Moteur de recommandations de films

RAPPORT D'ÉTUDE

Qu’est-ce qu’une bonne recommandation ?

Un site de cinéma nous a contacté
pour lancer un moteur de recommandation de films pour ses futurs clients :

un client indique un film
et le site lui recommande 5 autres films

Qu’attend un utilisateur en matière de recommandations de films ?

La réponse à cette question n’est pas triviale . Selon les utilisateurs, certains pourront attendre :

- des films le plus semblables possible à la référence qu’ils ont saisie : même directeur ? même acteurs ? avec quelle priorité ? Même note Imdb ? (même si le film de référence avait été plutôt mal noté ?), …..
- un panel de films variés, découvrir des nouveautés, des films bien notés, …

Pour savoir comment orienter notre système de recommandation , il faudrait pouvoir valider le modèle avec de vrais utilisateurs : s’ils cliquent sur la recommandation, c’est qu’elle est pertinente.

En attendant, nous devons choisir pour eux.

Pistes de recherche

Choix des variables

Pour ce projet, nous choisissons de recommander des films le plus semblables possibles, en utilisant presque toutes les variables disponibles et en leur donnant la même importance.

Données disponibles :

- directeur, acteurs, content_rating, genre, keywords
- budget, duration
- facenumber_in_poster
- language
- country
- title_year
- num_critic_for_reviews, num_voted_users, num_user_for_reviews
- Note imdb, gross (revenu),
- facebook likes des directeurs,, des acteurs, principaux, du casting complet, du film
- Aspect-ratio, color/NB

En fonction des résultats de la modélisation, nous verrons s'il est possible de conserver toutes ces variables où s'il est nécessaire d'en abandonner pour obtenir des meilleurs résultats

Modélisation

Notre démarche de modélisation est une **démarche non supervisée** puisque nous ne connaissons pas la réponse attendue par un utilisateur.

Pour identifier des films similaire à la référence choisie nous allons tester 2 familles d'algorithmes :

VOISINS

La recherche de **voisins,** à partir d'un film référence, l'algorithme recherche les films les plus proches de celui-ci en minimisant la distance par rapport au film référence. On utilisera une distance euclidienne calculée sur la totalité des variables disponibles. On pourra affecter des poids différents à ces variables. On utilisera l'algorithme **K-Nearest-Neighbors** de Scikit-Learn. Cette méthode à l'avantage de pouvoir préciser le nombre de voisins recherchés, ce qui correspond à notre objectif.

CLUSTERING

La décomposition de notre base de films en **clusters** à l'intérieur desquels une recommandation est possibles . on recommandera les films qui se trouvent dans le même cluster que le film de référence. On essaiera 2 algorithmes de clustering de Scikit-Learn : **K-Means** comme algorithme classique utilisées pour des applications variées, et **Agglomerative Clustering**, qui fonctionne différemment.

Evaluation

Dans cette démarche non supervisée, il n'y a pas une méthode évidente pour l 'évaluation des résultats de notre modélisation.

Nous utiliserons 2 méthodes :

- L' **évaluation empirique « à dire d'expert «** à partir d'un jeu d'essai de films. Il s'agit de consulter les recommandations et de d'évaluer « à dire d'expert » si elles sont pertinentes.

