

Cahier de charges

Factory de mots de passes

Alaa Abdelbaki
Youssef Derouiche

Table de matières

1	Introduction	2
1.1	Contexte	2
1.2	Objectifs du projet	2
1.3	Problématique	2
2	Analyse des besoins	3
2.1	Besoins fonctionnels	3
2.2	Besoins non fonctionnels	3
3	Analyse de différents modèles	4
3.1	Modèle de génération de texte	4
3.2	Modèle de génération de noms russes	4
3.2.1	Optimisation du modèle	5
3.2.2	Taux d'apprentissage	5
3.2.3	Taux de précision	6
3.2.4	Taux de pertes	7
4	Conclusion	9

1 Introduction

1.1 Contexte

Le projet **"Factory de mots de passes"** s'inscrit dans le contexte de l'apprentissage automatique pour la génération des mots de passes. Il consiste à développer une application qui implémente un réseau de neurones avec des options permettant d'entraîner le modèle sur des données d'entrée et l'évaluer en générant des nouvelles séquences à partir de quelques caractères de départ entrés par l'utilisateur.

Cette solution permet ainsi de générer automatiquement des mots de passes qui respectent les conventions d'une entreprise ou d'un domaine spécifique.

1.2 Objectifs du projet

Ce projet a les objectifs suivants à atteindre:

- Développer un modèle intelligent en utilisant la bibliothèque PyTorch [1] de Python [2].
- Développer des fonctions d'entraînement et d'évaluation pour le modèle.
- Préparer un corpus d'entraînement contenant des mots de passes respectant les conventions de sécurité.

1.3 Problématique

Ce projet vise à développer une solution pour générer des mots de passes à partir d'un corpus de données, qui sont à la fois sécurisés et facile à retenir.

La question à laquelle le projet tentera de répondre est:

Comment concevoir un modèle de génération de mots de passes performant qui respecte les normes de sécurité tout en garantissant une bonne expérience utilisateur?

2 Analyse des besoins

2.1 Besoins fonctionnels

- Les mots de passes générés doivent respecter les normes de sécurité.
- Les mots de passes générés doivent être basés sur un corpus d'entraînement.
- L'utilisateur doit pouvoir choisir une séquence de départ pour la génération des mots de passes.

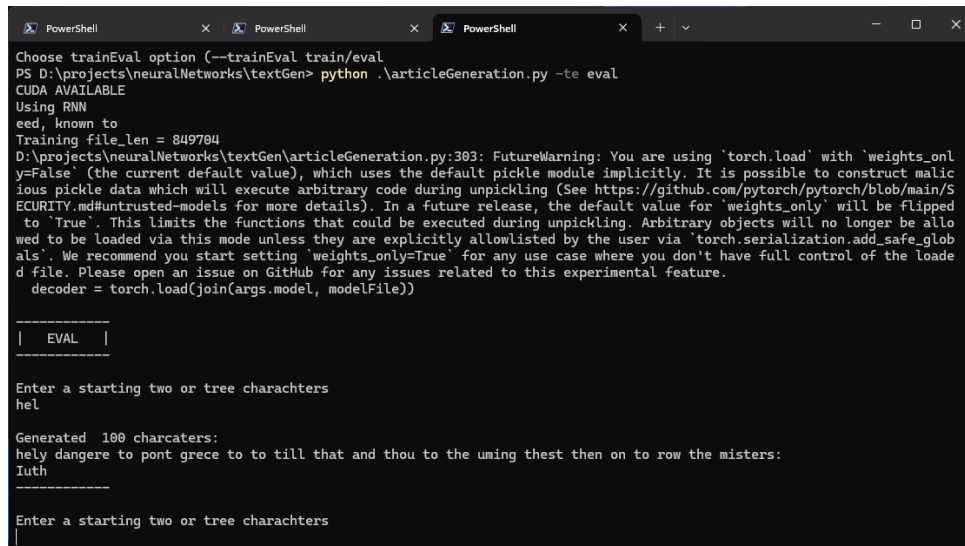
2.2 Besoins non fonctionnels

- Les mots de passes générés doivent être faciles à retenir.
- Le modèle ne doit pas prendre beaucoup de temps pour générer un mot de passe.

