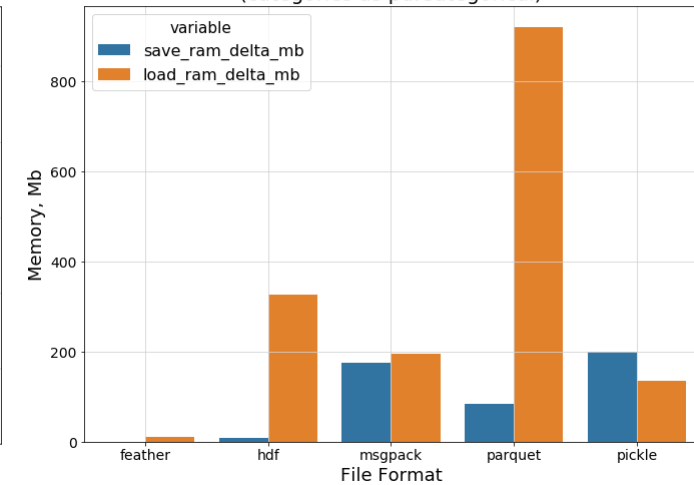
Memory Consumption Growth When Saving/Loading
(categories as strings)



Time to Save/Load a Data Frame
(categories as pd.Categorical)



Memory Consumption Growth When Saving/Loading
(categories as pd.Categorical)



Reviews PreProcessing

- Suppression des textes entre crochets
 - Suppression des liens internet
- Suppression des textes entre chevrons
 - Suppression des retours à la ligne
 - Suppression de la ponctuation
 - Mise en minuscule

Tokenisation

Suppression des stopwords:

Stopwords de la liste

`nltk.corpus.stopwords.words('english')` et enrichie des mots suivants:

`['us', 'im', 'c', 'youre', 'youve', 'youll', 'youd', 'thatll', 'shouldve', 'arent', 'couldnt', 'didnt', 'doesnt', 'dont', 'hadnt', 'hasnt', 'havent', 'isnt', 'mightnt', 'mustnt', 'neednt', 'shant', 'shouldnt', 'wasnt', 'werent', 'wont', 'wouldnt']`

Lemmatization avec le **WordNetLemmatizer** de **NLTK.stemm**

Suppression des mots n'appartenant pas au dictionnaire **`nltk.corpus.words.words()`**

Stemming avec le **`nltk.stem.snowball`**

La lemmatization possède l'avantage de garder un mot correct.

Ce n'est pas toujours le cas du stemming dont les algorithmes sont plus ou moins agressifs

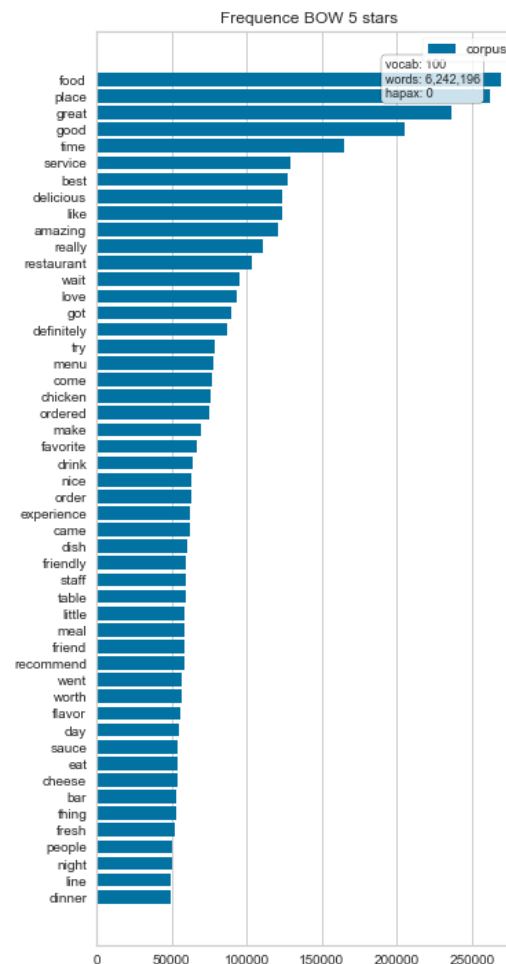
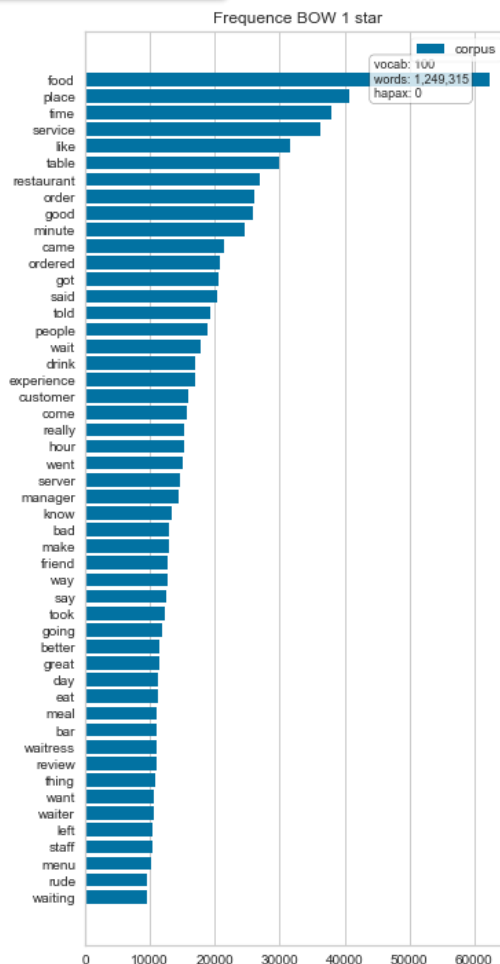
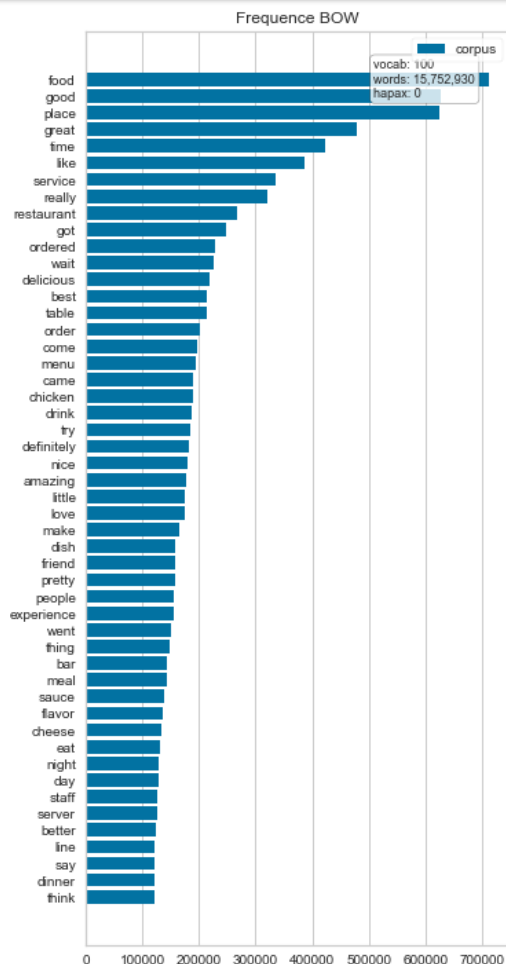
(exemple par ordre croissants d'agressivité:
`nltk.stem.porter`
`nltk.stem.snowball`
`nltk.stem.Lancaster`)

Problème dans le cas de l'utilisation d'un dictionnaire ou d'un modèle pré entraîné

Vectorisation : Bag of words

fonction `CountVectorizer` de `sklearn.feature_extraction.text`

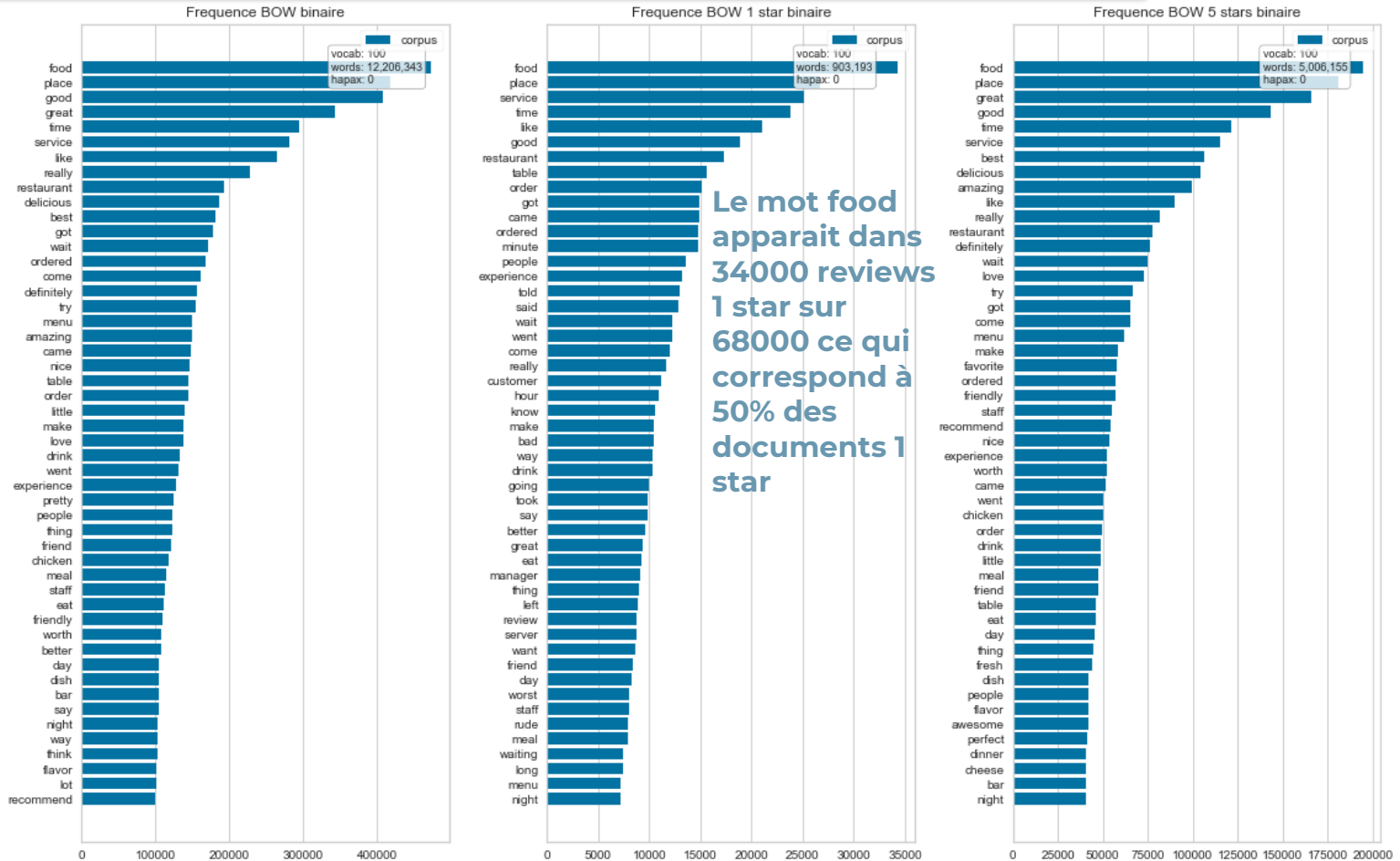
max_features 100 words/bigrams



Vectorisation : Bag of words avec l'option binaire

fonction CountVectorizer de sklearn.feature_extraction.text

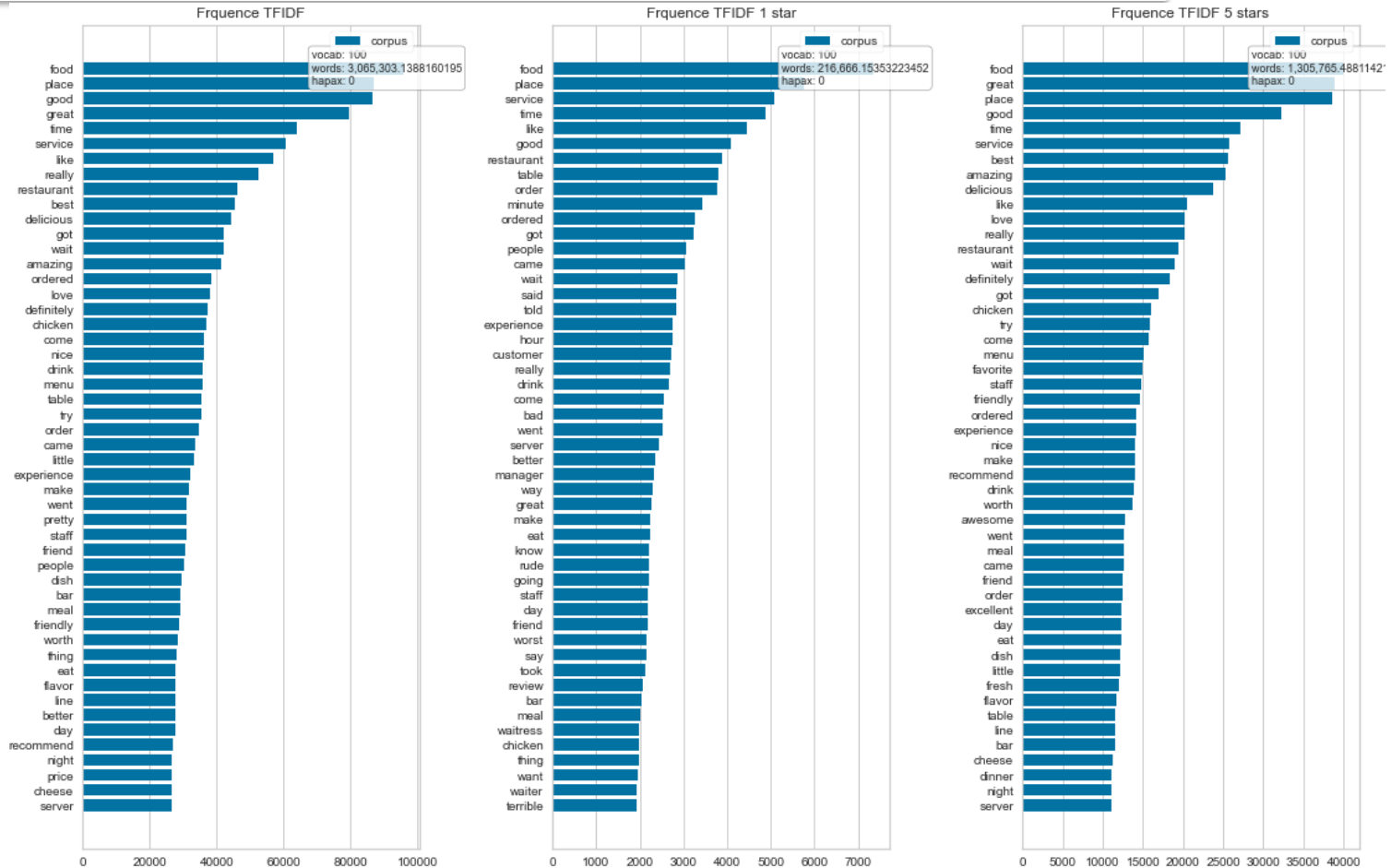
max_features 100 words/bigrams : l'option binaire (presence ou non du mot dans le document sans comptage multiple par document)