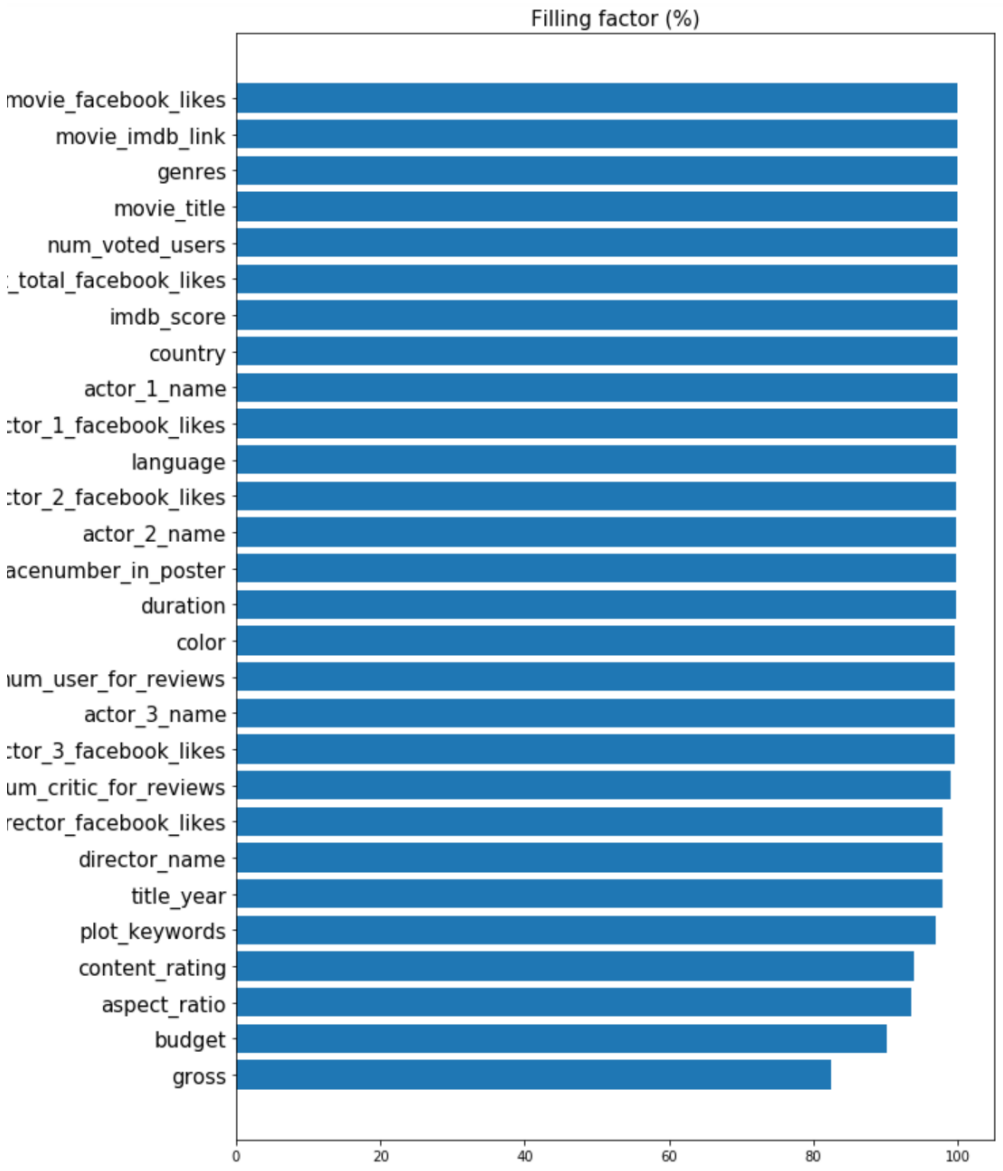
- **L'évaluation intrinsèque des modèles** avec : le « **score** » fournit pas scikit-learn pour chaque modèle, et , pour les modèles de clustering, le **coefficient de silhouette** et la **taille des clusters**.

Exploration et pre-processing

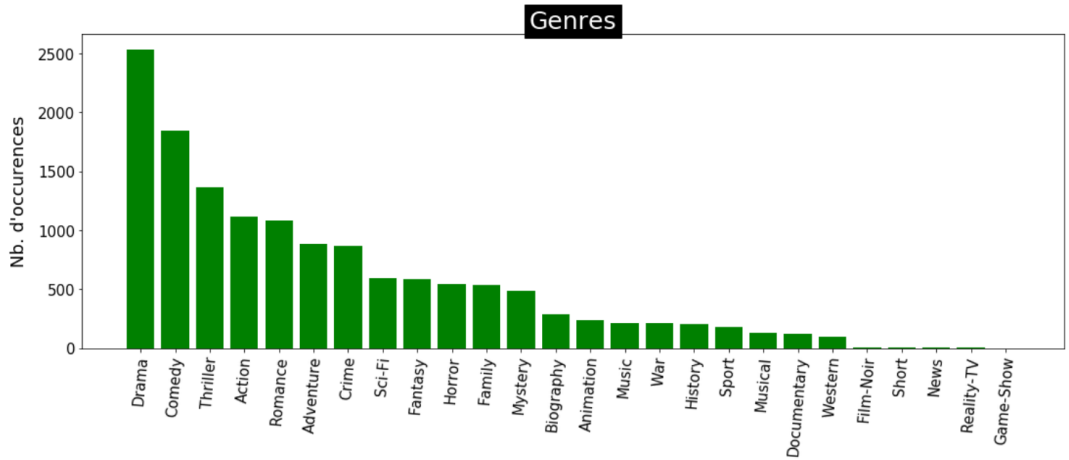
Complétude de la base de données initiale

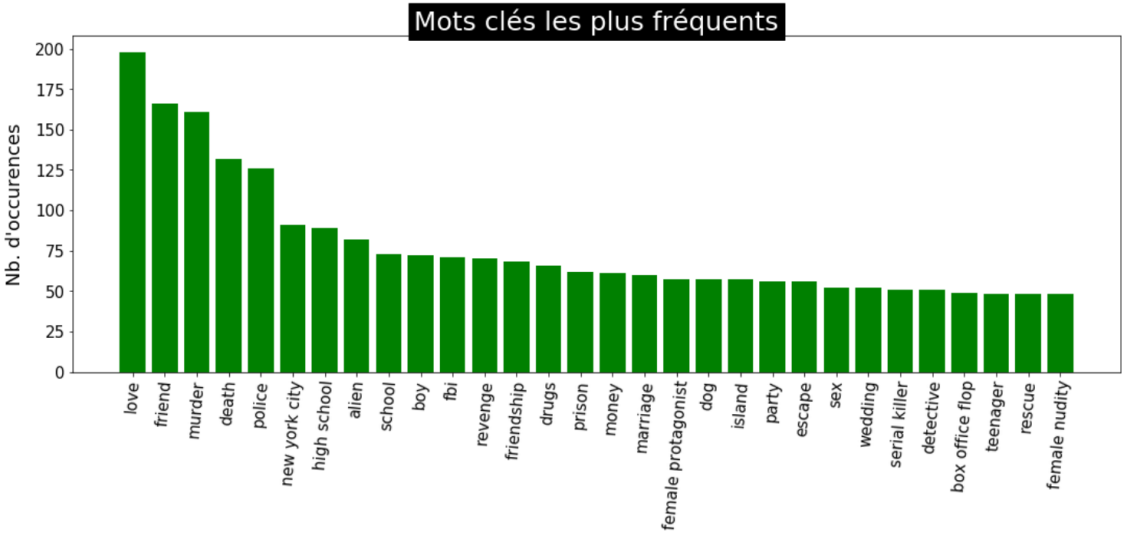
Toutes les variables ont un assez bon taux de complétude, supérieur à 80%.