3 Analyse de différents modèles

3.1 Modèle de génération de texte

Le modèle de génération de texte est un modèle de langage qui génère une séquence de mots. Ce modèle est basé sur un réseau de neurones de type LSTM (Long Short Term Memory).



```
PowerShell PowerShell PowerShell
Choose trainEval option (--trainEval train/eval)
PS D:\projects\neuralNetworks\textGen> python .\articleGeneration.py -te eval
CUDA AVAILABLE
Using RNN
eed, known to
Training file_len = 849704
D:\projects\neuralNetworks\textGen\articleGeneration.py:303: FutureWarning: You are using 'torch.load' with 'weights_only=False' (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for 'weights_only' will be flipped to 'True'. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via 'torch.serialization.add_safe_globals'. We recommend you start setting 'weights_only=True' for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  decoder = torch.load(join(args.model, modelFile))

-----
| EVAL |
-----

Enter a starting two or three characters
hel

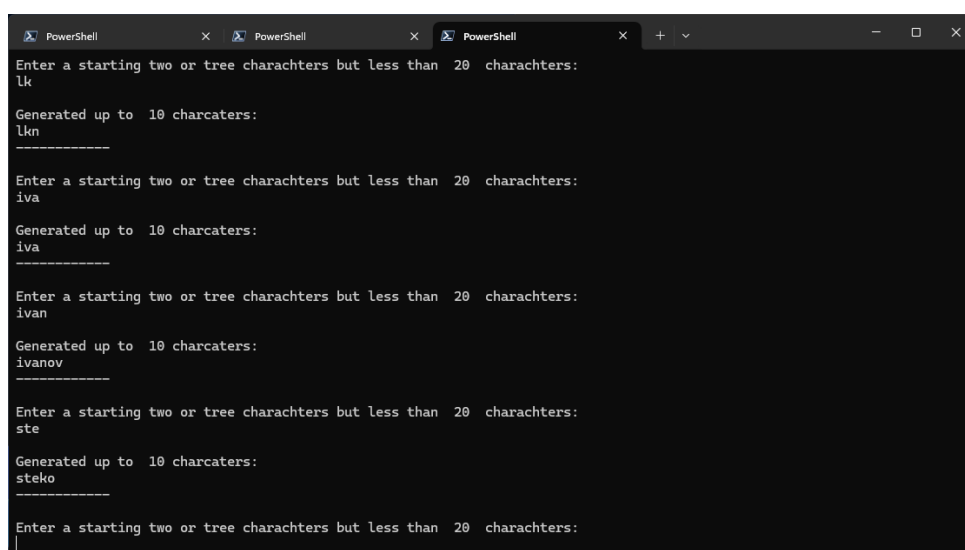
Generated 100 characters:
hely dangere to pont grece to to till that and thou to the uming thest then on to row the misters:
Iuth
-----

Enter a starting two or three characters
```

Figure 1: Modèle de génération de texte

3.2 Modèle de génération de noms russes

Le modèle de génération de noms russes est un modèle de langage qui génère une séquence de caractères pour former un nom russe. Ce modèle est basé sur un réseau de neurones récurrents (RNN).



```
PowerShell PowerShell PowerShell
Enter a starting two or three characters but less than 20 characters:
lk
Generated up to 10 characters:
lkn
-----

Enter a starting two or three characters but less than 20 characters:
iva
Generated up to 10 characters:
iva
-----

Enter a starting two or three characters but less than 20 characters:
ivan
Generated up to 10 characters:
ivanov
-----

Enter a starting two or three characters but less than 20 characters:
ste
Generated up to 10 characters:
steko
-----

Enter a starting two or three characters but less than 20 characters:
```

