TP : Création d'une IA Générative de Texte From Scratch en Python

🎯 Objectifs Pédagogiques

Dans ce TP, nous allons créer un modèle génératif de texte à partir de zéro en utilisant des méthodes statistiques et du machine learning.

Ce que vous apprendrez :

1. Comprendre les modèles génératifs de texte et leurs applications.

2. Prétraiter un dataset textuel pour l'entraînement.

3. Créer un modèle Markov et un modèle LSTM pour générer du texte.

4. Expérimenter avec des paramètres pour améliorer la génération.

📌Partie 1 : Introduction aux modèles génératifs

📝 Questions en texte libre

Répondez aux questions suivantes avant de commencer l'implémentation :

1. Qu'est-ce qu'un modèle génératif de texte et comment fonctionne-t-il ?

Cela consiste à produire automatiquements des textes qui sont cohérents et significatif. Ce processus fait appel au traitement automatique de la langue , au machine learning, au deep learning et au traitement automatique du langage naturel ainsi qu'à d'autres algorithmes d'apprentissage profond (source : IBM).

2. Quelle est la différence entre un modèle probabiliste (ex. : Markov) et un modèle neuronal
(ex. : LSTM) ?

Le modèle Markov repose sur des probabilités de transition d'un état à un autre avec une mémoire limitée du passé immédiat, or le modèle neuronal comme LSTM (long short term memory) repose sur des réseaux neuronaux pour traiter des séquences de données avec des capacités d'apprentissage profond et une mémoire plus étendues.

3. Dans quels domaines les modèles génératifs sont-ils utilisés ?

Les modèles génératifs sont utilisés dans plusieurs domaines comme :

- L'art (génération d'image).
- L'archéologie(reconstruction faciale).
- La médecine (génération de d'images médicales pour l'entraînement des modèles de détection).
- Simulation et modélisation(générations de données simulées dans des scénarios complexes, comme la modélisation de phénomènes physiques ou économiques).
- Criminologies (reconstructions faciales de cadavres calcinés ou de la pigmentation de la peau).

4. Quels défis peuvent apparaître dans la génération de texte automatique ?

Les défis sont :

- La cohérence et la fluidité
- La création de contenu originale
- Le contrôle de la sortie
- La compréhension du contexte

🗄️ Partie 2 : Préparation des données

Nous allons utiliser un dataset plus évolué, le corpus de contes de Grimm, pour entraîner notre modèle génératif.

📥 2.1. Téléchargement et chargement des données

📌 Exercice : Téléchargez un dataset textuel et chargez-le en Python.

Exemples:
https://www.innovatiana.com/post/best-datasets-for-text-classification
https://paperswithcode.com/datasets?task=text-classification

📌 Question :

• Quels types de prétraitements sont nécessaires avant d'utiliser un texte pour entraîner un modèle génératif ?

🛠️ 2.2. Prétraitement des données

📌 Exercice : Nettoyez le texte en supprimant la ponctuation et en mettant tout en minuscule.

```
import pandas as pd
```

```
import re
from collections import defaultdict
import string
import random

df = pd.read_excel("grimms_tale.xlsx")



def nettoyer_texte(texte):
    if isinstance(texte, str):
        texte = texte.lower()
        texte = texte.translate(str.maketrans("", "",
string.punctuation))
    return texte


df["Story"] = df["Story"].apply(nettoyer_texte)
```

🎲 Partie 3 : Génération de texte avec un modèle de Markov

Nous allons entraîner un modèle de Markov pour générer du texte en se basant sur les probabilités de transition entre les mots.
En savoir plus : https://datacorner.fr/markov/

🛠️ 3.1. Construction du modèle de Markov

📌 Exercice : Construisez une table de transition où chaque mot est associé aux mots qui le suivent.

```

```

```python
def table_transitions(texte):
    mots = texte.split()
    table = defaultdict(list)

    for i in range(len(mots)-1):
        table[mots[i]].append(mots[i+1])
    return dict(table)


tables_transitions = {}

for index, row in df.iterrows():
    if isinstance(row["Story"], str):
        tables_transitions[row["Title"]] =
table_transitions(row["Story"])
```

📝 3.2. Génération de texte

📌 Exercice : Utilisez le modèle pour générer une phrase de 20 mots.

📌 Questions :

```python
def generer_phrase(tables_transitions, longueur_phrase=20):
    titre = random.choice(list(tables_transitions.keys()))
    transitions = tables_transitions[titre]


    mot_depart = random.choice(list(transitions.keys()))

    phrase = [mot_depart]


    while len(phrase) < longueur_phrase:
        mot_suivant = random.choice(transitions[phrase[-1]])
        phrase.append(mot_suivant)

    return ' '.join(phrase)


phrase_generee = generer_phrase(tables_transitions, 20)
print(phrase_generee)
```

• Pourquoi ce modèle peut-il produire des phrases peu cohérentes ?

Le modèle produit des phrases peu cohérentes car le modèle dépend du mot mot précédent, il y a aussi une absence de structure grammatical et il y a des transitions aléatoires.

• Comment améliorer la qualité de la génération avec Markov ?

Pour améliorer la qualité de la génération avec un modèle de Markov, vous pouvez augmenter l'ordre du modèle (par exemple, passer à un modèle de Markov de deuxième ou troisième ordre), enrichir le prétraitement du texte pour éliminer les mots vides et introduire des règles grammaticales simples pour garantir une meilleure structure des phrases. Enfin, l'utilisation de modèles neuronaux plus avancés (comme les RNN ou transformers) pourrait considérablement améliorer la fluidité et la cohérence des phrases générées.

🧠 Partie 4 : Génération de texte avec un LSTM

Les modèles LSTM (Long Short-Term Memory) permettent d'apprendre les structures de phrases en capturant des relations plus complexes.

📌 4.1. Préparation des données pour un réseau de neurones

Nous devons convertir le texte en une séquence de nombres pour que le modèle puisse apprendre. Exercice : Utilisez Tokenizer de Keras pour encoder le texte.

📌 4.2. Création du modèle LSTM

Nous allons construire un réseau de neurones récurrent (RNN) avec une couche LSTM.

📌 4.3. Entraînement du modèle

Nous allons exécuter l'entraînement du modèle (model.fit)

📌 4.4. Génération de texte avec le modèle LSTM

Nous allons maintenant utiliser le modèle pour générer du texte.

📝 Conclusion et réflexions

1. Comparez la qualité du texte généré avec le modèle Markov et le modèle LSTM.

2. Quels sont les avantages d'un LSTM par rapport à un modèle probabiliste ?

Le modèle LSTM garde en mémoire les dépendances longue, comprends la grammaire et elle a un output plus fluide et naturelle

3. Comment améliorer davantage la génération de texte ?

🚀 Extensions possibles

• Entraîner sur un dataset plus grand (ex. : articles Wikipédia).
• Ajouter une couche plus profonde dans le modèle LSTM.
• Tester avec des modèles transformers comme GPT.