On complètera les valeurs manquantes pour faciliter la modélisation.



Aperçu de quelques valeurs intéressantes





4917 films	**65 pays**
2398 directeurs	**47 langages**
6256 acteurs	**91 années**

Cleaning

Suppressions des doublons

Recherche de valeurs aberrantes

Imputation des valeurs manquantes

Regroupement des variables acteurs

Traitement d'un outlier

Normalisation des variables numériques entre 0 et 1

Suppression des doublons portant le **même titre de film**. A l'issue de cette suppression, il reste 4917 films dans la base.

Recherche de valeurs aberrantes

Imputation des valeurs manquantes pour les variables numériques.

Etant donné que les modèle fonctionnent mal ou pas lorsque des valeurs sont manquantes, nous remplaçons les valeurs manquantes par des valeurs « les moins fausses possibles » ici la **médiane**.

Regroupement des variables acteurs

Nous regroupons les variables acteur 1, acteur 2 et acteur 3 en faisant l'hypothèse que si un acteur apparaît en acteur 1 dans le film de référence, il ne faut pas exclure des recommandations un film dans lequel cet acteur jouerait en acteur 2.

Nous créons donc une nouvelle variable 'actors' qui concatène les 3 variables

Traitement d'un outlier

Avant de normaliser , vous avons traité l'outlier du budget mis en évidence lors de l'exploration.

Nous avons remplacé la valeur du plus fort budget par la valeur du budget le plus élevé suivant, multipliée par un coefficient de 1,1.

Normalisation

Nous avons **normalisé les variables numériques entre 0 et 1**

Encoding des variables catégorielles

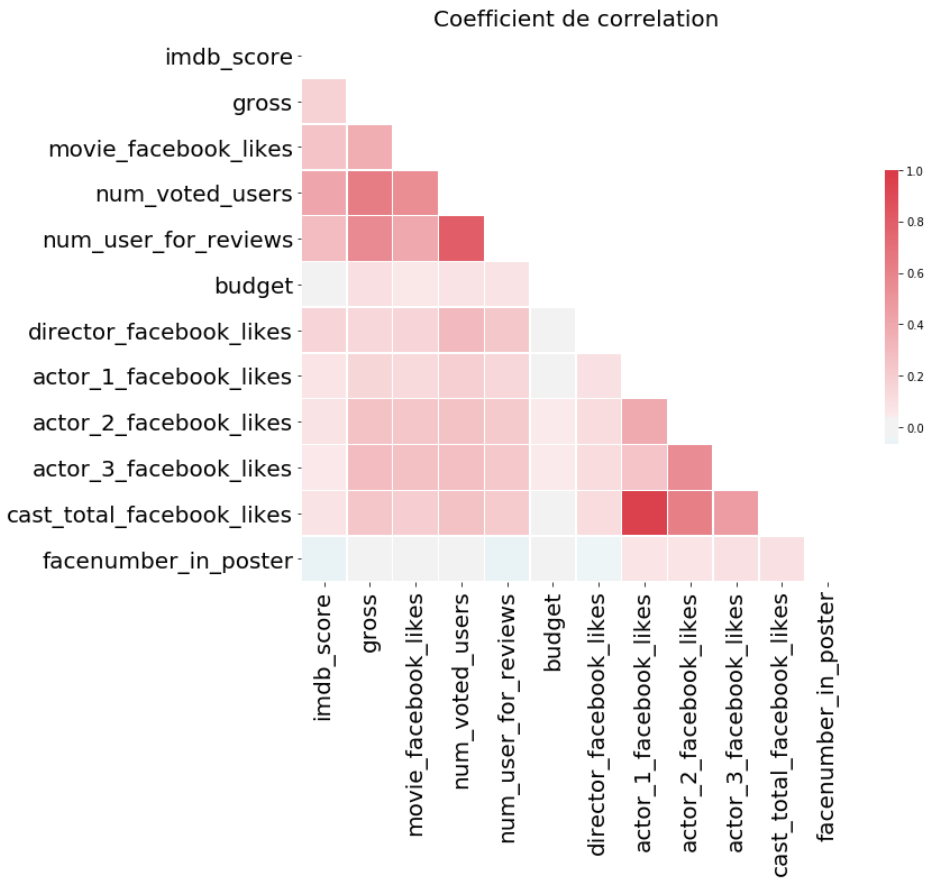
Encoding

Nous avons utilisé un encoding **OneHot** pour les variables : **Director_name**, **country, language**