Figure 2: Modèle de génération de texte

3.2.1 Optimisation du modèle

Afin d'optimiser le modèle de génération de noms russes, on a étudié l'impact de utilisation des cellules GRU (Gates Recurrent Unit) [3] au lieu de RNN, et on a comparé les résultats de chaque modèle avec les hyperparamètres suivants:

- Nombre de couches: $2 \rightarrow 8$
- Nombre de couches cachées: $16 \rightarrow 256$
- Nombre d'époques: 10,000
- Taux d'apprentissage: $0.01 \rightarrow 0.001 \rightarrow 0.0001$

3.2.2 Taux d'apprentissage

Au niveau des deux modèles, on remarque que les performances sont meilleures avec un taux d'apprentissage de 0.0001, où la précision atteint une valeur de 12%.

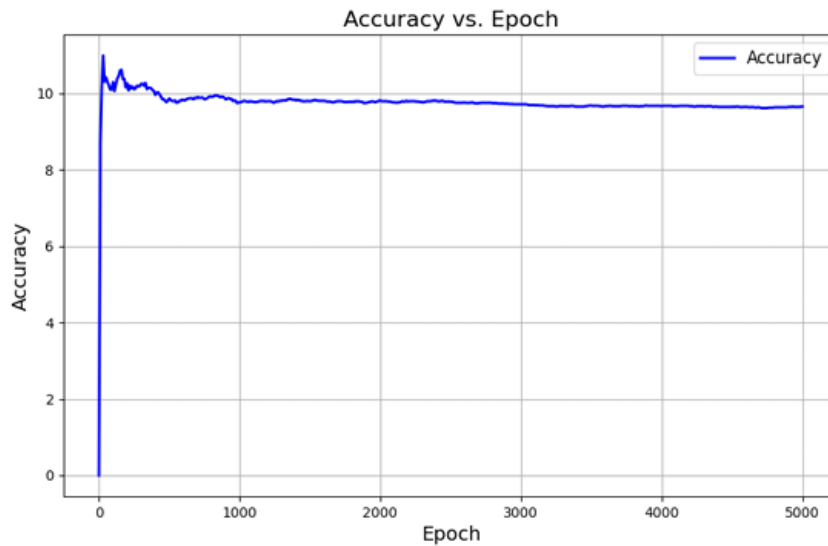


Figure 3: Précision avec un taux d'apprentissage de 0.01

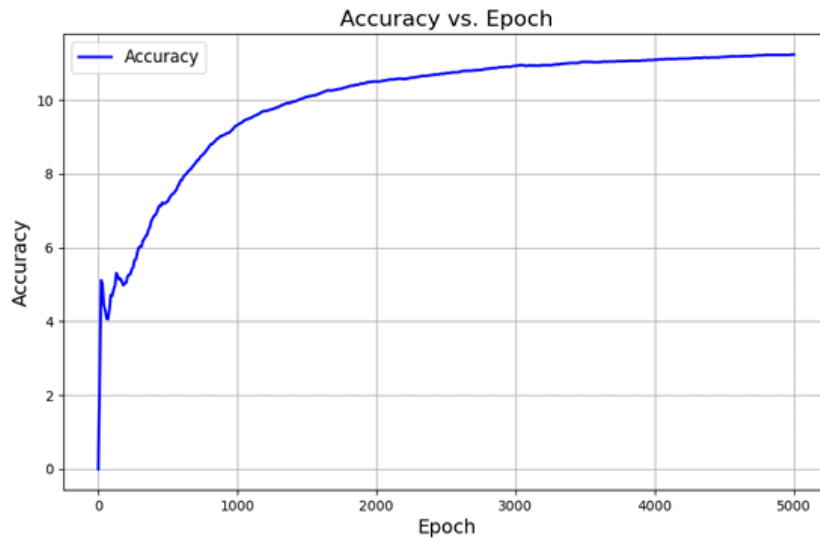


Figure 4: Précision avec un taux d'apprentissage de 0.0001

3.2.3 Taux de précision

On remarque qu'avec les différents hyperparamètres, que les performances du modèle avec GRU sont un peu mieux que celles du modèle avec RNN (2% en moyenne), mais le modèle avec RNN est plus rapide que celui avec GRU.

