Examen – Python pour les Humanités

Jean Barré et Alexandre Lionnet - Janvier 2026

Exercice : Analyse des textes et des personnages 10 points

L'objectif de cet exercice est d'analyser les relations stylistiques entre des romans et leurs personnages à l'aide de représentations de type *Bag of Words* et de mesures de similarité vectorielle. Vous testerez chacune de vos fonctions dans votre notebook avant de les intégrer dans un module Python.

Contexte
Vous disposez de deux types de fichiers issus de 10 romans différents :

- **10 fichiers** `.txt` : chaque fichier contient le texte complet d'un roman ;
- **10 fichiers** `.book` : fichiers au format `json` contenant des informations sur les personnages détectés dans les romans (genre, nombre de mentions, verbes associés).

Pour cet exercice, vous vous limiterez aux **5 personnages les plus importants** de chaque roman.

Version courte :
Créer un module Python capable de :

- extraire les 100 mots les plus fréquents associés aux personnages, tous romans confondus ;
- calculer des représentations *Bag of Words* pour les romans, les genres et les personnages ;
- calculer la distance cosinus entre ces représentations ;
- sauvegarder l'ensemble des résultats sous forme de fichiers CSV.

La ligne de commande suivante doit réaliser l'ensemble du travail :

```
python3 examen.py -ia chemin_dossier_txt -ib chemin_dossier_book -oa BoW_100_romans.csv
-ob BoW_100_genres.csv -oc BoW_100_personnages.csv -od distances_personnages_romans.csv
-oe distances_personnages_genres.csv
```

Version longue :

- Créer une fonction qui parcourt l'ensemble des fichiers `.book` et construit une liste des **100 mots les plus fréquents** associés aux personnages, tous romans confondus (vous pouvez vous limiter aux 5 personnages les plus importants par roman). Dans cet examen, les mots des personnages sont l'ensemble des lemmes présents dans les champs 'agent', 'patient', 'mod', 'poss' du dictionnaire. Chaque .book est à ouvrir comme un fichier json, qui comporte une liste de dictionnaire de personnage, listé du premier personnage au dernier.

- À partir de cette liste de 100 mots, calculer la **fréquence relative** de ces mots pour chaque roman, construire un `DataFrame` Pandas (10 lignes = 10 romans, 100 colonnes = 100 mots) et sauvegarder le résultat sous le nom :

  `BoW_100_romans.csv`

- À l'aide du fichier `genre_labels.json`, calculer le **vecteur moyen** pour chacun des deux genres romanesques (`policier` et `sentimental`), regrouper ces vecteurs dans un `DataFrame` à deux lignes, et sauvegarder le résultat sous le nom :

  `BoW_100_genres.csv`

- Pour chaque roman, calculer un *Bag of Words* (basé sur les mêmes 100 mots) pour chacun des **5 personnages les plus importants**, regrouper ces résultats dans un `DataFrame` et sauvegarder le résultat sous le nom :

  `BoW_100_personnages.csv`

- Calculer la **distance cosinus** :

  - entre le *Bag of Words* du roman et celui de chacun de ses personnages ;
  - entre le *Bag of Words* moyen du genre romanesque et celui de chacun des personnages.

  Sauvegarder les matrices de distances résultantes sous les noms :

  `distances_personnages_romans.csv`
  `distances_personnages_genres.csv`

- Visualiser les deux matrices de distances à l'aide d'une **heatmap** (avec `matplotlib` ou `seaborn`). Pour chaque roman, identifier le personnage le plus proche du roman puis du genre, et commenter brièvement ces résultats : que signifie ici la notion de proximité ? Que mesure réellement la distance cosinus ?

- **Bonus :** proposer une autre visualisation pertinente et justifier votre choix.

- Implémenter un module Python `examen.py` qui utilise les fonctions précédemment définies et sauvegarde l'ensemble des fichiers produits. La ligne de commande suivante doit s'exécuter correctement :

  ```
  python3 examen.py -ia chemin_dossier_txt -ib chemin_dossier_book -oa BoW_100_romans.csv
  -ob BoW_100_genres.csv -oc BoW_100_personnages.csv -od distances_personnages_romans.csv
  -oe distances_personnages_genres.csv
  ```

Consignes générales

- **Rendu** : le devoir devra être envoyé sous la forme d'un **unique fichier** `.zip` contenant :
  - le notebook de l'examen ;
  - le script Python `examen.py` ;
  - les données fournies et les fichiers produits.
- **Notebook** :
  - le notebook doit s'exécuter entièrement **sans erreur** (il est fortement conseillé de redémarrer le kernel et de ré-exécuter toutes les cellules avant l'envoi) ;
  - chaque fonction doit être accompagnée d'au moins une **cellule de test**, sauf si elle est directement utilisée dans une autre fonction ;
  - le notebook rendu ne doit pas être un brouillon : les cellules inutiles ou redondantes devront être supprimées.
- **Structure** :
  - chaque étape de l'exercice devra être clairement identifiée, idéalement à l'aide de cellules en texte (Markdown) ;
  - les fonctions devront être documentées (commentaires ou `docstrings`) et utiliser des noms explicites.
- **Aide et évaluation** :
  - si vous êtes bloqué sur une étape intermédiaire nécessaire pour poursuivre l'examen, vous pouvez demander les données correspondantes ; tous les points ne seront toutefois pas attribués à la question concernée ;
  - si vous ne parvenez pas à implémenter une solution en Python mais que vous comprenez la démarche, vous êtes encouragé à **décrire précisément en français** la solution envisagée (raisonnement, références, documentation consultée).
- **Ressources** : les documentations, forums et outils d'assistance (y compris les LLMs) sont autorisés, à condition de ne pas copier une solution sans la comprendre. Toute source utilisée devra être explicitement mentionnée.
- **Restriction** : Les fonctions lambda ne sont pas autorisées.

Remarque finale : l'évaluation portera autant sur la **qualité du raisonnement** et de la démarche que sur l'obtention d'un résultat correct. Toute tentative sérieuse et explicitée sera valorisée.

Bonne chance, et n'hésitez pas à commenter votre notebook pour faciliter la compréhension de votre démarche !