Nous avons utilisé un encodage **Tfidf sans idf** pour les variables **genres, plot_keywords et actors**, car cet encoder permettait d'utiliser un tokenizer (séparation sur '|')

Pour la variable **content-rating**, nous avons réalisé un **encoding ordinal** « manuel »suivant la largeur du public concerné par le film.

Recherche de corrélations



On peut conserver seulement cast_total_facebook_likes et exclure les likes facebooks des acteurs 1, 2 et 3

Pistes de Modélisation

1 Baseline
2 Simplification du modèle
3 Recherche de paramètres

Baseline KNN

Une première modélisation a été réalisée avec :

Avec les variables

- Genre
- Directeur
- Acteurs
- Pays
- Langage
- keywords
- 'duration', 'director_facebook_likes,
- 'gross', 'num_voted_users', 'cast_total_facebook_likes', 'facenumber_in_poster',
- 'num_user_for_reviews','budget', 'title_year', 'imdb_score',
- 'movie_facebook_likes', 'content_rating', 'aspect_ratio’

Avec le modèle :

- K-Nearest Neighbours
- Les paramètres par défaut du modèle
- Un nombre de voisins de 6 (pour 5 recommandations car l’algorithme renvoie le film de référence parmi les résultats)

Baseline KNN : évaluation des résultats à dire d'expert

L'examen des recommandations pour 4 films donne :

- Des résultats corrects pour la similitude sur le titre, le genre, le langage (exemples : star wars, un documentaire, un film en japonais)

- Mais des **résultats notoirement insuffisants pour la similitude sur le directeur ou les acteurs** :
  - Pour Notting Hill par exemple, les recommandations ne contiennent aucun film du même directeur ou avec au moins un des acteurs principaux.

Baseline K-Mean

Une première modélisation a été réalisée avec :

Avec les variables

- Genre
- Directeur
- Acteurs
- Pays
- Langage
- keywords
- 'duration', 'director_facebook_likes,
- 'gross', 'num_voted_users', 'cast_total_facebook_likes', 'facenumber_in_poster',
- 'num_user_for_reviews','budget', 'title_year', 'imdb_score',
- 'movie_facebook_likes', 'content_rating', 'aspect_ratio’

Avec le modèle :

- K means
- Les paramètres par défaut du modèle
- 700 clusters pour viser une taille de 6 par cluster.

Baseline K-Mean : évaluation des résultats à dire d'expert

Même constat qu'avec le KNN :

L'examen des recommandations pour 4 films donne :

- Des résultats corrects pour la similitude sur le titre, le genre, le langage (exemples : star wars, un documentaire, un film en japonais)

- Mais des **résultats notoirement insuffisants pour la similitude sur le directeur ou les acteurs** :
  - Pour Notting Hill par exemple, les recommandations ne contiennent aucun film du même directeur ou avec au moins un des acteurs principaux.

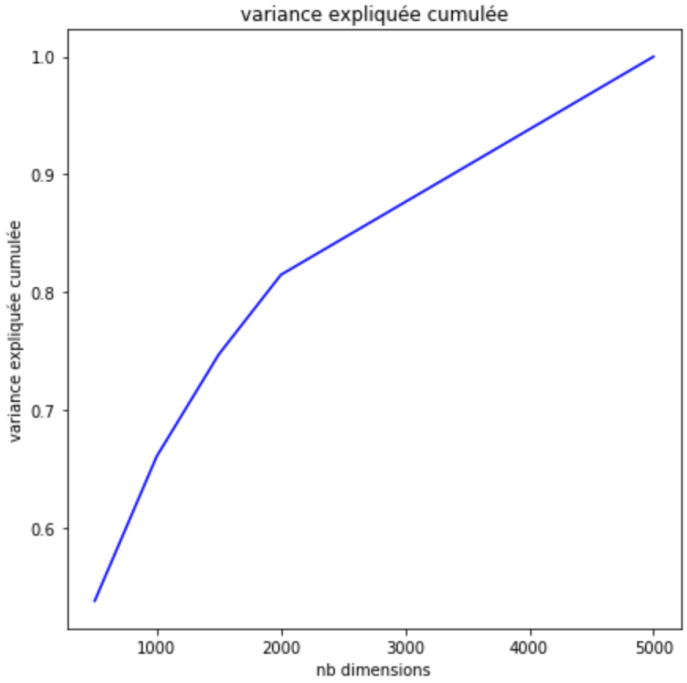
Pistes de Modélisation

1 Baseline
2 Simplification du modèle
3 Recherche de paramètres

Réduction de dimension par

SINGLE VALUE DECOMPOSITION

Après encoding de toutes les variables catégorielles, la base de données contient **16890 colonnes.**



On passer de 16890 variables à 2000 features en conservant 80% de la variance expliquée.

