



E1 - Cas pratique

Projet chef d'oeuvre : Générateur d'histoire pour enfants

Titre professionnel "Développeur/se en intelligence artificielle"
de niveau 6 enregistré au RNCP sous le n°34757

Hourdin Charlène

Passage par la voie de la formation
Parcours de 19 mois achevé le 12 février 2022

Table des matières

1 Contexte	1
1.1 Besoin exprimé pour le projet	1
1.2 Préconisation de solution	1
2 Gestion de projet	2
2.1 Pilotage du projet	2
2.2 Méthodologie agile	2
3 Analyse du besoin	3
3.1 Formulation des Personas	3
3.2 Product backlog	4
3.3 Calendrier, Sprint Review et outils mis en oeuvre	5
4 La base de données analytique	6
4.1 L'analyse exploratoire	7
5 Choix des modèles d'intelligence artificielle	10
5.1 Approche	10
5.2 Les modèles choisis	10
5.2.1 Le modèle LSTM Bidirectionnelle	10
5.2.2 Nettoyage et pré-traitement des données pour le modèle BiLSTM	11
5.2.3 L'architecture du modèle	12
5.2.4 L'entraînement du modèle	12
5.2.5 Les performances du modèle	13
5.2.6 Le modèle GPT-2	13
5.2.7 Nettoyage et pré-traitement des données pour le modèle GPT-2	14
5.2.8 L'architecture du modèle	14
5.2.9 L'entraînement du modèle	15
5.2.10 Les performances du modèle	16
5.2.11 Modèle fr-boris (Cedille.ai)	19
5.2.12 Inférence	19
6 L'application	20
6.1 La base de données	20
6.1.1 Implémentation de la base de données	21
6.2 Le développement de l'application	22
6.2.1 Le parcours utilisateur	23
6.2.2 L'architecture de l'application	24
6.2.3 Création de l'interface	25
6.2.4 Création du back-end de l'application	25
6.2.5 Gestion des sessions utilisateur	26
6.3 Les tests unitaires	27
6.4 Le monitoring	28
6.4.1 Alerting	31
7 Les axes d'améliorations	31

8 Conclusion	32
9 Annexe	33
9.1 Gestion de projet - Tableau de bord Trello	33
9.2 Analyse exploratoire	34
9.3 Modèle BiLSTM + Nettoyage	56
9.4 Modèle GPT-2	62
9.5 Modèle fr-boris (Cedille.ai)	69
9.6 Evaluation Andersen	71
9.7 Interface de l'application	73
9.8 Test Unitaire	79
9.9 Monitoring	81
9.9.1 Utilisation horaire de l'API	81
9.9.2 Utilisation de l'API multi-versions	82
9.9.3 Utilisation quotidienne de l'API	83
9.9.4 Performances des API	84
9.9.5 Rapport	85
9.9.6 Rapport	86

1 Contexte

Le présent projet a été réalisé individuellement, et ce, de septembre 2021 à janvier 2022. Il a été développé en parallèle de mon activité au sein de l'entreprise Groupama-Loire-Bretagne en qualité d'alterante.

1.1 Besoin exprimé pour le projet

La lecture peut être définie comme une activité psychosensorielle qui vise à donner un sens à des signes graphiques recueillis par la vision et qui implique à la fois des traitements perceptifs et cognitifs. Les capacités cognitives de notre corps nous aident à mieux comprendre le monde qui nous entoure, il est possible d'appliquer ce principe à l'apprentissage machine.

Le besoin exprimé pour mon projet est que de plus en plus d'enfants développent des troubles d'apprentissage en lecture et en écriture appelée le plus souvent trouble dyslexique (trouble de l'apprentissage du langage écrit).

Le plus souvent, suite à leurs troubles, ces enfants développent une phobie de la lecture. Pour les aider, je souhaite développer un outil permettant de générer des histoires en permettant à l'utilisateur de débuter une histoire.

Cet outil peut également convenir à plusieurs cas d'usage :

- Un auteur qui présente un syndrome de leucosélophobie (syndrome de la page blanche)
- Un enseignant qui souhaite créer un nouvel exercice basé sur une histoire (par exemple création d'un dessin en fonction de l'histoire lue)
- Un parent ou un enfant qui souhaite créer une histoire unique.

1.2 Préconisation de solution

La préconisation de solution présentée en réponse au besoin exprimé est le développement d'une application de génération de texte sous forme d'histoires pour enfant.