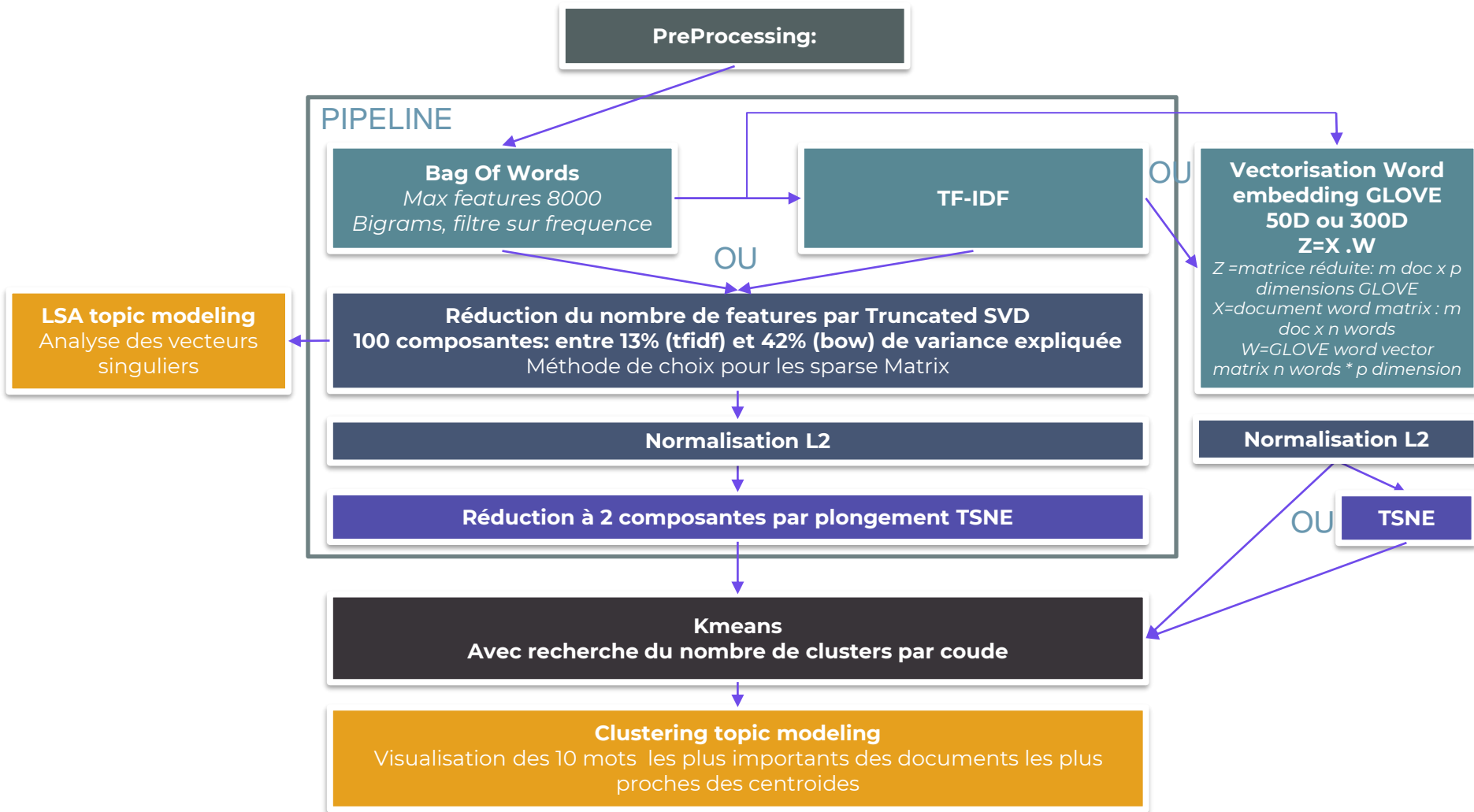
Vectorisation : TF-IDF (permet de donner moins de poids aux tokens trop fréquents et donc peu discriminants)

fonction `TfidfVectorizer` de `sklearn.feature_extraction.text`

max_features 100 words/bigrams : l'option binaire (presence ou non du mot dans le document sans comptage multiple par document)

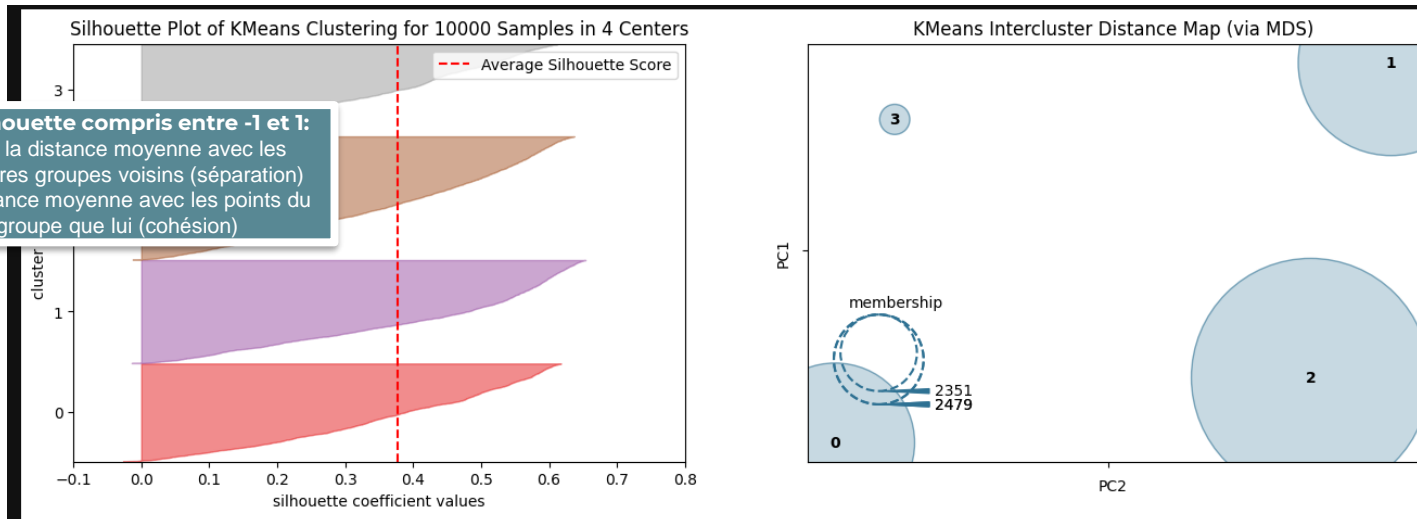
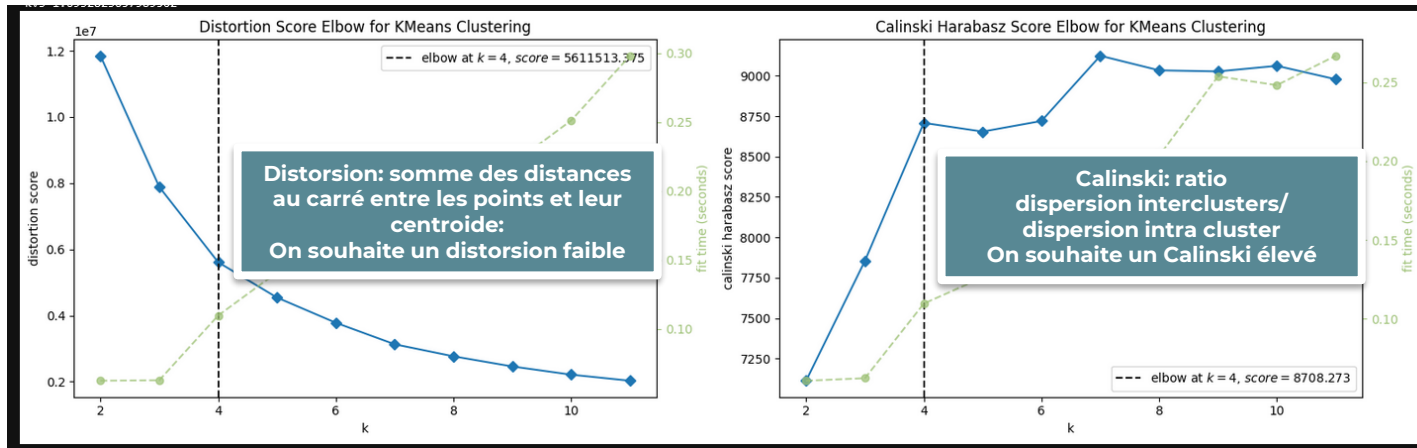


BAD-REVIEWS – TOPIC Modeling – CLUSTERING



Cas: bag of words: filtre [2%-50%] / max features < 8000

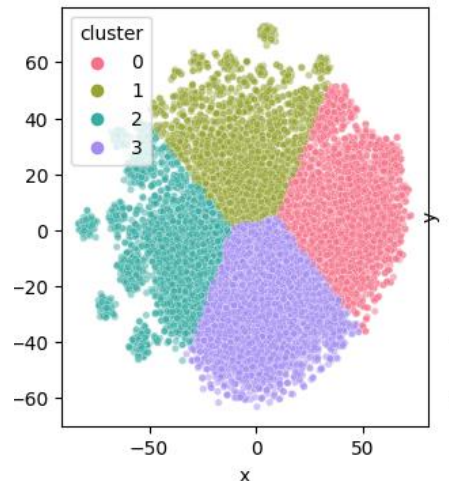
recherche du k -> 4



Cas: bag of words – Mots importants des centroides

Visualisation TSNE des clusters

: 4clus-bow-TSVD-100-TSNE-10000 doc



Mots les plus importants des documents les plus proches des centroides

```
Cluster 0 words:[['like' 'minute' 'hour' 'came' 'saw' 'minute later' 'right' 'customer' 'staff' 'wait']]
```

```
Cluster 1 words:[['got' 'meal' 'tried' 'night' 'literally' 'time' 'dollar' 'awful' 'fact' 'menu']]
```

```
Cluster 2 words:[['taste' 'like' 'loud' 'used' 'portion' 'food' 'fish' 'sweet' 'expensive']]
```

```
Cluster 3 words:[['hostess' 'cold' 'like' 'food' 'want' 'waiter' 'friendly' 'disappointment' 'chicken' 'meal']]
```

Cluster 0 : attente
Cluster 1 : plat, prix
Cluster 2 : plat, prix
Cluster 3 : service, plat

Documents filtrés les plus proches des centroides des clusters

4998 suppose would good ever make oh wait consciously gave someone else never made new lunch today ordered two turkey two transaction sec
ond apart came half hour told mine way ten minute later saw turkey materialize eagerness counter boss um ten minute go another turkey anoth
er customer twenty minute came best part staff try look like right like exist like maybe acknowledge mistake go away got refund demanding s
ince wasted hour life ton attitude food back

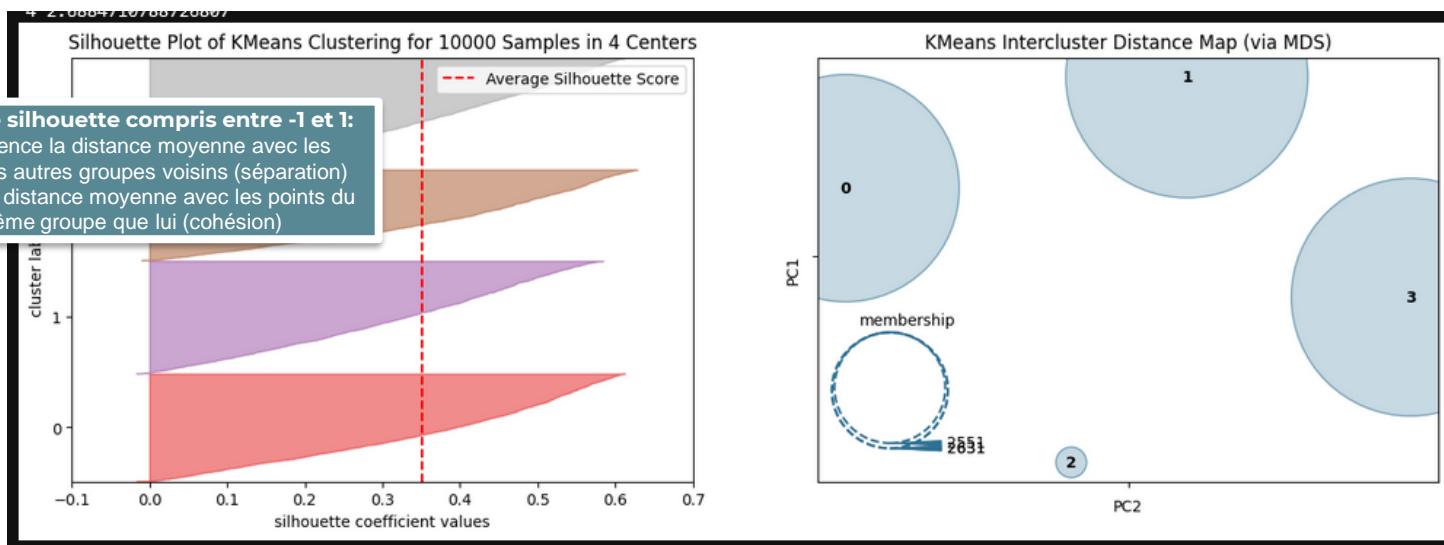
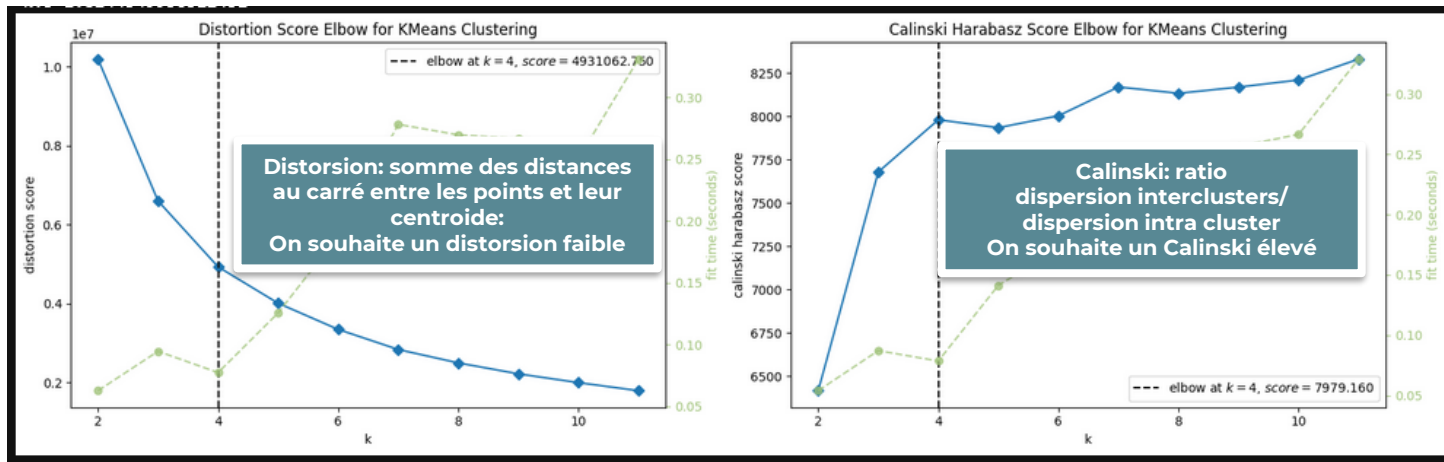
3410 night got possibly worst meal ever life tried write e mail give opportunity correct almost week never back ordered dish amazing menu
picture fact almost positive came literally soup butter spent dollar meal tried eat since closed time got nothing could got literally hour
later ended spending night throwing awful experience shame used love think well thanks

5487 sweet careful diabetes loud like fish market except taste like food expensive term portion taste ingredient used

5852 excited trying restaurant disappointment food cold service poor waiter friendly felt rushed entire meal fried chicken cold oxtail gr
it entirely salty honest server unfriendly one staff around like zombie like want exception hostess hostess nice back

Cas: TFIDF: filtre [2%-50%] / max features < 8000

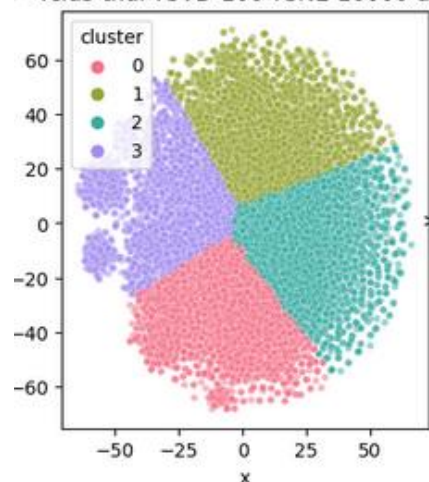
recherche du k -> 4



Cas: TF-IDF– Mots importants des centroides

Visualisation TSNE des clusters

4clus-tfidf-TSVD-100-TSNE-10000 doc



Mots les plus importants des documents les plus proches des centroides

```
Cluster 0 words:[['food good' 'attitude' 'love' 'trying' 'busy' 'gave' 'need' 'cold'
'ask' 'check']]

Cluster 1 words:[['morning' 'suck' 'forgot' 'word' 'house' 'entire' 'literally' 'high'
'feel' 'person']]

Cluster 2 words:[['chef' 'matter' 'idea' 'soon' 'cook' 'speak' 'breakfast' 'potato'
'believe' 'egg']]

Cluster 3 words:[['based' 'chance' 'pick' 'unless' 'short' 'average' 'giving' 'course'
'steak' 'quite']]
```

Cluster 0 : food, service
Cluster 1 : service
Cluster 2: food
Cluster 3: ?