Réduction de dimension par filtrage des valeurs

Après encoding de toutes les variables catégorielles, la base de données contient **16882 colonnes.**

SVD permet de ramener à 2000 dimensions, ce qui reste élevé pour une base de 5000 films.

Nous avons donc étudié une stratégie complémentaire pour réduire la dimension.

Pour chaque variable catégorielle, **ne conserver que les valeurs utilisées pour plus de n films, c'est-à-dire remplacer par None l**es valeurs utilisées pour moins de n films

La conséquence est que lors de l'encodage OneHot, le nombre de colonnes résultantes est plus faible

Et nous avons fait varier n . Plus n est grand, plus la force de filtrage est forte, plus on perd d'information mais plus on réduit la dimension.

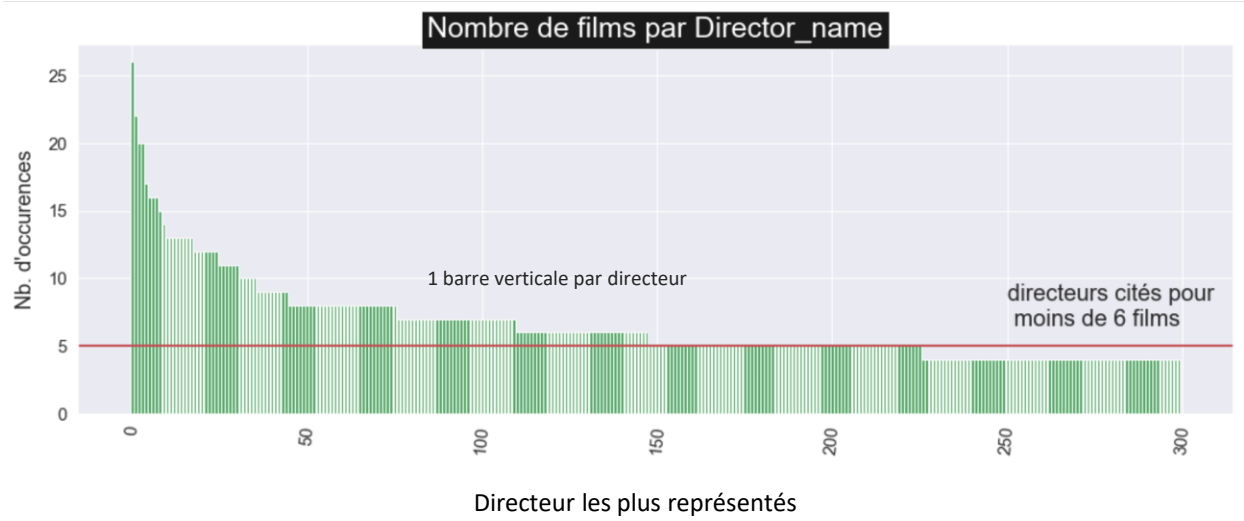
Nous avons étudié les paramètres suivants :

2 : valeur utilisée pour au moins 2 films

6 : valeur utilisée pour au moins 6 films

Valeurs des variables utilisées pour au moins n films, illustration

Exemple pour Director_name



Abandon des mots clés

Les variables keywords apportent une complexité trop grande au modèle. Une piste de réduction de la dimension serait de pratiquer :

une lemmisation (regroupement des mots clés de racine identique)

et un word embedding (réduction de dimension dans un espace de vecteurs sémantiques)

Nous n'avons mis en œuvre ces pistes qui seront étudiées davantage dans le projet Stackoverflow

Nous avons donc décidé d'exclure les plot_keywords de notre modélisation.

Pistes de Modélisation

1. Baseline
2. Simplification du modèle
3. Recherche de paramètres

Optimisation du clustering:

recherche de paramètres

Nous avons procédé à une recherche des meilleurs paramètres parmi les options suivantes :

- **Algorithme** : K-Means, Agglomerative Clustering

- **Niveaux de filtrage des valeurs** pour les variables catégorielles :

  Pour chaque variable catégorielle on conserve avant encodage seulement les valeurs qui sont présentes pour au moins **n** films. Avec n parmi [1, 2, 6]

- **Nombre de clusters** : [100, 200, 300, 400]