Cette solution tend à générer des histoires en anglais et en français.

Le but de cette application est de créer une histoire avec l'aide d'une intelligence artificielle, cette application permet aux utilisateurs de :

- Développer l'imagination et la créativité,
- Leurs faire aimer la lecture,
- Apprendre à utiliser un clavier et des outils numériques

2 Gestion de projet

2.1 Pilotage du projet

Le projet étant personnel, la gestion de projet a été respectivement mise en œuvre par mes soins. J'ai été amenée entre autres à :

- Définir les méthodes projet à mettre en œuvre et à appliquer,
- Recueillir et traiter les données ;
- Réaliser le benchmark et développer le modèle d'intelligence artificielle ;
- Développer la base de données et l'application .

2.2 Méthodologie agile

Pour la gestion de projet, j'ai choisi d'appliquer la méthodologie de travail Scrum.

Cette méthode fait partie des méthodes agiles inspirées du manifeste agile édité en 2001, elle a été inventée par Ken Schwaber et Jeff Sutherland.

Selon les principes de **Scrum**, le travail sur le projet est divisé en plusieurs itérations appelées « sprint ». Lors d'un sprint, l'équipe essaye de produire un incrément, autrement dit un ou plusieurs éléments de l'application. Tous les éléments de l'application sont représentés en forme de « **user story** » et se trouvent dans le « **Product backlog** ». À la fin du « sprint », le résultat est discuté et présenté au client, afin d'avoir une vision commune du projet et de savoir qui travaille sur quelle partie, l'équipe **Scrum** utilise un tableau de bord (« **kanban** »).

En général, il est divisé en quatre colonnes :

- **Product backlog** : contiens les user stories ;
- **A faire** : les tâches à réaliser ;
- **En cours** : les tâches en cours de réalisation ;
- **Fait** : les tâches terminées.

À la différence de product backlog, sprint backlog contient les éléments de l'application retenus pour le « sprint » et découpés en tâches. La colonne « en cours » répertorie les tâches en cours et « fait », les tâches finies. L'équipe partage la même définition du fini et peut déplacer une tâche dans la colonne « fait » si tout le monde est d'accord.

Les membres de l'équipe communiquent régulièrement. Au début du « sprint », ils se réunissent pour découper les éléments de l'application à produire en tâches et évaluer leurs difficultés.

Chaque jour, l'équipe participe au « daily scrum meeting », une réunion courte durant laquelle chacun raconte sa partie du travail. À la fin du sprint, les membres de l'équipe se réunissent pour la présentation du travail réalisé.

Le manifeste agile repose sur 4 grandes valeurs :

- **Les individus et les interactions plus que les processus et les outils ;**
- **Des logiciels opérationnels plus qu'une documentation exhaustive ;**
- **La collaboration avec les clients plus que la négociation contractuelle ;**
- **L'adaptation au changement plus que le suivi d'un plan.**

Les 6 pratiques suivantes sont requises pour garantir une transition efficace et une implémentation réussie.

- **Visualiser le workflow** : l'objectif est de comprendre le workflow, de suivre la progression et de localiser les goulots d'étranglement afin de faire les ajustements requis ;
- **Limiter les Travaux En Cours (TEC)** : la multiplication des tâches à tendance à générer de l'inertie et de l'inefficacité. La limite de TEC à justement pour but d'éviter ce cas de figure. Un système pull est ainsi intégré à tout ou partie du workflow. Cette limite permet de s'assurer qu'une tâche ne pourra être commencée ou déplacée à la prochaine étape que si de “l'espace” est disponible pour elle. Cette manière de procéder permet de visualiser les goulots d'étranglement et de traiter cela ;
- **Gérer les flux** : l'objectif est de fluidifier et rationaliser le workflow (flux de travail ou ensemble des opérations du processus de production). Le focus est donc sur la gestion du travail et non sur les personnes. Ce qui est recherché, c'est un flux rapide et régulier ;
- **Clarifier le processus décisionnel** : le processus doit être clairement défini, décrit et partagé. C'est la connaissance de ce dernier qui rendra possible son amélioration ;
- **Boucles de feedback** : des réunions régulières doivent avoir lieu pour partager les informations et la connaissance. Celles-ci ont vocation à être courtes (entre 10 et 15 minutes), à se faire face à un tableau kanban et chaque participant partage ce qu'il a réalisé la veille et ce qu'il projette de faire le lendemain ;
- **Améliorer la collaboration** : pour que les changements soient durables et l'amélioration continue, il est nécessaire que l'objectif et les problématiques du workflow qu'il va falloir solutionner ;