6129 food fantastic service caught much attitude place go back last time gave try two server busy chatting behind bar place full waiting five minute ask table got cold stare told would minute chat left love food good need put like hobo trying check

1026 like food taste like something forgot microwave night robber forced eat upon breaking house morning look wall literally like high school gym floor smell ambience restaurant carried repulsive odor ground literally like construction crew dirt paint throughout entire restaurant random imagery word painted across place like selected crazy person random choose put lightly place suck like feel like garbage eat garbage place

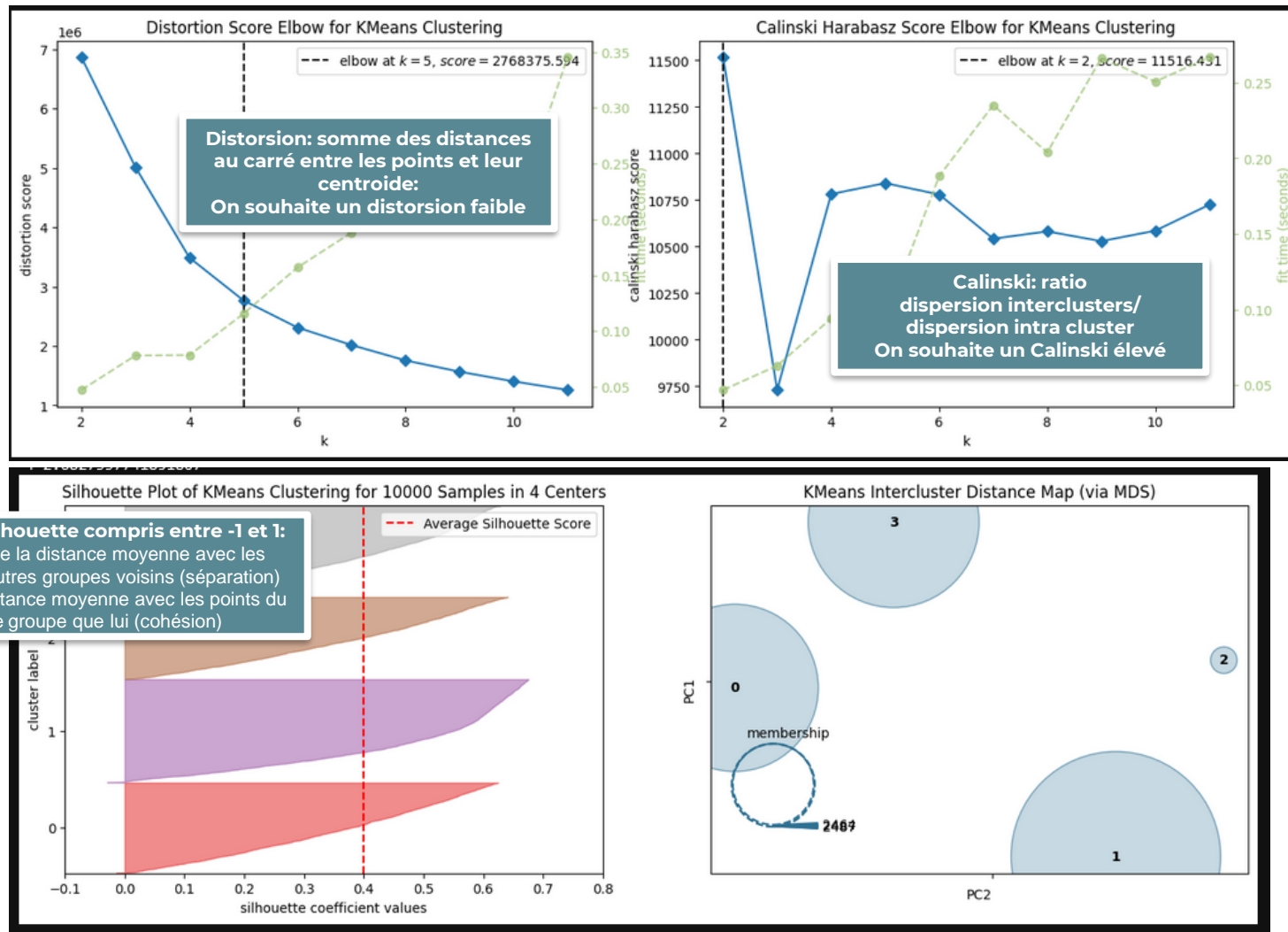
9605 husband went breakfast thought order southern breakfast husband ordered simple breakfast egg medium potato sausage toast breakfast came egg cold told server egg cold removed plate table moment later brought dish back minus egg said egg soon minute egg time everything else dish cold speak manager wrong decision part manager person ever dealt service position told could would cook bring hot egg another manager came said would check chef making new egg chef owner please customer told would serve leave believe owner would say never dealt person person idea customer service moral story eat send like matter

8571 many better steak house option city actually would honor business would stay away capital grill least sort quality service quite possibly worst city tend eat bar every time go often find getting sort quality service capital grill come request almost like hook check average meaning ask get even ask guarantee actually get unless regular course short based bartender capital grill almost appear union mentality work ethic riding gravy train completely fact good service high check average money cater little interest grow customer base rather pick alternative steak house many better one city would love business place stink done giving yet another chance good luck

Documents filtrés
les plus proches
des centroides
des clusters

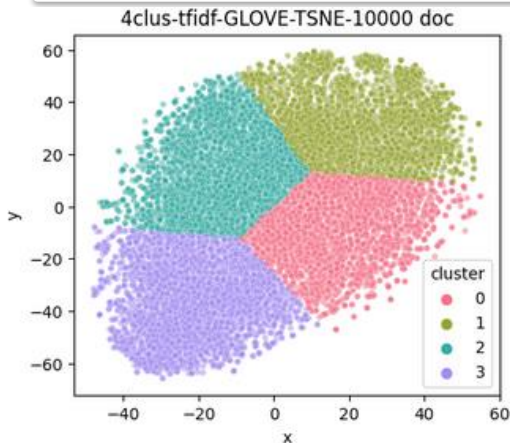
Cas: TFIDF: filtre [1%-40%] / reduction word embedding GLOVE 50D

recherche du k -> 4/5



Cas: TF-IDF- GLOVE– Mots importants des centroides

Visualisation TSNE des clusters



Mots les plus importants des documents les plus proches des centroides

```
Cluster 0 words:['love' 'simple' 'thanks' 'local' 'breakfast' 'excited' 'reason' 'home'
'pretty' 'tried']

[1327 2551 865 129 891 9454 2313 1551 50 117]
Cluster 1 words:['husband' 'restaurant' 'manager' 'group' 'leave' 'reservation' 'stated'
'host' 'new' 'like']

[20405 154 362 1522 7635 2004 9956 4906 1633 86908]
Cluster 2 words:['waiter' 'told' 'head' 'knew' 'pork' 'extra' 'awful' 'kitchen' 'seat'
'thats']

[ 5795 823 3806 20631 2529 3017 460 6969 2056 495]
Cluster 3 words:['cheese' 'hand' 'meat' 'lettuce' 'picked' 'hair' 'order' 'checked'
'clearly' 'issue']
```

Cluster 0 : food, service
Cluster 1 : service
Cluster 2: food,
atmosphere, service
Cluster 3: food

9397 there reason get excited place menu simple enough make home thanks mi tried love love mister breakfast pancake pretty boring local find back

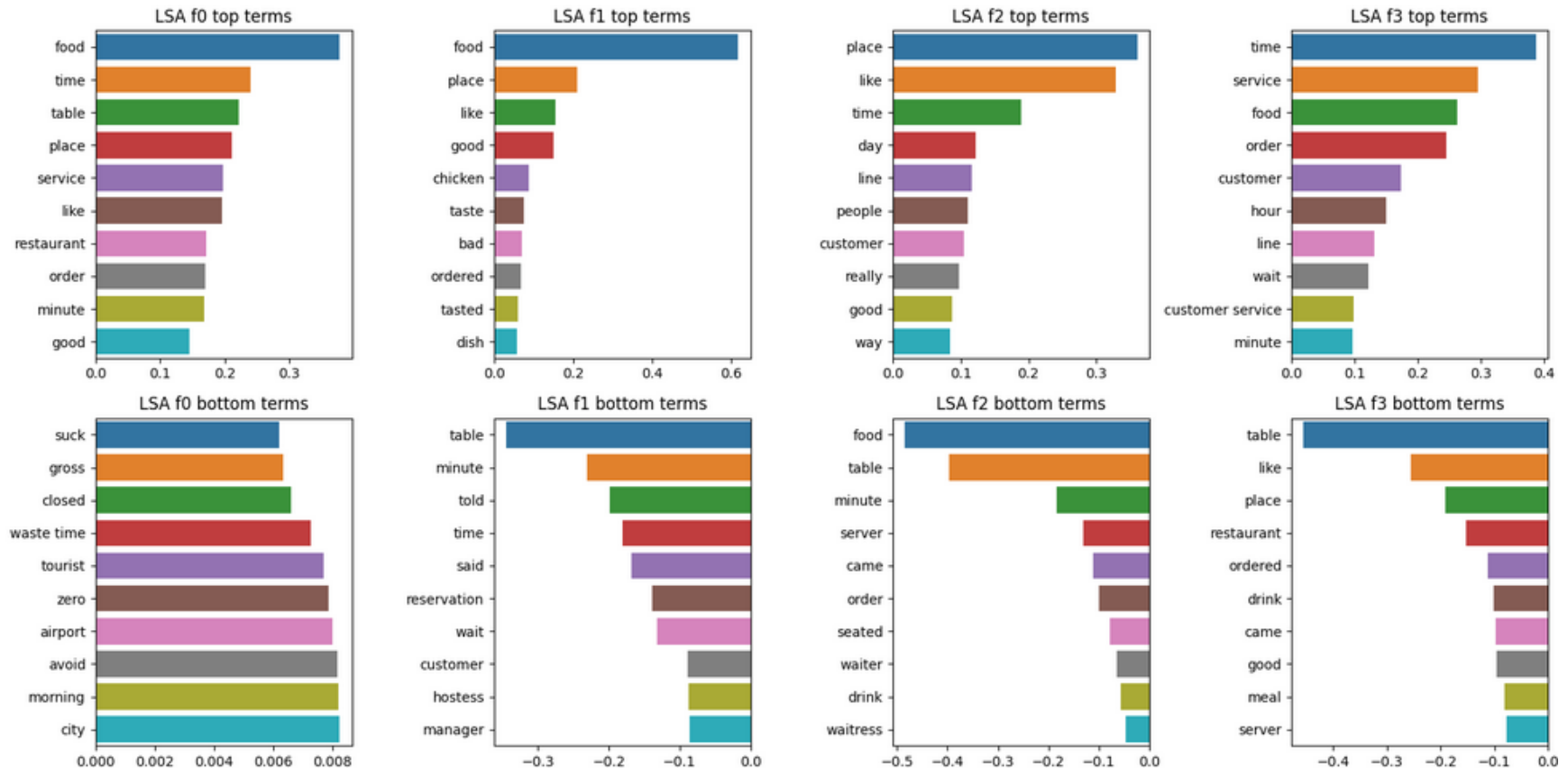
6277 option leave zero star would review quite different even get opportunity eat restaurant however made reservation group restaurant bit early checked informed added another guest reservation stated full please wait door husband went back ask bar see first come first serve disappointed expressed back group manager husband used profanity towards host husband speaking wife said put kind behavior away husband went host area manager husband appreciate unprofessional coming group like concern something issue correct way handle customer manager stated need leave felt underneath restaurant black couple leave restaurant term something wrong never like typically travel spend weekend multiple time year never step foot restaurant manager ashamed acting like towards new customer eat restaurant even get table reservation made day advanced sorry respect mean trying new restaurant

569 review wack want seat party separate table told hurry order kitchen extra sheet dim sum could split check waiter away shaking head besides horrible service might worst cuisine ever thats broad umbrella pork dumpling uncooked told waiter awful food waiter said really already knew

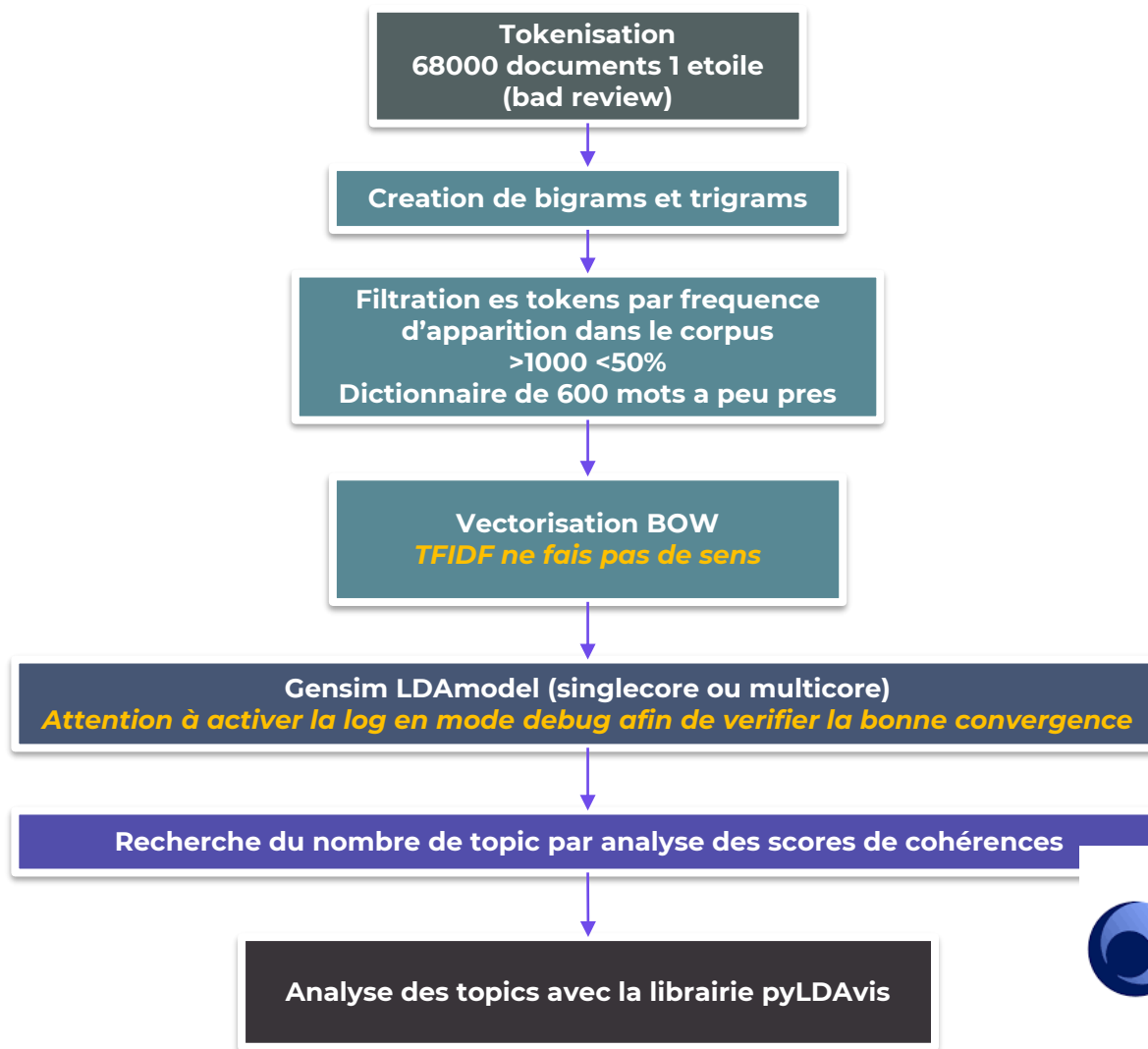
8997 order al carbon substituted meat lettuce due dietary restriction dairy meat upon arrival checked order find cheese even come al carb on waitress could remake cheese although clearly looking took hand bring right back obvious someone hand picked cheese caught attitude someone else deliver new issue also hair side first experience nightmare