Optimisation du clustering : recherche de parametres

```
n_frequency : 1, n_clusters : 100 , model : km - nb de features = 2630 - silhouette_score = 0.04097501801565403
n_frequency : 1, n_clusters : 100 , model : ag - nb de features = 2630 - silhouette_score = 0.022056028308632993
n_frequency : 1, n_clusters : 200 , model : km - nb de features = 2630 - silhouette_score = 0.05537790372524769
n_frequency : 1, n_clusters : 200 , model : ag - nb de features = 2630 - silhouette_score = 0.042105775704098174
n_frequency : 1, n_clusters : 300 , model : km - nb de features = 2630 - silhouette_score = 0.06315649959006447
n_frequency : 1, n_clusters : 300 , model : ag - nb de features = 2630 - silhouette_score = 0.05987388127516744
n_frequency : 1, n_clusters : 400 , model : km - nb de features = 2630 - silhouette_score = 0.06690172044694036
n_frequency : 1, n_clusters : 400 , model : ag - nb de features = 2630 - silhouette_score = 0.07352089759527637
n_frequency : 2, n_clusters : 100 , model : km - nb de features = 1438 - silhouette_score = 0.05739608188637543
n_frequency : 2, n_clusters : 100 , model : ag - nb de features = 1438 - silhouette_score = 0.0334454852386158
n_frequency : 2, n_clusters : 200 , model : km - nb de features = 1438 - silhouette_score = 0.0746082787020628
n_frequency : 2, n_clusters : 200 , model : ag - nb de features = 1438 - silhouette_score = 0.06451340004285028
n_frequency : 2, n_clusters : 300 , model : km - nb de features = 1438 - silhouette_score = 0.09087461797810369
n_frequency : 2, n_clusters : 300 , model : ag - nb de features = 1438 - silhouette_score = 0.08734677048337557
n_frequency : 2, n_clusters : 400 , model : km - nb de features = 1438 - silhouette_score = 0.10037137712093268
n_frequency : 2, n_clusters : 400 , model : ag - nb de features = 1438 - silhouette_score = 0.10571957873851455
n_frequency : 6, n_clusters : 100 , model : km - nb de features = 311 - silhouette_score = 0.1657557926380957
n_frequency : 6, n_clusters : 100 , model : ag - nb de features = 311 - silhouette_score = 0.1511178926897437
n_frequency : 6, n_clusters : 200 , model : km - nb de features = 311 - silhouette_score = 0.21375189744114317
n_frequency : 6, n_clusters : 200 , model : ag - nb de features = 311 - silhouette_score = 0.20685339917812245
n_frequency : 6, n_clusters : 300 , model : km - nb de features = 311 - silhouette_score = 0.22452831321824257
n_frequency : 6, n_clusters : 300 , model : ag - nb de features = 311 - silhouette_score = 0.23876049723856535
n_frequency : 6, n_clusters : 400 , model : km - nb de features = 311 - silhouette_score = 0.24789007118322204
n_frequency : 6, n_clusters : 400 , model : ag - nb de features = 311 - silhouette_score = 0.25985163789182747
```

Légende :

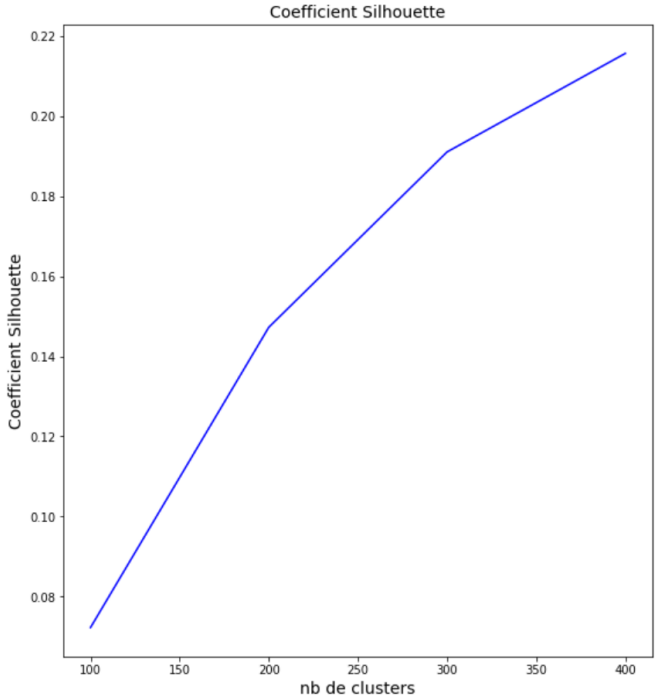
n_frequency : pour n les valeurs utilisées pour moins de n fils sont remplacées par None, non encodées en OneHot