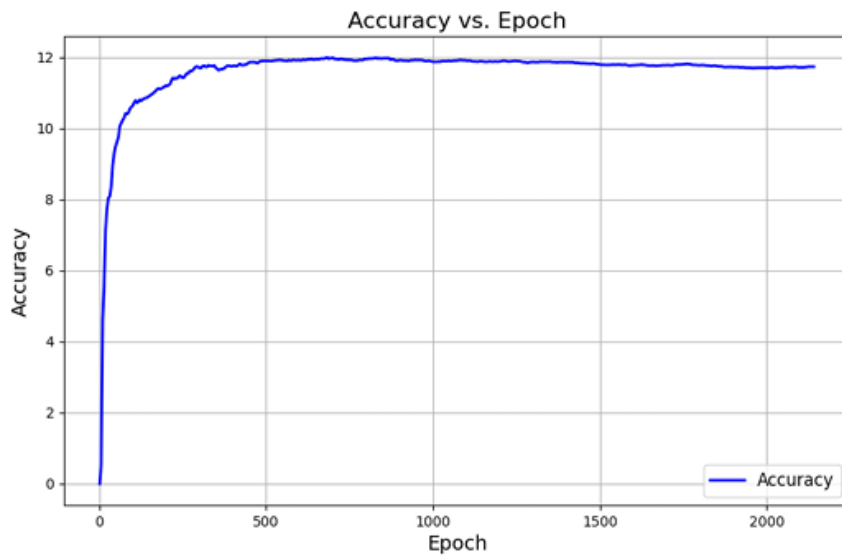


Figure 5: Précision du modèle avec GRU 12%

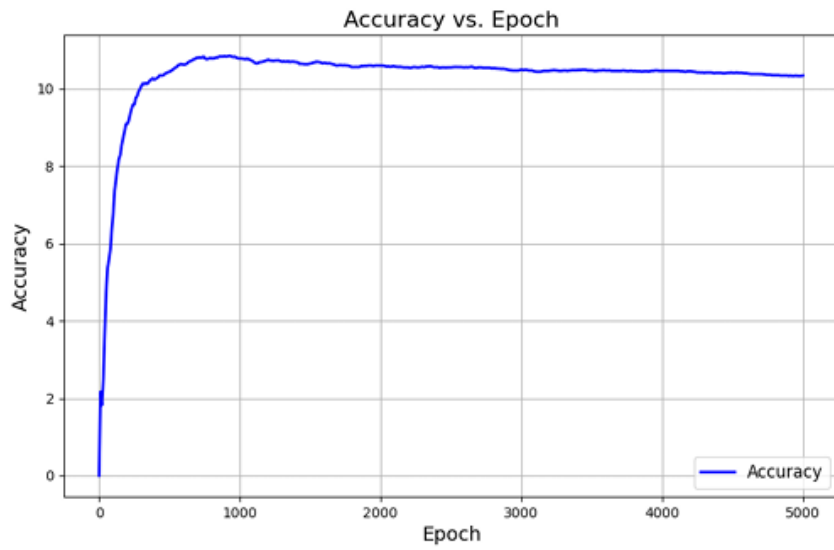


Figure 6: Précision du modèle avec RNN 10%

3.2.4 Taux de pertes

On remarque que au bout de 2,000 époques les pertes des deux modèles deviennent stable et ne changent pas beaucoup.