Documents filtrés
les plus proches
des centroides
des clusters

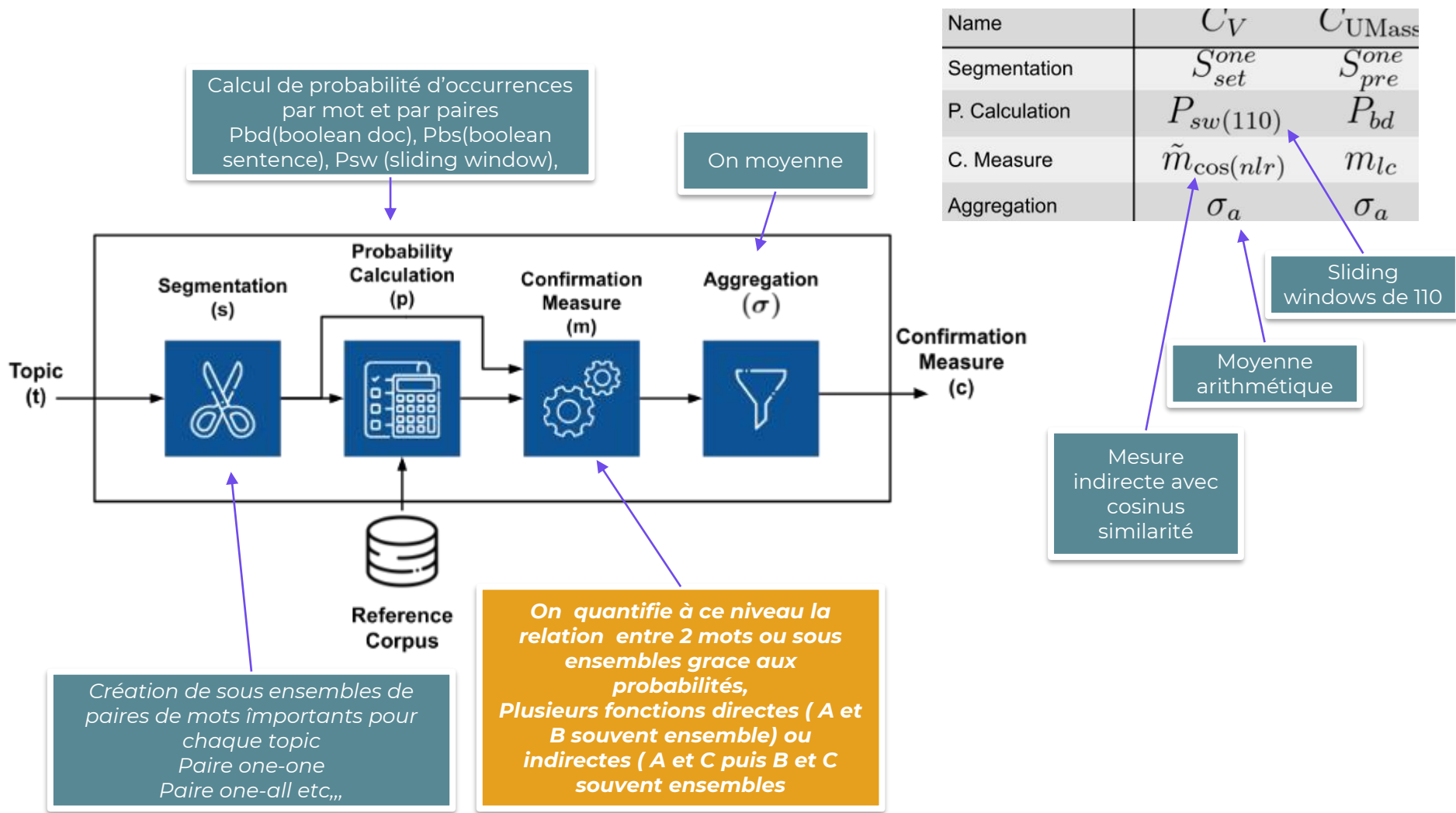
Latent Semantic Analysis (BOW) – analyse des premiers vecteurs singuliers qui peuvent être vus comme des concepts/topics



Latent Dirichlet Allocation – LDA – Topic Modelling



Mesure de cohérence – pipeline de 4 étapes



Latent Dirichlet Allocation – LDA – Topic Modelling

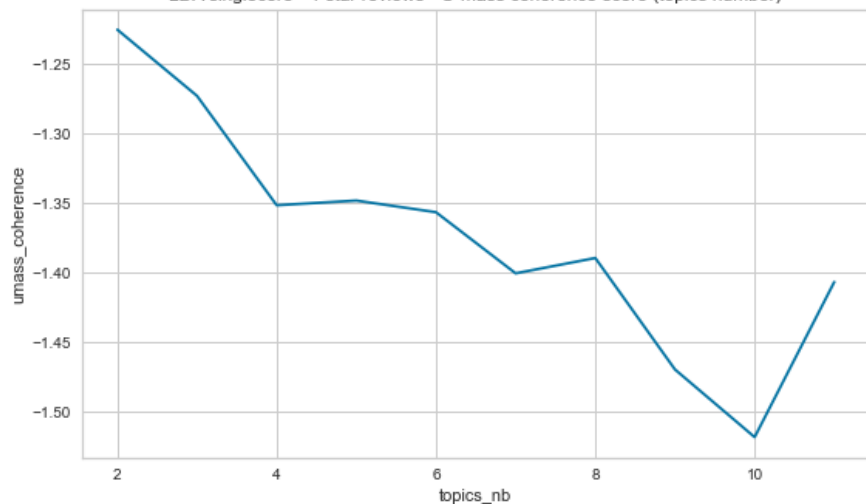
*U mass cohérence
entre -14 et +14*

Cohérence score:
Mesure de similarités
sémantiques entre 2 mots à score
élevé du même topic

C_V cohérence entre 0 et +1

*0,3 mauvais
0,4 à améliorer
0,6 bon
0,7 très bien
Au dessus: improbable*

LDA singlecore - 1 star reviews - U-mass coherence score (topics number)



LDA singlecore - 1 star reviews - C_V coherence score (topics number)



**Possibilité d'améliorer via
gridsearch sur filtre**

TOPIC 1 : servie & attente : '0.024*"table" + 0.017*"told" + 0.016*"minute" + 0.016*"would" + 0.015*"order" + 0.014*"wait" + 0.014*"time" + 0.014*"said" + 0.013*"get" + 0.012*"restaurant"

TOPIC 2 : food & service : '0.033*"food" + 0.018*"service" + 0.014*"came" + 0.014*"ordered" + 0.013*"restaurant" + 0.013*"time" + 0.013*"server" + 0.013*"order" + 0.012*"one" + 0.012*"table"

TOPIC 3 : food & service : '0.028*"food" + 0.026*"place" + 0.019*"get" + 0.017*"time" + 0.015*"go" + 0.014*"like" + 0.013*"people" + 0.012*"service" + 0.011*"one" + 0.011*"good"

TOPIC 4 : food & gout : '0.030*"food" + 0.023*"place" + 0.019*"good" + 0.015*"like" + 0.013*"ordered" + 0.011*"one" + 0.011*"restaurant" + 0.009*"even" + 0.009*"taste" + 0.009*"would"

Latent Dirichlet Allocation – LDA – Topic 1

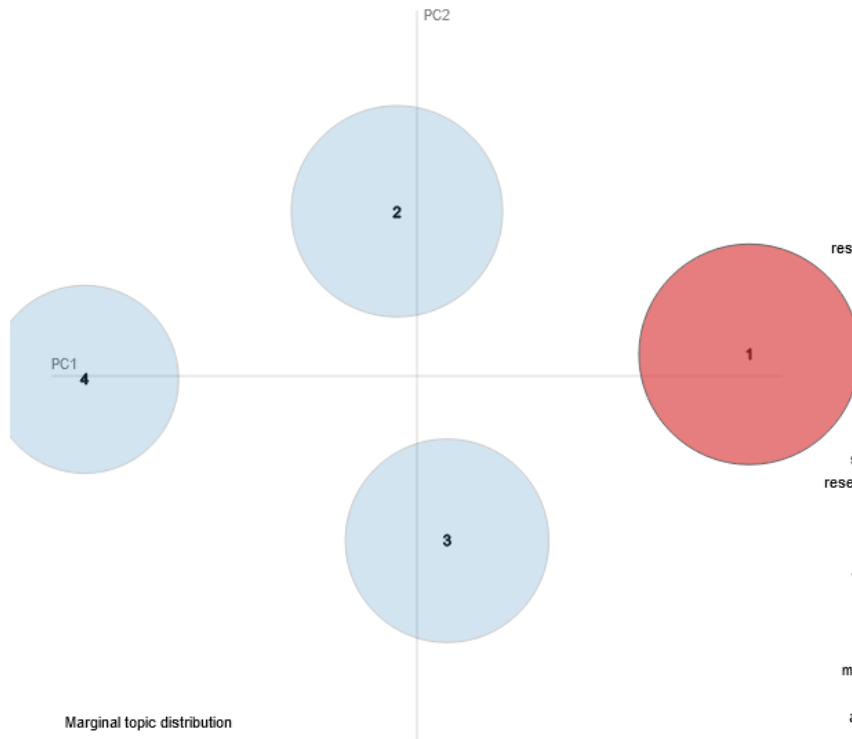
Selected Topic: 0

Slide to adjust relevance metric:⁽²⁾

$\lambda = 1$

0.0 0.2 0.4 0.6 0.8 1.0

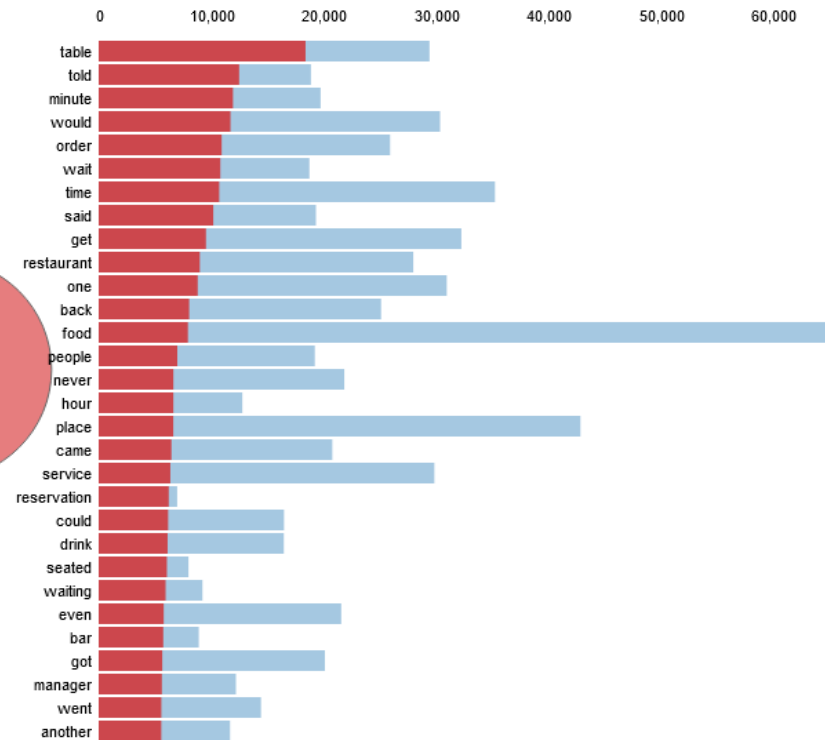
Intertopic Distance Map (via multidimensional scaling)



Marginal topic distribution



Top-30 Most Relevant Terms for Topic 1 (28.6% of tokens)



Overall term frequency

Estimated term frequency within the selected topic

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t)) for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = $\lambda * p(w | t) + (1 - \lambda) * p(w | t)/p(w)$; see Sievert & Shirley (2014)

TOPIC 1: service/attente

Latent Dirichlet Allocation – LDA – Topic 2

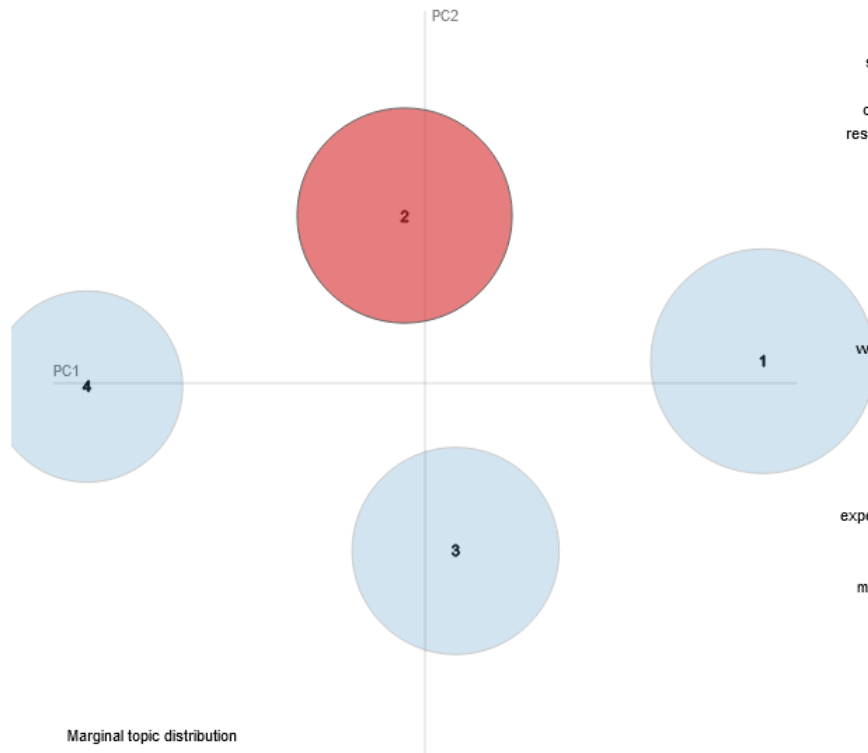
Selected Topic: Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:⁽²⁾