3 Analyse du besoin

3.1 Formulation des Personas

Conformément à l'expression du besoin fournie ci-dessus, trois types d'utilisateurs ont été définis :

- **L'utilisateur** : L'utilisateur final de l'application qui entre un texte dans l'endroit prévu afin de générer un début d'histoire
- **Le développeur** : La personne en charge du développement et de la maintenance du prototype
- **L'administrateur** : La personne chargée de la supervision du fonctionnement de l'application au travers d'un outil de monitoring qui recueille les alertes en cas de dysfonctionnement du système.

3.2 Product backlog

En se basant sur les objectifs de l'application et les fonctionnalités souhaitées, j'ai créé le product backlog composé de 9 user stories :

- **En tant qu'utilisateur** je dois pouvoir m'authentifier **afin d'utiliser** l'application de manière sécurisée.
- **En tant qu'utilisateur** je dois pouvoir accéder à la page qui me permet de générer du texte **afin de** créer mon histoire.
- **En tant qu'utilisateur** je dois pouvoir visualiser le résultat du modèle **afin de** pouvoir le lire.
- **En tant qu'utilisateur** je dois pouvoir accéder aux histoires enregistrées **afin de** pouvoir les exploiter ou les supprimé.
- **En tant qu'utilisateur** je dois pouvoir m'enregistrer sur le site **afin de** pouvoir accéder à mon compte.
- **En tant qu'administrateur** je dois pouvoir visualiser l'état de fonctionnement de l'application **afin de** m'assurer de son bon fonctionnement.
- **En tant qu'administrateur** je dois pouvoir être alerté en cas de dysfonctionnement de l'application **afin d'y** remédier.
- **En tant que data scientist** je dois pouvoir ré-entraîné facilement le modèle **afin d'optimiser** ses performances.
- **En tant que data scientist** je dois pouvoir déployer l'application **afin de** la rendre disponible aux utilisateurs.

3.3 Calendrier, Sprint Review et outils mis en oeuvre

Le projet a été découpé en 5 sprints de 4 semaines chacun, le projet à duré 20 semaines. J'ai pu suivre l'avancement des travaux à l'aide d'un tableau de bord Trello contenant les différentes tâches lié à mon projet en lien avec les compétences du référentiel pour le passage du Titre de développeur IA.

Le tableau Trello vous est présenté en annexe 9.1 page 33.

Lors de la Sprint Review, l'ensemble de l'équipe Scrum est invité ainsi que le client si cela est possible, si le client est absent le rôle du product owner est de lui rendre compte de l'état d'avancement du projet en cours. Elle a lieu à chaque fin de sprint

Voici par exemple les étapes possibles lors d'une Sprint Review :

- Présenter les items terminés,
- État d'avancement du produit (MVP, release...),
- Démonstration de ce qui est présentable (les user-stories),
- Demande de feedback,
- Expliquer la vision pour la suite

Voici le découpage de mes sprints et leurs contenus :

- **Sprint 1** : Obtention, analyse et nettoyage des données ;
- **Sprint 2** : Implémentations des modèles Bi-LSTM et GPT-2 ;
- **Sprint 3** : Création de la base de données et de l'application ;
- **Sprint 4** : Réalisation des tests unitaire et du monitoring ;
- **Sprint 5** : *En cours* : Amélioration de la base de données, de l'application.

Durant chaque sprint, un graphique burn down chart est réalisé, c'est une représentation du travail restant à faire par rapport au temps. Le travail restant (ou arriéré) est souvent sur l'axe vertical, avec le temps sur l'horizontale.

Il est utile pour voir l'avancement du sprint en cours et prévoir quand tous les travaux seront terminés.