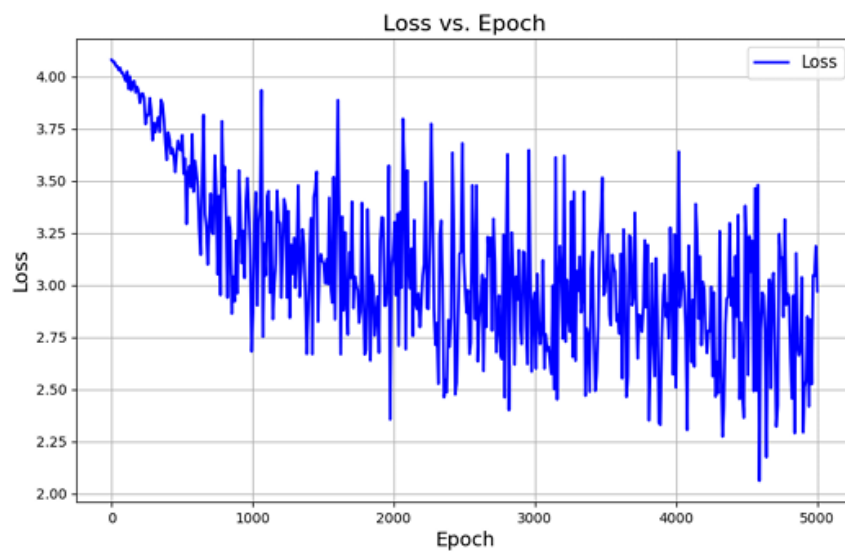


Figure 7: Pertes du modèle avec GRU

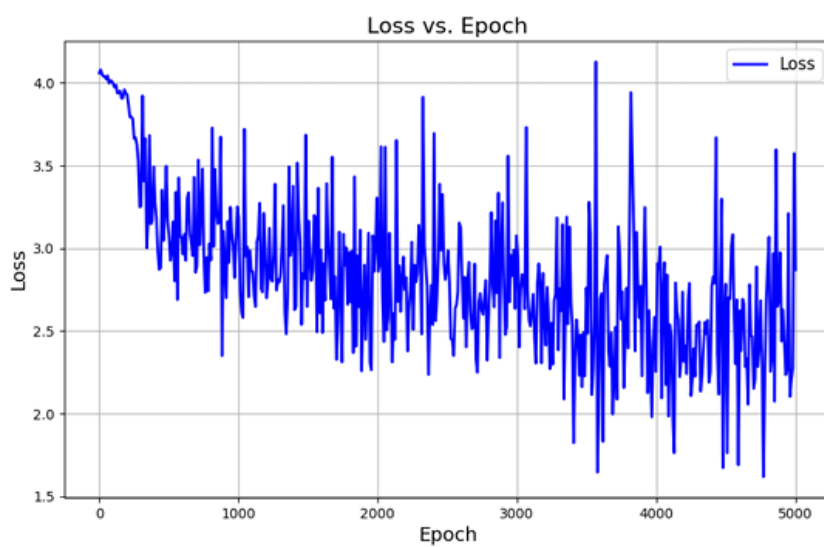


Figure 8: Pertes du modèle avec RNN

4 Conclusion

L'exploitation du code source de génération des noms russes a permis de comprendre le fonctionnement des réseaux de neurones récurrents (RNN) et des cellules GRU (Gates Recurrent Unit) pour la génération de texte.

Ces résultats permettent de mieux choisir le bon modèle pour la génération des mots de passes, et d'optimiser les hyperparamètres pour obtenir des résultats satisfaisants.

Le tableau suivant résume les choix à faire pour le modèle de génération des mots de passes:

Modèle	+++	RNN est le modèle optimal pour ce problème.
Taux d'apprentissage	+++	Une valeur autour de 0.0001 est idéale
Précision	+	Une précision de 12% (faible)
Époques	+++	Inutile de dépasser 2,000 époques
Output	++	Génération des noms russes inexistantes

Table 1: Résumé des choix pour le modèle de génération des mots de passes

References

- [1] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch. <https://pytorch.org>, 2019. Consulté le 8 novembre 2024.
- [2] Fondation Python Software. Langage de programmation python. <https://www.python.org>, 2023. Consulté le 8 novembre 2024.
- [3] PyTorch Contributors. torch.nn.gru. <https://pytorch.org/docs/stable/generated/torch.nn.GRU.html>, 2024.