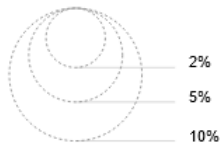
$\lambda = 1$

0.0 0.2 0.4 0.6 0.8 1.0

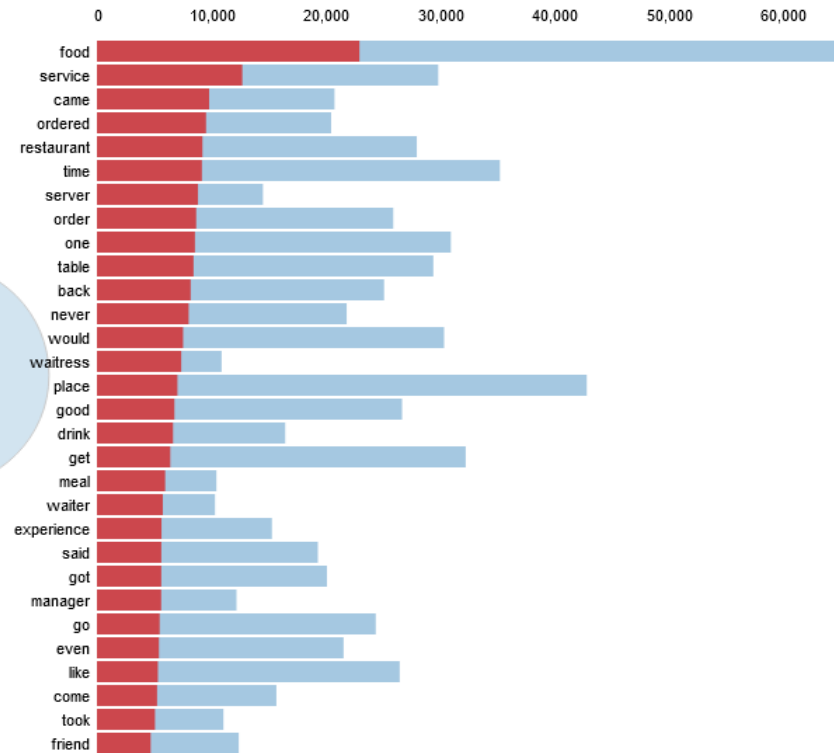
Intertopic Distance Map (via multidimensional scaling)



Marginal topic distribution



Top-30 Most Relevant Terms for Topic 2 (26.3% of tokens)



Overall term frequency

Estimated term frequency within the selected topic

1. $\text{saliency}(\text{term } w) = \text{frequency}(w) * [\sum_t p(t | w) * \log(p(t | w)/p(t))]$ for topics t ; see Chuang et. al (2012)
2. $\text{relevance}(\text{term } w | \text{topic } t) = \lambda * p(w | t) + (1 - \lambda) * p(w | t)/p(w)$; see Sievert & Shirley (2014)

Topic 2: food & service

Latent Dirichlet Allocation – LDA – Topic 3

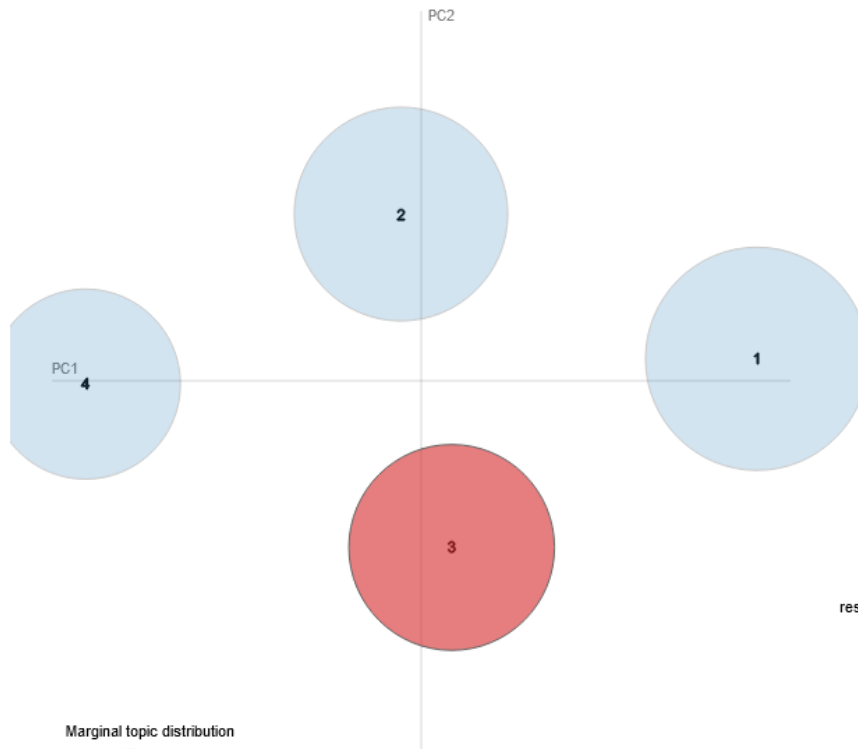
Selected Topic:

Slide to adjust relevance metric:⁽²⁾

$\lambda = 1$

0.0 0.2 0.4 0.6 0.8 1.0

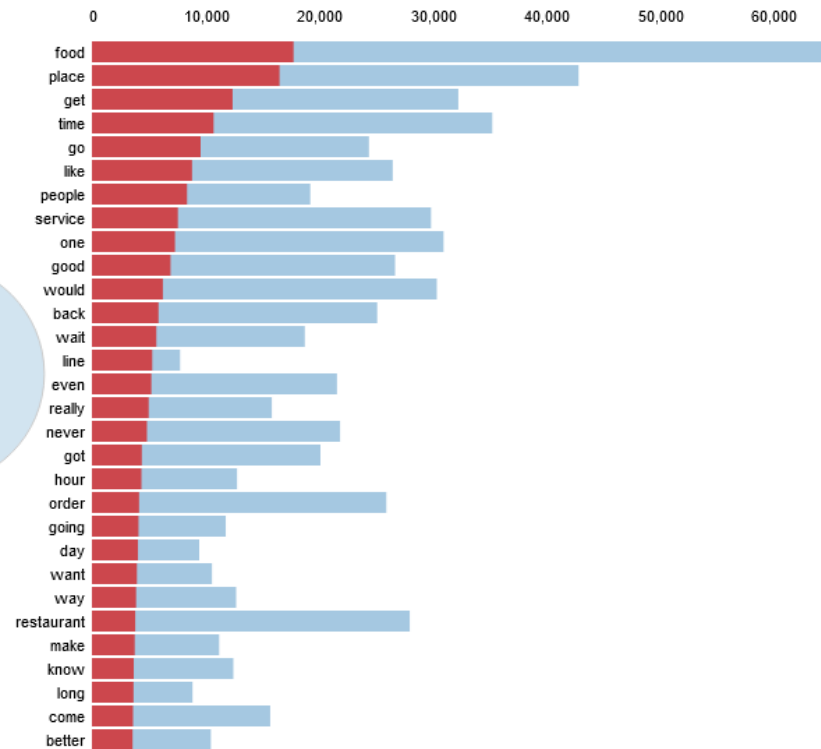
Intertopic Distance Map (via multidimensional scaling)



Marginal topic distribution



Top-30 Most Relevant Terms for Topic 3 (24.4% of tokens)



Overall term frequency

Estimated term frequency within the selected topic

1. saliency(term w) = frequency(w) * $[\sum_t p(t | w) * \log(p(t | w)/p(t))]$ for topics t ; see Chuang et. al (2012)
2. relevance(term w | topic t) = $\lambda * p(w | t) + (1 - \lambda) * p(w | t)/p(w)$; see Sievert & Shirley (2014)

Topic 3: food & service

Latent Dirichlet Allocation – LDA – Topic 4

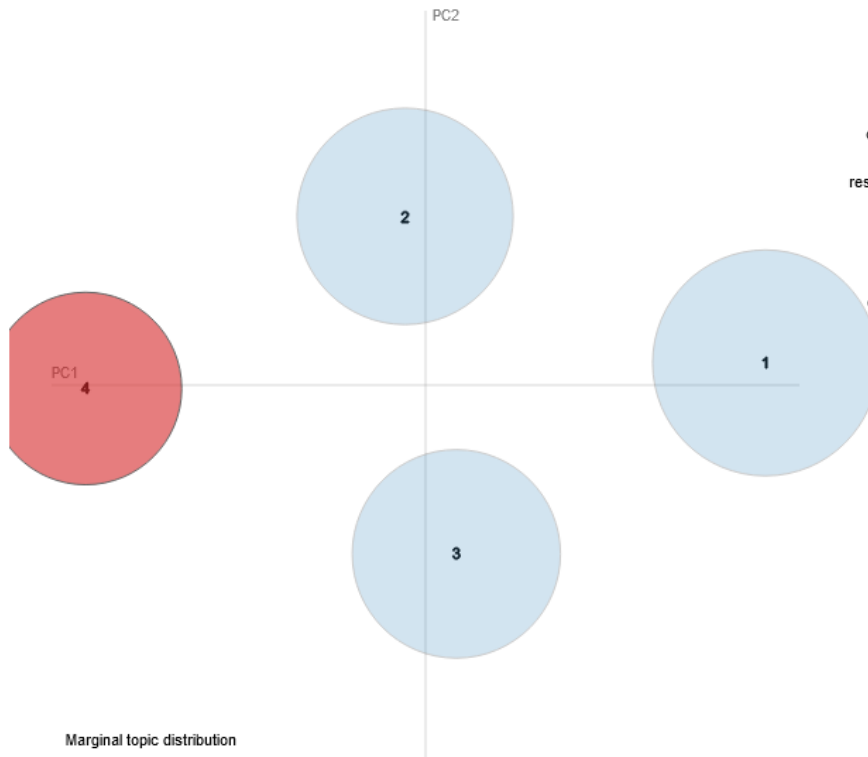
Selected Topic: 4

Slide to adjust relevance metric:⁽²⁾

$\lambda = 1$

0.0 0.2 0.4 0.6 0.8 1.0

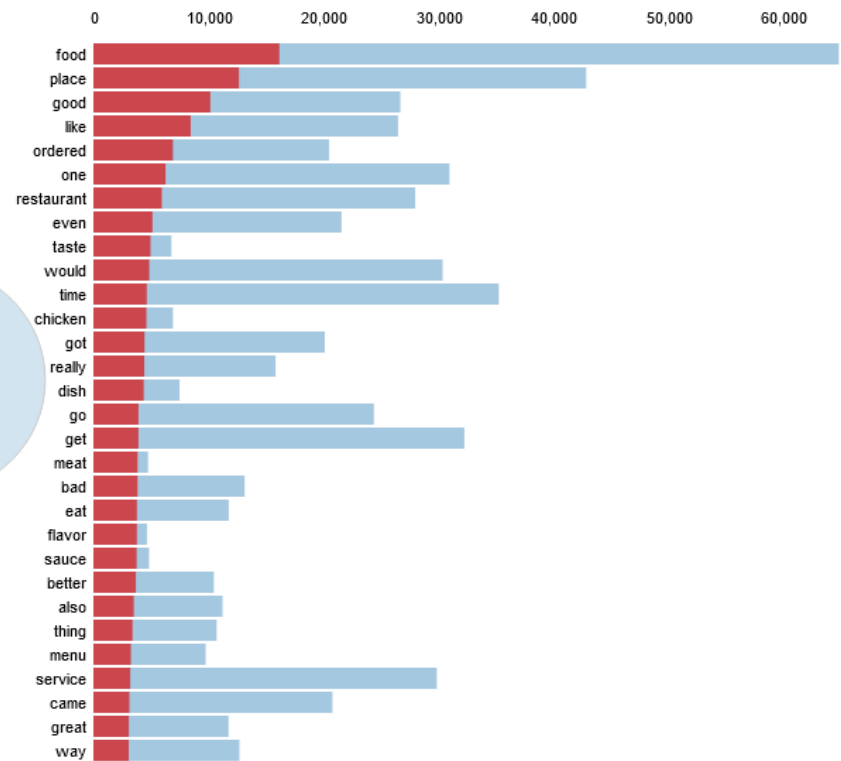
Intertopic Distance Map (via multidimensional scaling)



Marginal topic distribution



Top-30 Most Relevant Terms for Topic 4 (20.8% of tokens)



Overall term frequency
Estimated term frequency within the selected topic

$1 - \text{relevance}(\text{term} | w) = \text{frequency}(w) * [\sum_t p(t | w) * \log(p(t | w) / p(t))]$ for topics t ; see Chuang et. al (2012)
 $p(w | \text{topic } t) = \lambda * p(w | t) + (1 - \lambda) * p(w | t) / p(w)$; see Sievert & Shirley (2014)

Topic 3: food & gout



PROJET VISUALISATION: ANALYSER LES PHOTOS POUR DÉTERMINER LES CATÉGORIES DES PHOTOS

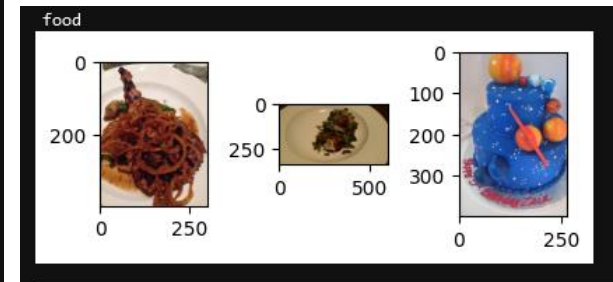
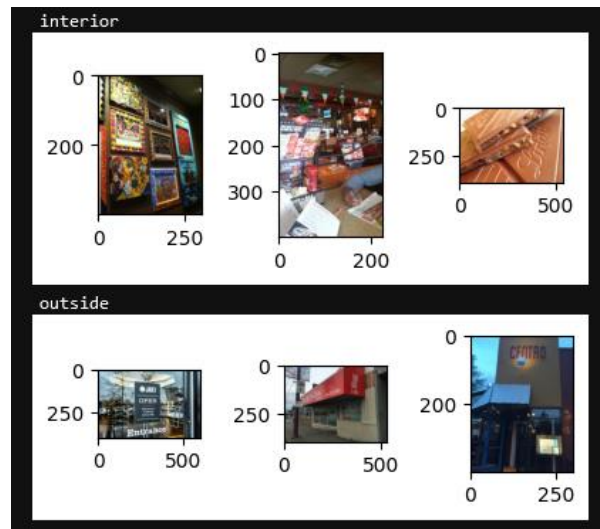
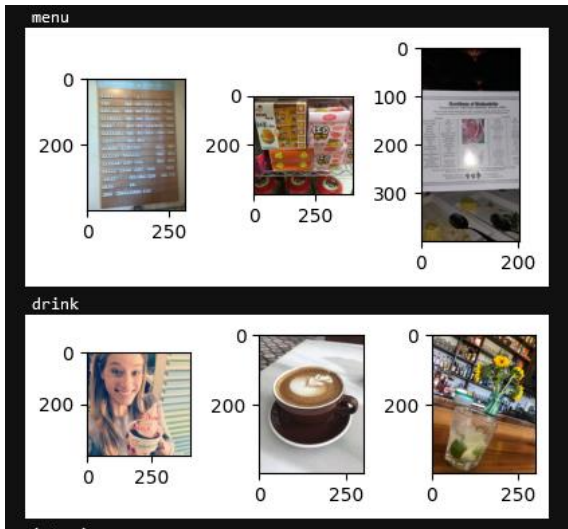
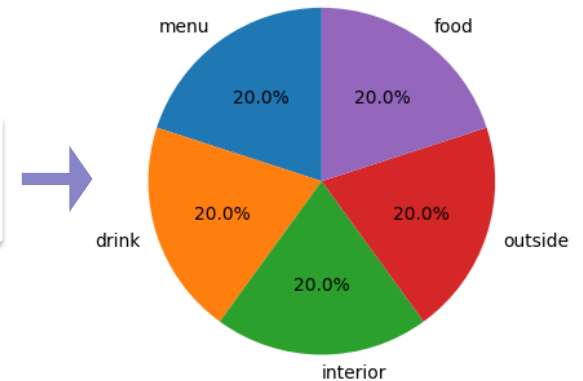
Analyse rapide des images YELP

Download du fichier image yelp .TAR:
<https://www.yelp.com/dataset>
puis décompression

Lecture du fichier json de description
des images

Split du fichier : entraînement / test
En veillant à garder la structure
équilibrée des labels

Infos principales:
Images portant sur 39438 business
200000 photos / 5 labels équilibrés