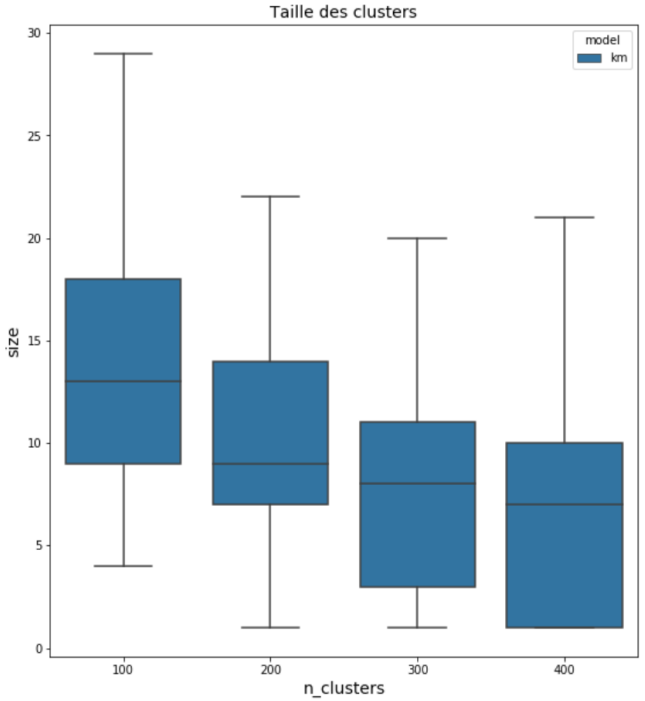
n_clusters : nb de clusters

Model : km = Kmean, ag = Agglomerative Clustering

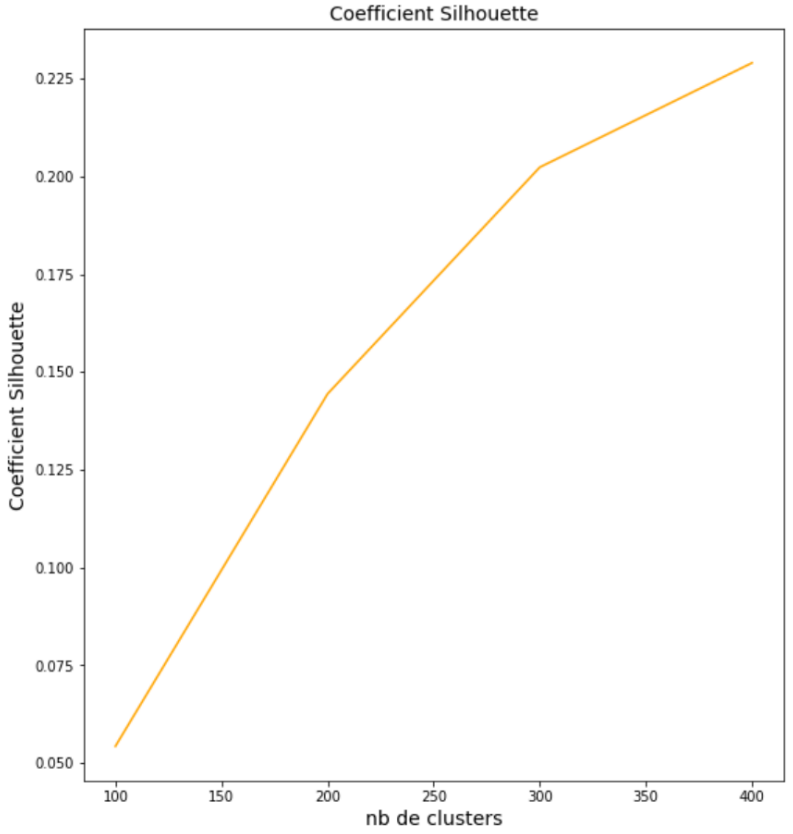
Nb de features : nb de colonnes après filtrage des valeurs de niveau n_frequency puis encoding

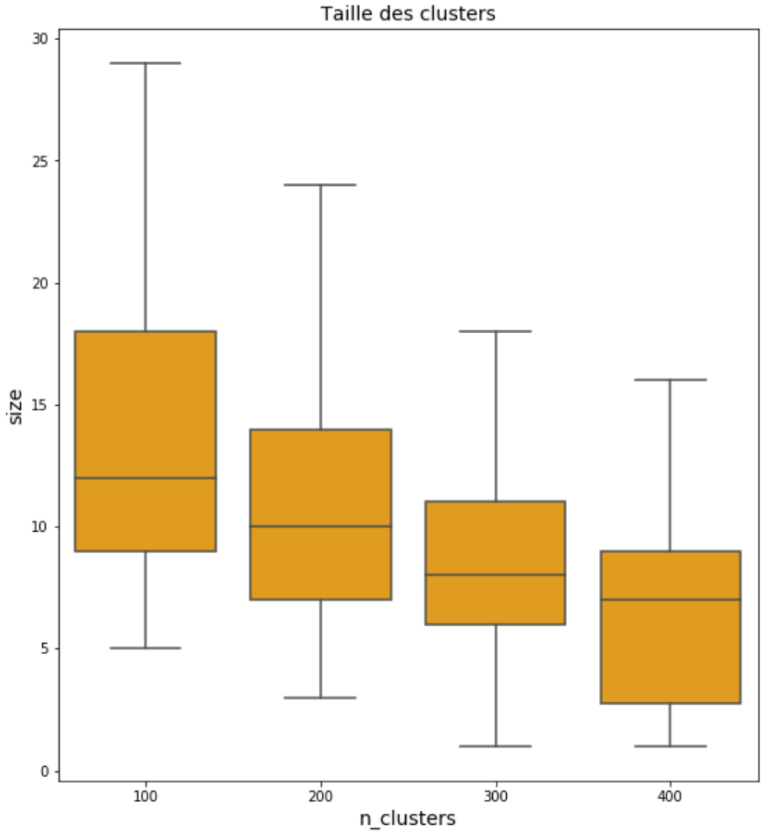
Optimisation du clustering : résultats pour K-Mean (pour n_frequency = 6)