Préprocessing avec



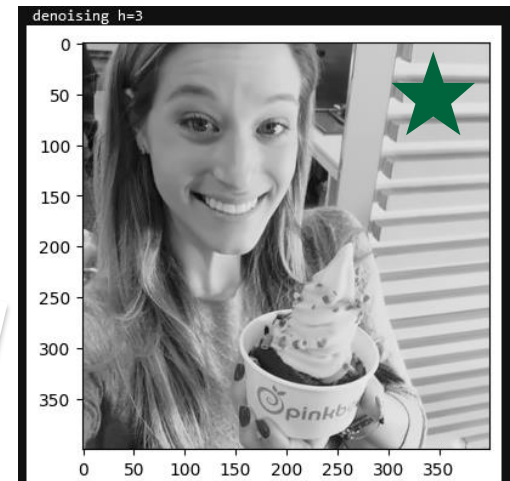
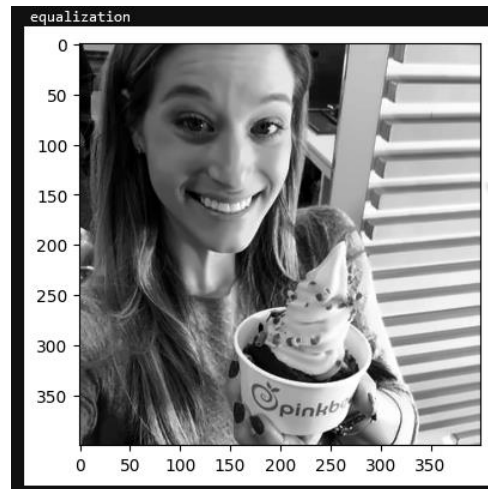
Lecture de l'image en
nuances de gris



Denoising avec la
fonction
fastNlMeansDenoising:
Attention a ne pas
prendre un h trop grand
au risque de trop lisser/
flouter la photo



Equalization avec la
fonction **equalizeHist**
Permet d'améliorer le
contraste en uniformisant
la distribution
(maximisation de
l'information/entropie) de
l'intensité



Extraction de FEATURES avec



Detection et extraction des features avec SIFT (Scale-Invariant Feature Transform) codé sur 128 valeurs

On se limite a 500 features par photo pour ne pas exploser les compteurs

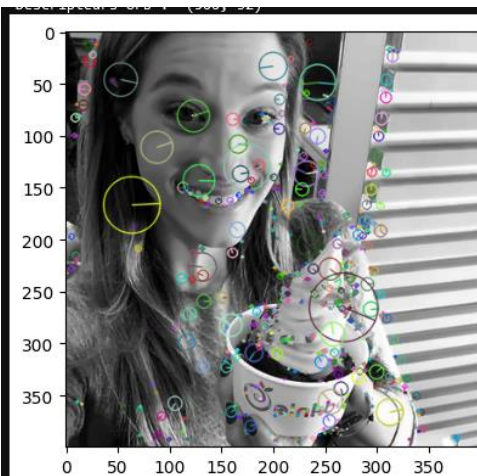
Detection des features avec STAR et extraction des features avec BRIEF codé sur 64 valeurs

6 à 10 fois plus rapide que SIFT

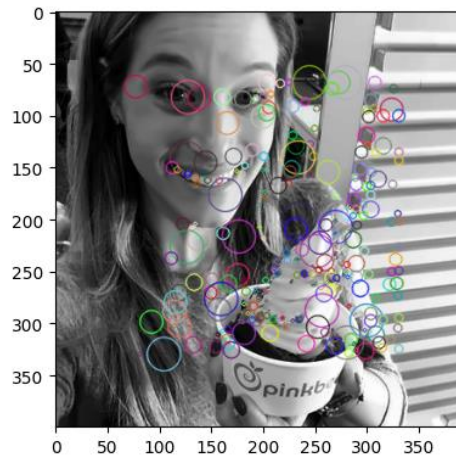
Detection et extraction des features avec ORB codé sur 32 valeurs
(alternative gratuite à SIFT et SURF developpé par opencv)

On se limite a 500 features par photo

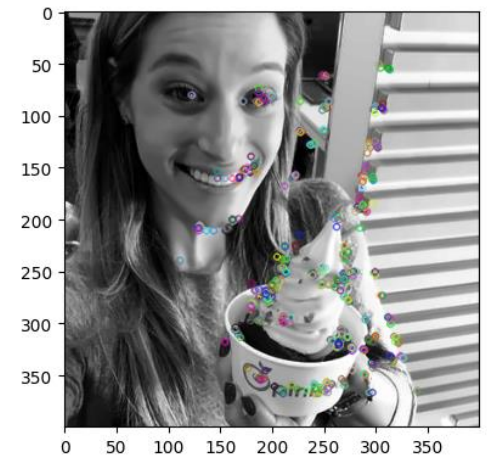
6 à 10 fois plus rapide que SIFT



https://docs.opencv.org/3.4/daf5/tutorial_py_sift_intro.html

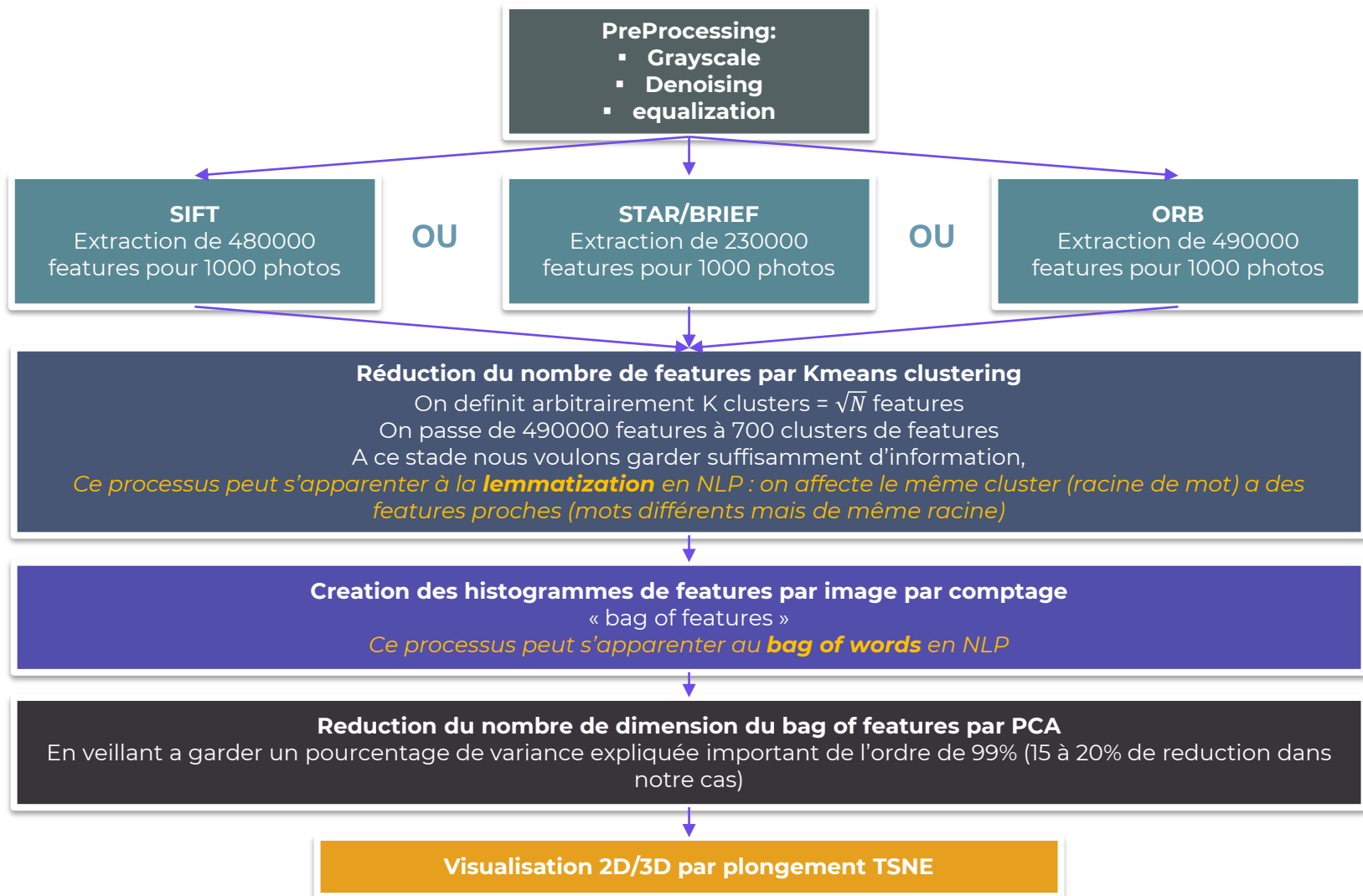


https://docs.opencv.org/3.4/dc/d7d/tutorial_py_brief.html



https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html

Traitement de 1000 photos avec

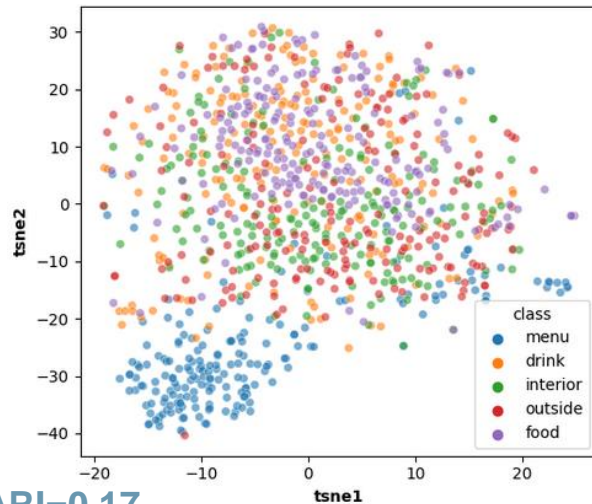


IMAGES: labels initiaux vs TSNE kmeans clustering (K=N labels initiaux)

Visuellement les menus sont assez bien clusterisés,

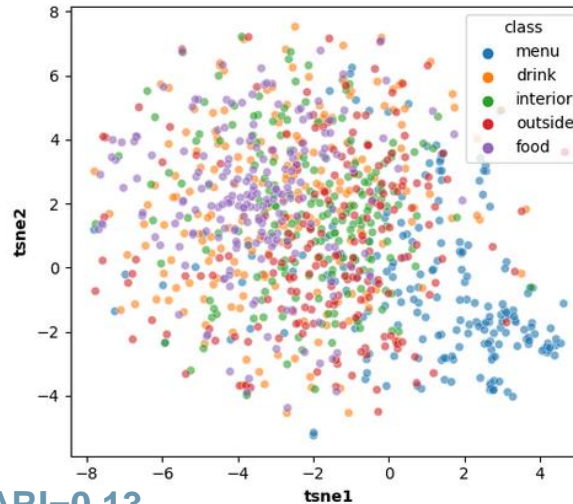
les métriques ARI de comparaison de clustering (label vs features) ne sont pas fantastiques (pour rappel 1 = matching parfait, 0 = aleatoire)

TSNE SIFT selon les labels initiaux



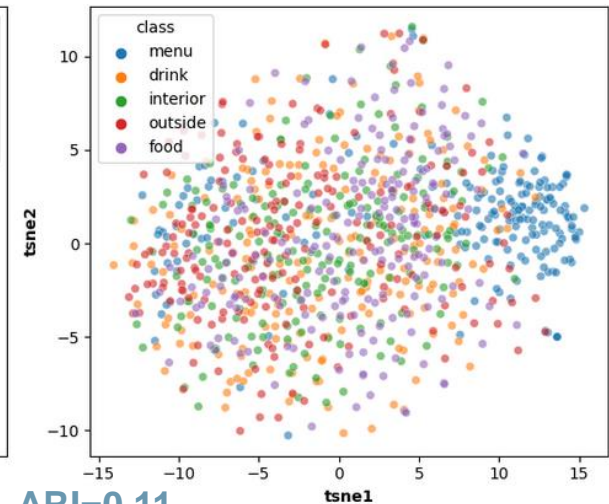
TSNE SIFT: labels 5 MEANS clustering

TSNE BRIEF selon les labels initiaux



TSNE BRIEF: labels 5 MEANS clustering

TSNE ORB selon les labels initiaux

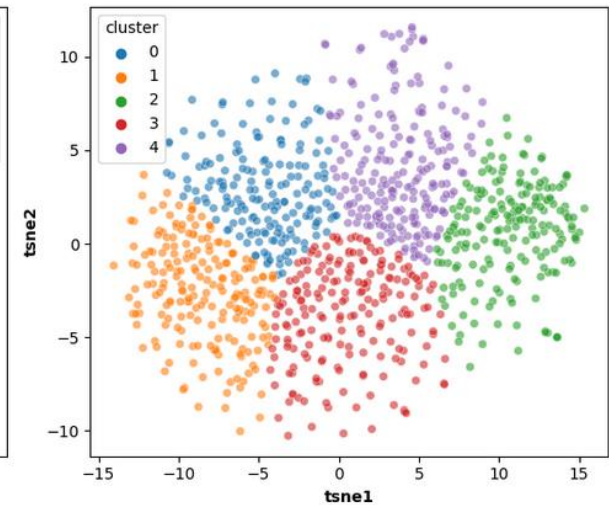
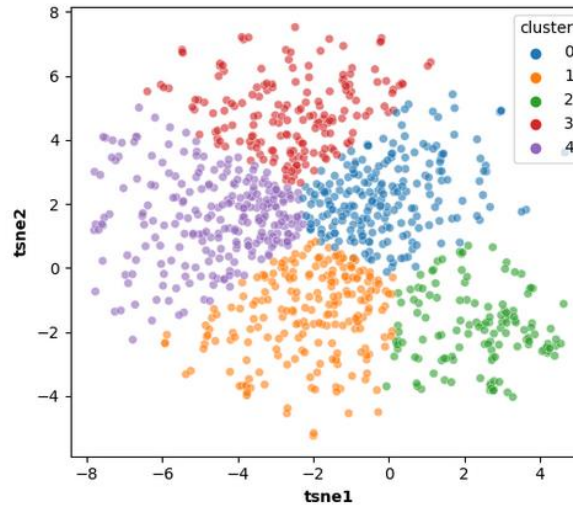
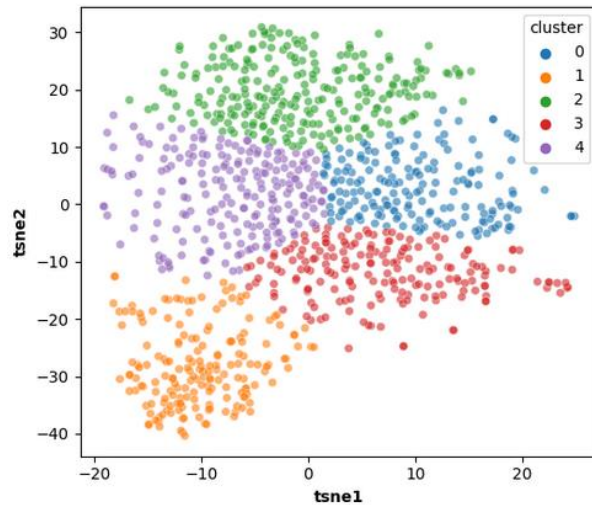


TSNE ORB: labels 5 MEANS clustering

ARI=0,17

ARI=0,13

ARI=0,11



IMAGES: labels initiaux vs TSNE kmeans clustering (K=N labels initiaux)

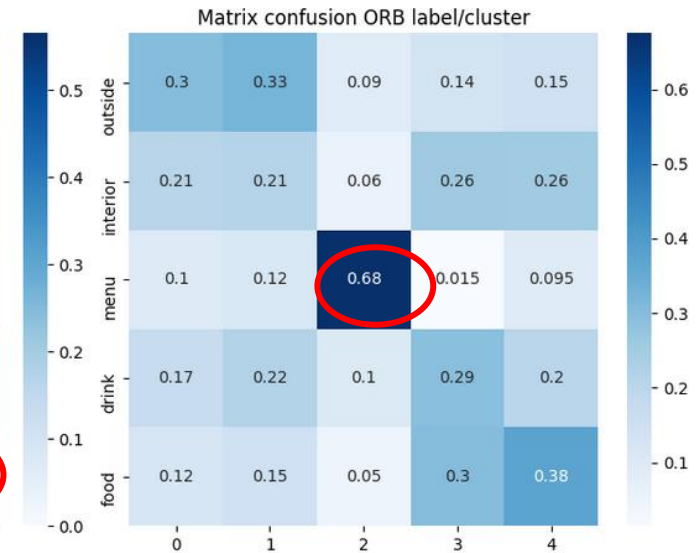
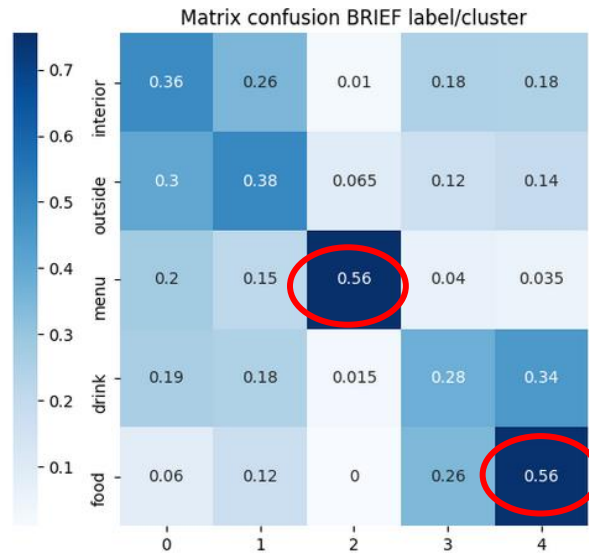
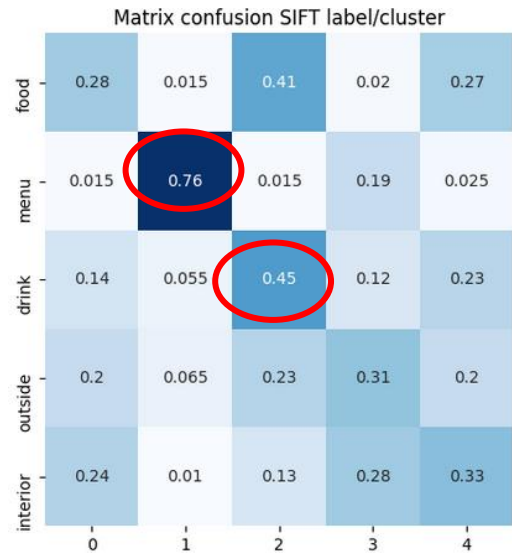
Accuracy (entre 37 et 43% des photos sont correctement classées)

et matrice de confusion (les menus s'en sortent mieux que les autres catégories)

sift		precision	recall	f1-score	support
	0	0.32	0.28	0.30	200
	1	0.84	0.76	0.79	200
	2	0.36	0.45	0.40	200
	3	0.33	0.31	0.32	200
	4	0.31	0.33	0.32	200
accuracy				0.42	1000
macro avg		0.43	0.42	0.43	1000
weighted avg		0.43	0.42	0.43	1000

brief		precision	recall	f1-score	support
	0	0.33	0.36	0.34	200
	1	0.34	0.38	0.36	200
	2	0.86	0.56	0.68	200
	3	0.32	0.28	0.30	200
	4	0.45	0.56	0.50	200
accuracy				0.43	1000
macro avg		0.46	0.43	0.44	1000
weighted avg		0.46	0.43	0.44	1000

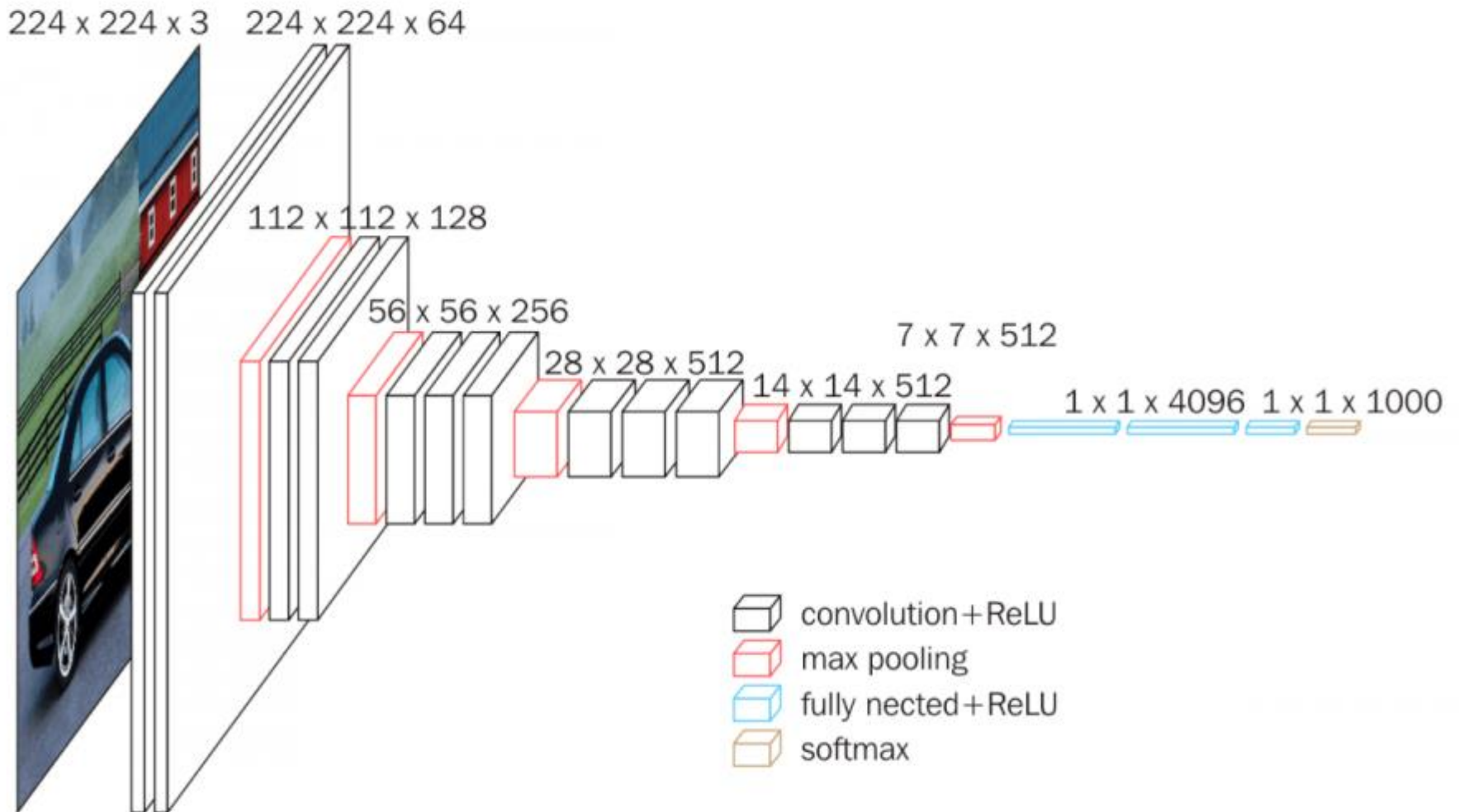
orb		precision	recall	f1-score	support
	0	0.33	0.30	0.31	200
	1	0.21	0.21	0.21	200
	2	0.69	0.68	0.68	200
	3	0.29	0.29	0.29	200
	4	0.35	0.38	0.36	200
accuracy				0.37	1000
macro avg		0.37	0.37	0.37	1000
weighted avg		0.37	0.37	0.37	1000



Voyons si les reseaux de neurones peuvent faire mieux: VGG16

- **En matière de reconnaissance d'images, quelques réseaux ont fait leur preuve au cours des 10 dernières années, grâce notamment au challenge imagenet « ILSVRVC »**
 - 2014 – VGG16 : 92,7% au top 5 test accuracy d'imagenet
 - 2014 – GoogleNet: 93,3% au top 5 test accuracy d'imagenet
 - 2015 – ResNet
 - 2015 – Inception V3
- **Pour nos besoins nous avons choisi de nous concentrer sur le reseau VGG16 qui est populaire:**
 - C'est un réseau de 16 couches et 140 millions de paramètres
 - On peut le diviser en 2 parties:
 - Les 13 premières couches constituées de 5 blocs de couches de convolution (filtrage)
 - Des couches supplémentaires de pooling après chaque bloc jouent le rôle de réducteur de dimension
 - Les 3 dernières qui sont les couches fully connected débouchant sur la classification
 - Il prend en entrée des images de 224x224x3
 - En sortie on récupère une matrice de N lignes/images K colonnes/proba d'appartenance à la classe K

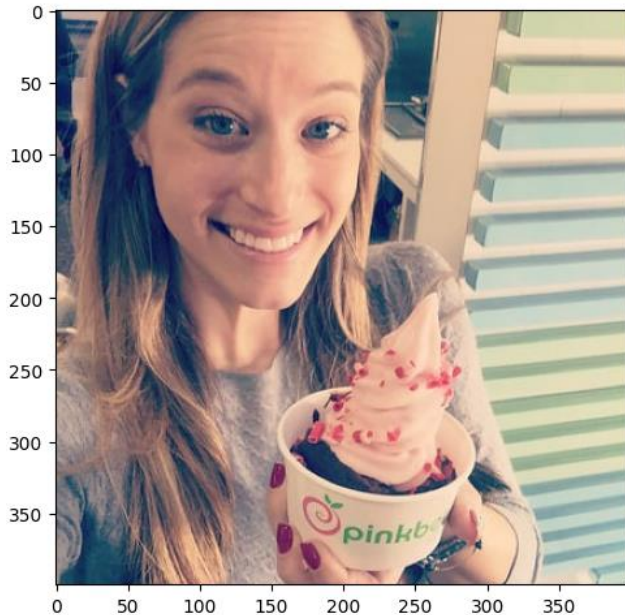
Architecture VGG 16



Test VGG 16 sur 1 ou 2 photos avec

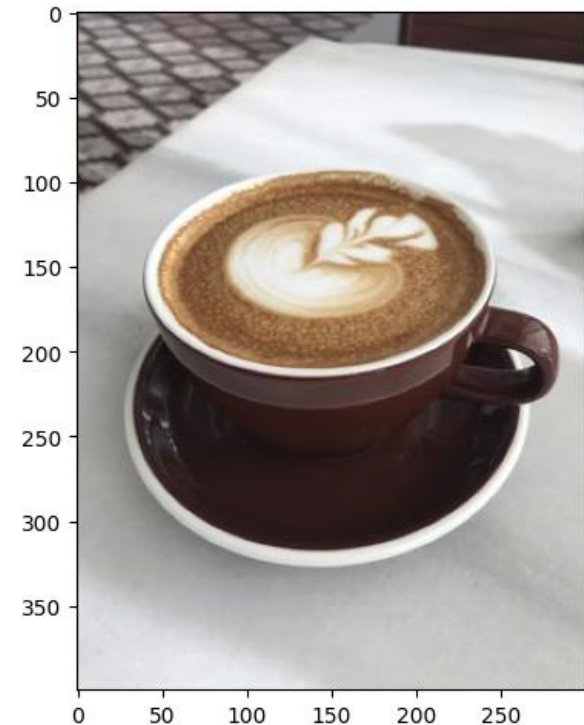


```
Top 3 : [('n07615774', 'ice_lolly', 0.16271389), ('n04357314', 'sunscreen', 0.12443831), ('n03476991', 'hair_spray', 0.11060534)]
```



Sucette glacée à 16%
Crème solaire à 12%

Expresso à 20%
Palet à 15% (soucoupe peut etre)



```
Top 3 : [('n07920052', 'espresso', 0.20633043), ('n04019541', 'puck', 0.15391573), ('n03259280', 'Dutch_oven', 0.10202522)]
```

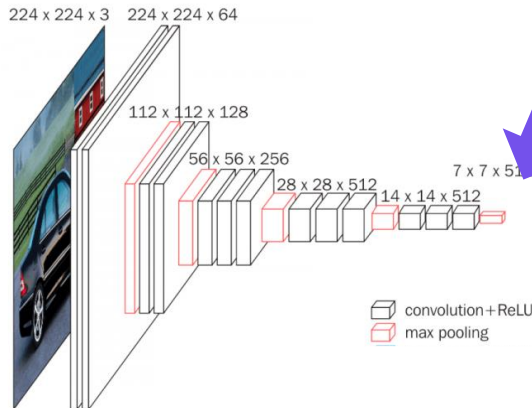
Traitement de 1000 photos avec extraction des features de VGG16 post convolutions via



VGG 16 PreProcessing de la collection des 1000 images

Extraction des features à la fin des blocs de convolution de VGG16:

On ajoute en sortie une couche de pooling max pour passer d'un array (1000,7,7,512) à une matrice (1000,512)



Les features sont recuperees à ce niveau là

Reduction du nombre de dimension du bag of features par PCA

En veillant à garder un pourcentage de variance expliquée important de l'ordre de 99% (17% de reduction de taille dans notre cas)

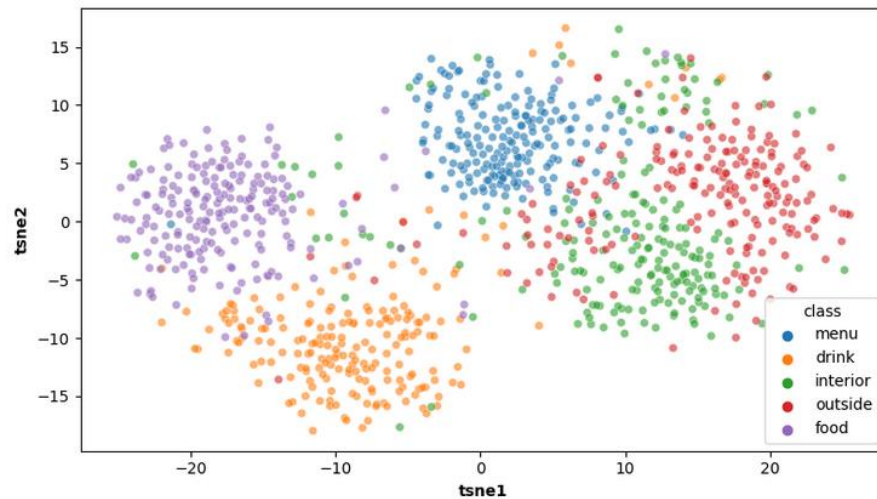
Visualisation 2D/3D par plongement TSNE

IMAGES: labels initiaux vs VGG16 features TSNE kmeans clustering (K=N labels initiaux)

Accuracy de 80%

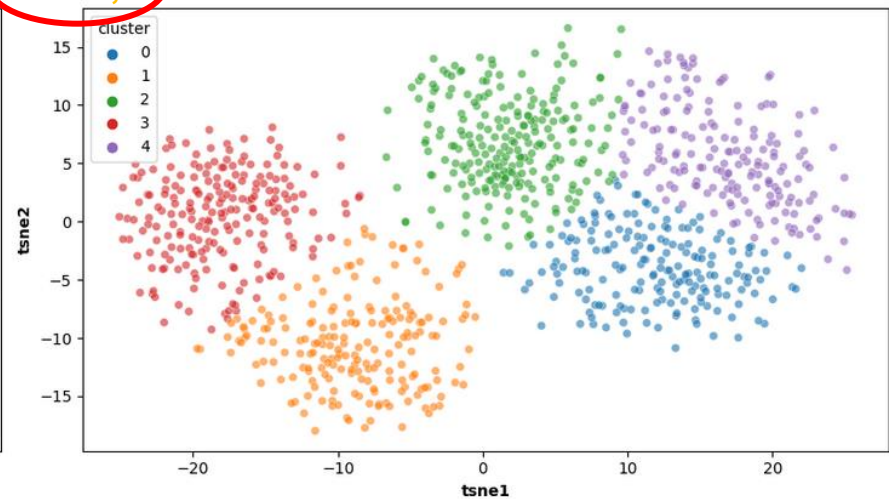
Diagonale limpide de la matrice de confusion : de bien meilleurs résultats que le SIFT