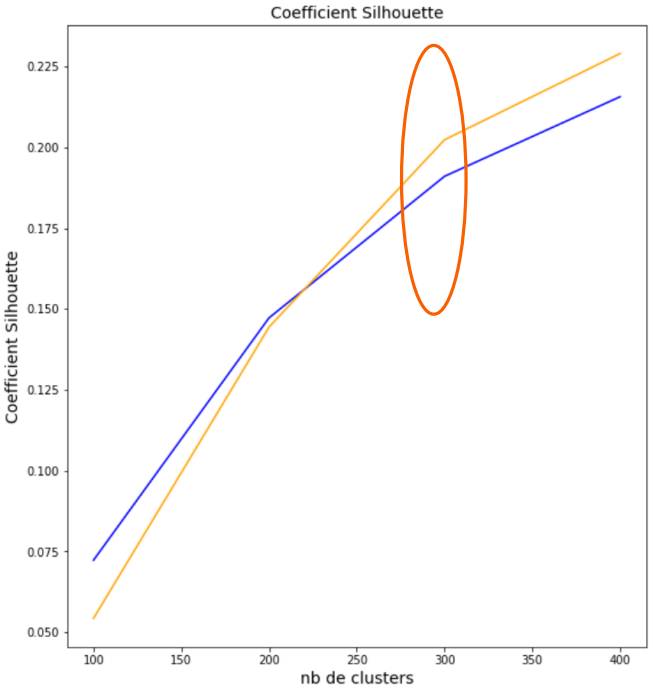


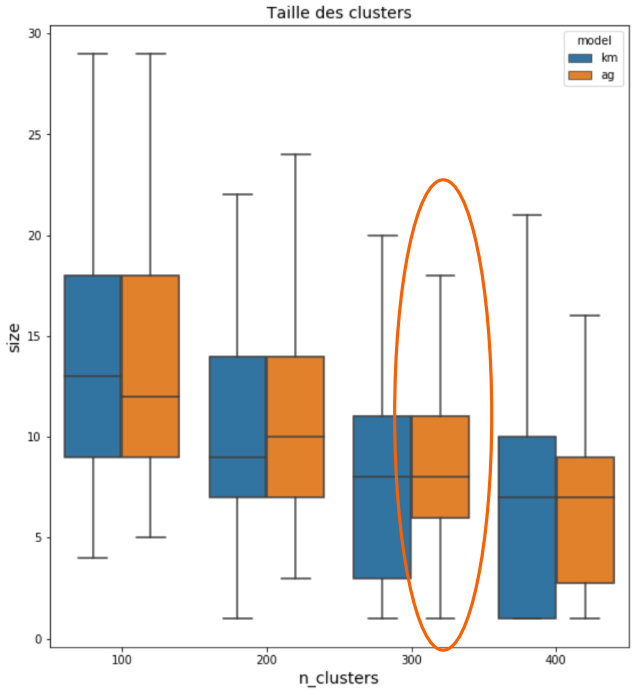
Optimisation du clustering : résultats pour Agglomerative clustering (pour n_frequency = 6)





Comparaison des 2 modèles (pour n_frequency = 6)





CONCLUSION

Le score de silhouette augmente avec le nb de clusters

Les scores de silhouette ne sont pas significativement différents entre les 2 modèles.

Agglomerative clustering offre des tailles de clusters plus régulières et suffisantes

Avec 300 clusters on pourra recommander au moins 5 films dans 75% des cas.)

MODÈLE RETENU

Variables

- abandon des mots clés
- suppression des valeurs présentes pour moins de 6 films

Modèle

Agglomerative Clustering
300 clusters

Une démo

de notre système de recommandation

est accessible via une API

API Endpoint

```
http://recommender.machinelearn.it
```

Example Request

```
GET http://recommender.machinelearn.it/recommend/{ID-FILM}
```

Example Response

```
{
"_results": [
{"id": 320, "name": "Mad Max Beyond Thunderdome"},
{"id": 1023, "name": "Mad Max 2: The Road Warrior"},
{"id": 2497, "name": "Mad Max: Fury Road"},
{"id": 2682, "name": "Babe: Pig in the City"},
{"id": 4047, "name": "Mad Max"}
]
}
```

Quelques résultats

Star Wars: Episode III - Revenge of the Sith

Avatar
Highlander: Endgame
Star Wars: Episode IV - A New Hope
Star Wars: Episode V - The Empire Strikes Back
Beastmaster 2: Through the Portal of Time

Notting Hill

The Importance of Being Earnest
Pretty Woman
Cheri
The Other End of the Line
Full Frontal

Sea Rex 3D: Journey to a Prehistoric World

March of the Penguins
A Lego Brickumentary
Indie Game: The Movie
51 Birch Street
Winged Migration

Princess Mononoke

Howl's Moving Castle
Spirited Away
Ponyo

Merci de votre attention
Bonne séance de cinéma !

contact : brigitte.maillere@MachineLearn.it