TSNE VGG16 features post convolution/post pca selon les labels initiaux

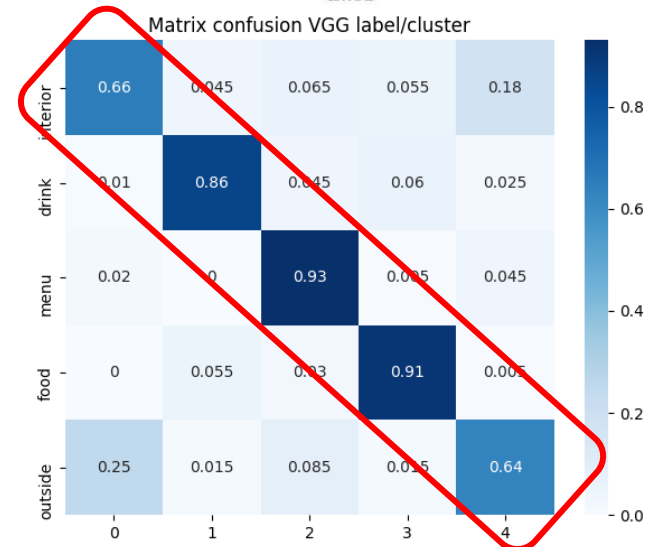


ARI=0,6

TSNE VGG16 post convolution/post pca/post 5 means



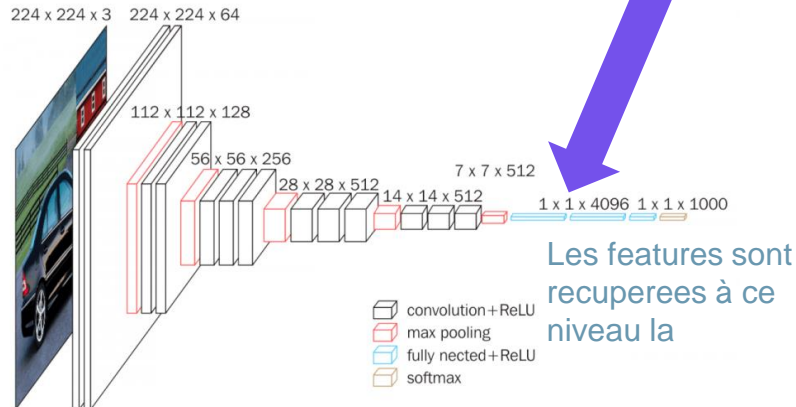
	precision	recall	f1-score	support
0	0.70	0.66	0.68	200
1	0.88	0.86	0.87	200
2	0.81	0.93	0.86	200
3	0.87	0.91	0.89	200
4	0.71	0.64	0.67	200
accuracy			0.80	1000
macro avg	0.79	0.80	0.79	1000
weighted avg	0.79	0.80	0.79	1000



Traitement de 1000 photos avec extraction des features de VGG16 après la première couche fully connected via

VGG 16 PreProcessing de la collection des 1000 images

Extraction des features après la première couche fully connected:
On récupère en sortie une matrice (1000,4096)



Reduction du nombre de dimension du bag of features par PCA

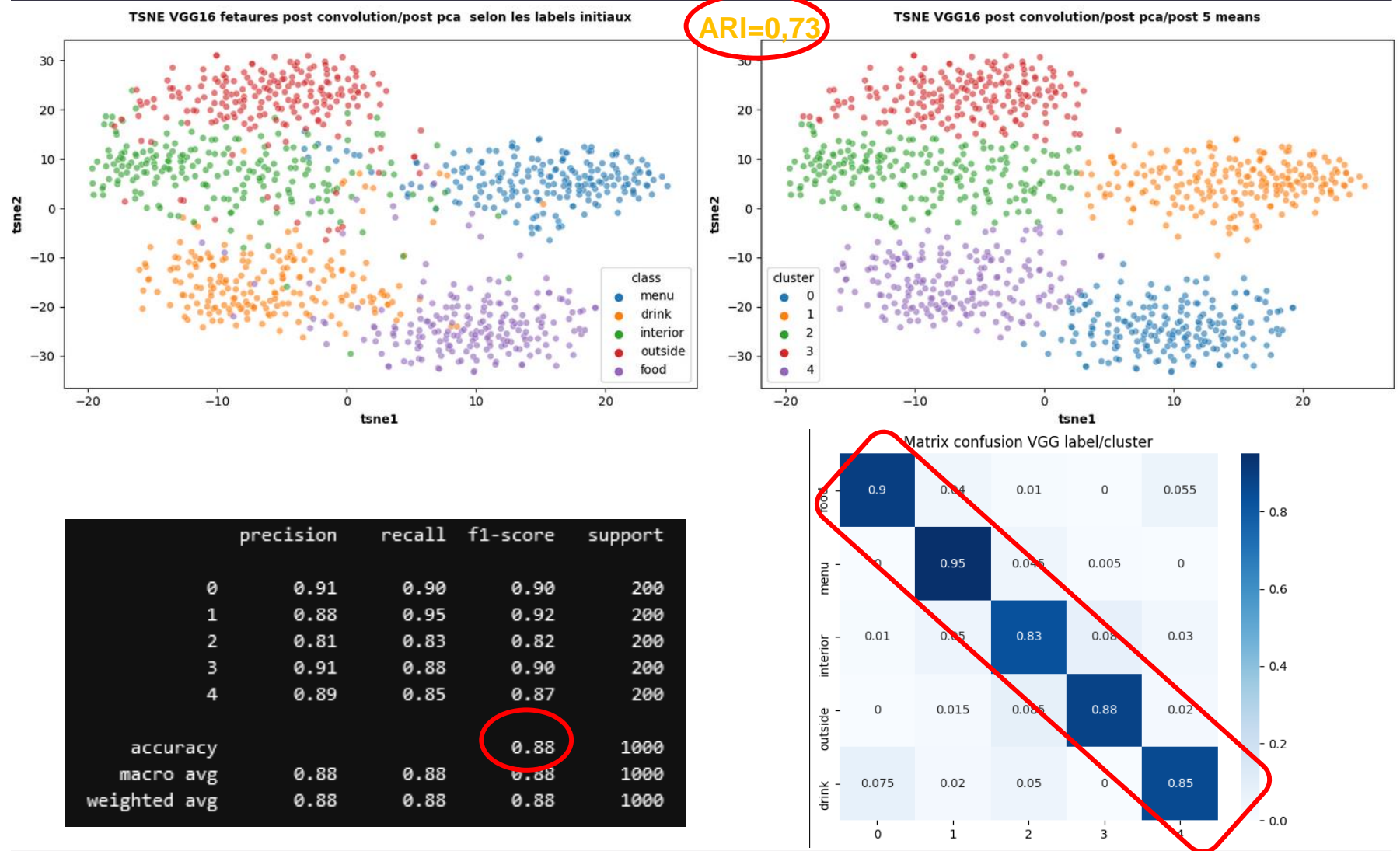
En veillant a garder un pourcentage de variance expliquée important de l'ordre de 99% (79% de reduction de taille dans notre cas : il reste 830 colonnes)

Visualisation 2D/3D par plongement TSNE

IMAGES: labels initiaux vs VGG16 features TSNE kmeans clustering (K=N labels initiaux)

Accuracy de 88%

Diagonale limpide de la matrice de confusion : Très bon clustering



VGG 16 Transfer Learning :

couches de convolutions identiques et non entrainables

Couches de classification bespoke et entrainables

VGG 16 PreProcessing de la collection des 1000 images

Couches de convolution VGG16 non entrainables : poids imagenet

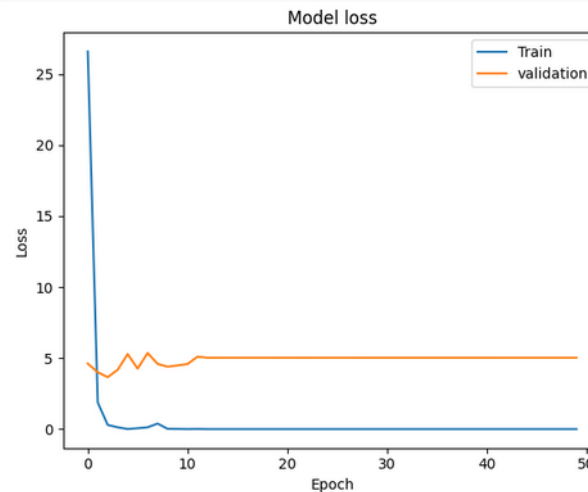
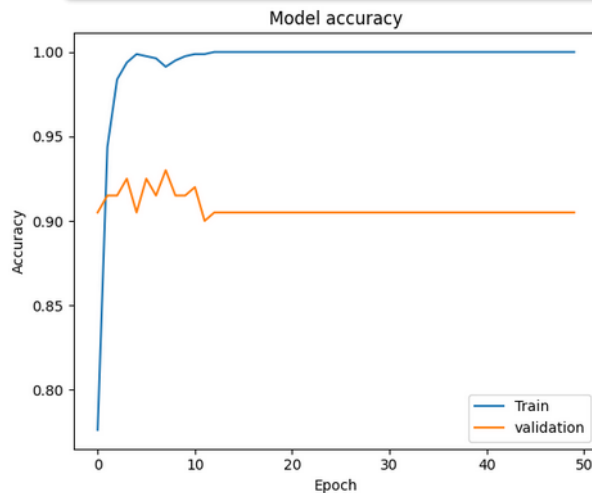
Couche flatten pour passer de 4D à 2D

Couche dense de 1024 neurones avec activation RELU

Couche dense de sortie 5 classes avec activation softmax (generalisation de la sigmoïde à plusieurs classes)

Compilation du VGG bespoke: 40 millions de paramètres dont 25 millions entrainables

Entrainement des couches entrainables sur 1000 photos du jeu d'entrainement avec validation split de 0,2 et 50 epochs



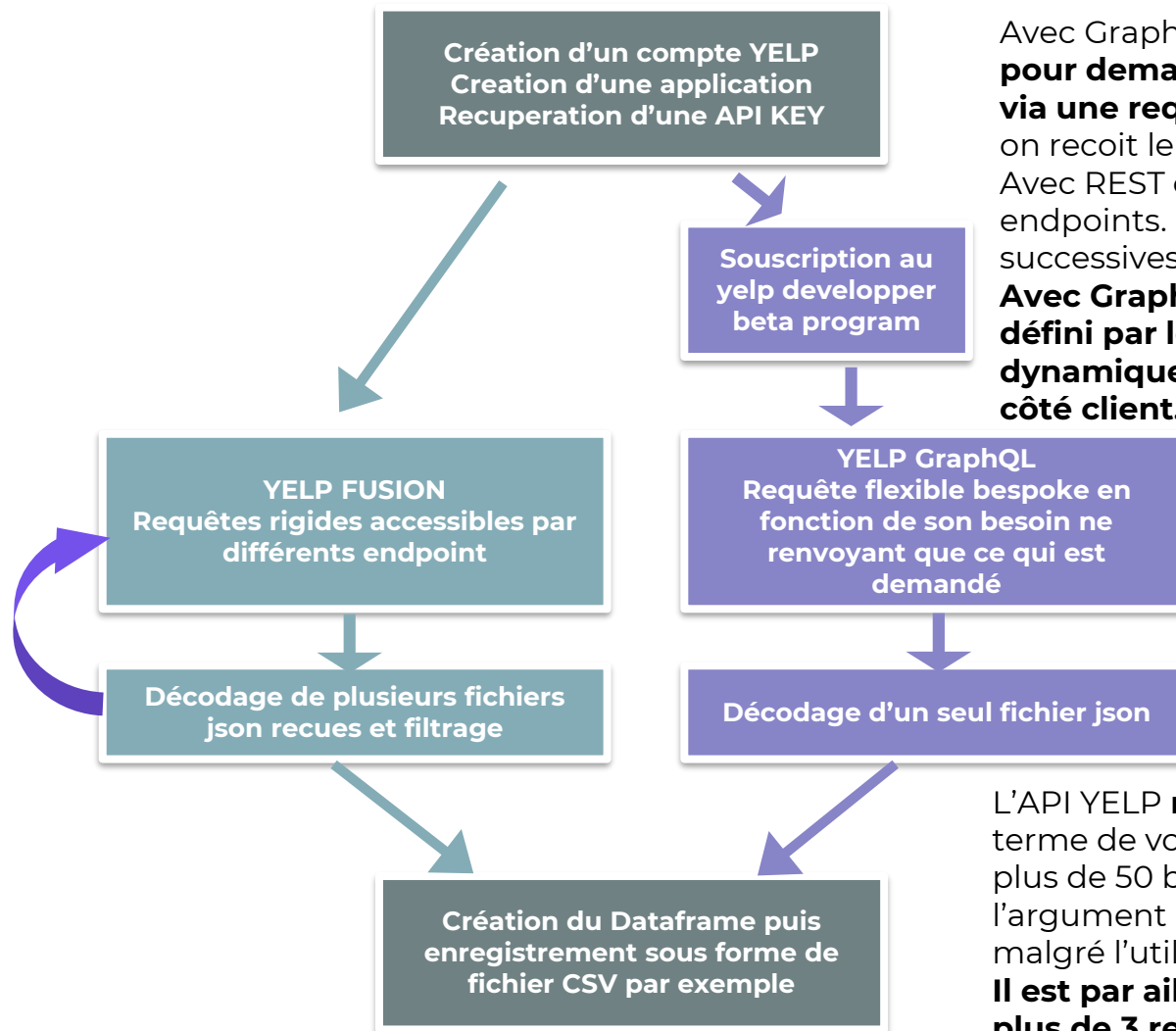
La Prédiction sur 1000 photos du jeu de test Donnera un ARI de 0,79 avec les vrais labels

D

**PROJET API:
COLLECTER UN
ECHANTILLON DE
DONNEES DE 200
RESTAURANTS VIA L'API
YELP**

API YELP

2 possibilités plus ou moins flexibles mais limitées



Avec GraphQL, Un **seul POST est nécessaire pour demander exactement ce que l'on veut via une requête GraphQL**. Le serveur gère, et on reçoit le payload complet.

Avec REST on recevait des objets définis par des endpoints. Nécessité de faire plusieurs requêtes successives pour récupérer son besoin

Avec GraphQL, on n'adapte pas un objet défini par le backend, mais on définit dynamiquement l'objet que l'on va recevoir côté client. Et ça, ça change tout

L'API YELP **reste relativement limitée** en terme de volume de donnée : Pour récupérer plus de 50 business, nous devons ruser avec l'argument offset en lançant plusieurs requêtes malgré l'utilisation de GraphQL, **Il est par ailleurs impossible de récupérer plus de 3 reviews par restaurant**

Partie data de la requête GraphQL

```
#boucle sur les offset pour recup plus de 50 requetes de buisness
for offset in range(0, 200, 50):
    datas = ""
```

```
{
  search(location: "New York City",
    categories: "restaurants",
    sort_by:"review_count",
    limit:50,
    offset:""+str(offset)+"") {
    total
    business {
      id
      name
      rating
      location {
        city
        country
        address1
      }
      price
      review_count
      reviews {
        id
        rating
        user{
          id
        }
        text
        time_created
      }
    }
  }
}
```

Filtre sur la ville de New York

Filtre sur la catégorie « restaurant »

Limite de résultat à 50 business par requete mais on ne peut pas aller au dela

Utilisation de la variable offset afin de recuperer 200 business differents avec 4 requetes successives de 50 resultats

Données que l'on souhaite recuperer, sur mesure , par rapport à notre besoin