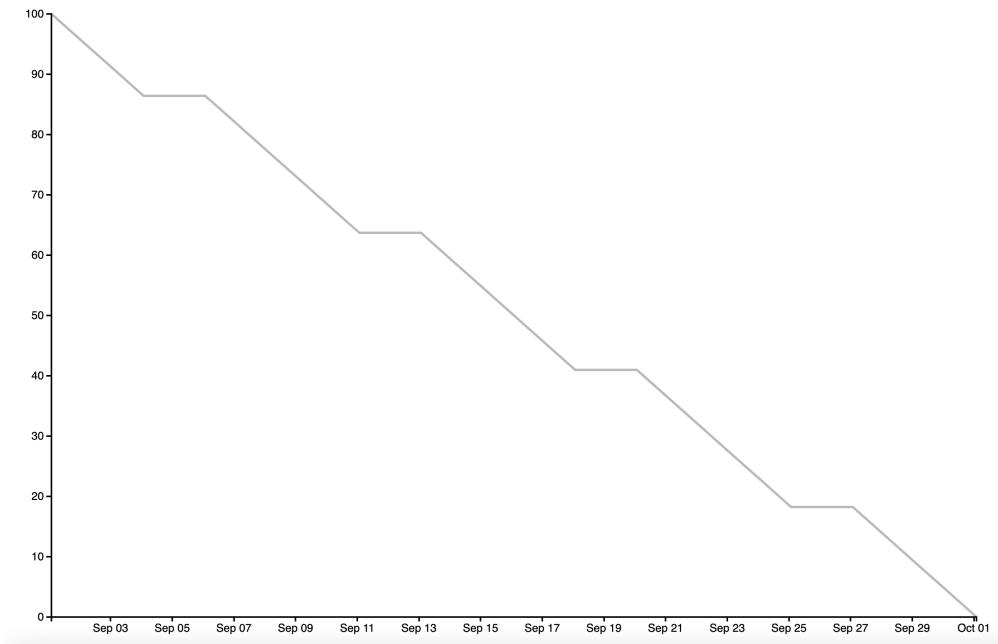


FIGURE 1 – burn down chart du sprint 1

4 La base de données analytique

L'objectif du projet étant de pouvoir générer des histoires, il est nécessaire de disposer d'un volume important d'histoires.

J'ai donc initiée une recherche de sources de données répondant aux critères suivants :

- **Données disponible** faisant partie du domaine public (les données venant d'un auteur font partie du domaine public 50 ans après sa mort.)
- **Données appropriées** aux enfants (sans contenus injurieux, etc.)

Pour le projet, j'ai besoin de données en français et en anglais. Les données utilisées pour ce projet peuvent être la propriété de créateur de contenu, dans mon cas des auteurs, j'ai décidé d'utiliser un ensemble de données accessible au public contenant différentes histoires. J'ai récupéré l'ensemble des données en effectuant du web scraping via une API pour les datasets des contes de Grimms en Français et j'ai récupéré les datasets des contes d'Andersen en français et en anglais ainsi que les contes de Grimms en anglais sur le projet Github d'un utilisateur ayant utilisé ses ensembles de données pour un projet traitant du NLP (Natural language preprocessing)

L'ensemble des données utilisés dans ce projet sont :

- **Les contes de Grimms :**

- 115 histoires en français ;
- 201 histoires en anglais.

— **Les contes d'Andersen :**

- 53 histoires en français ;
- 156 histoires en anglais.

Ces données faisant partie du domaine public, je me prémunis contre les conséquences juridiques potentielles relatives à un droit d'auteur en cours ou à un recours intensif au web scraping dont la légalité est discutable.

4.1 L'analyse exploratoire

L'analyse exploratoire des données (EDA) est une des étapes essentielles dans toute analyse de données. Le but de cette analyse est d'explorer et d'analyser un jeu de données via l'utilisation de graphiques et de visuels. Les visualisations ci-dessous ont pour objectif de présenter les principales caractéristiques des datasets utilisé dans le cadre de ce projet.

Voici les statistiques descriptives des datasets des contes de Grimms et d'Andersen en français :

	nb_caractere	nb_mots		nb_caractere	nb_mots
count	115.000000	115.000000		count	53.000000
mean	7525.339130	1347.669565		mean	14223.943396
std	5077.688742	913.776267		std	11167.577593
min	803.000000	136.000000		min	2020.000000
25%	3329.500000	593.500000		25%	6871.000000
50%	6799.000000	1230.000000		50%	10338.000000
75%	10635.500000	1953.500000		75%	18734.000000
max	21650.000000	3879.000000		max	63286.000000
					11202.000000

Ces tableaux nous donnent les descriptions suivantes :

- le nombre d'histoires ;
- la moyennes des mots et des caractères ;
- l'écart type ;
- la valeur maximum du nombre de mots et de caractères ;
- la valeur minimum du nombre de mots et de caractères.

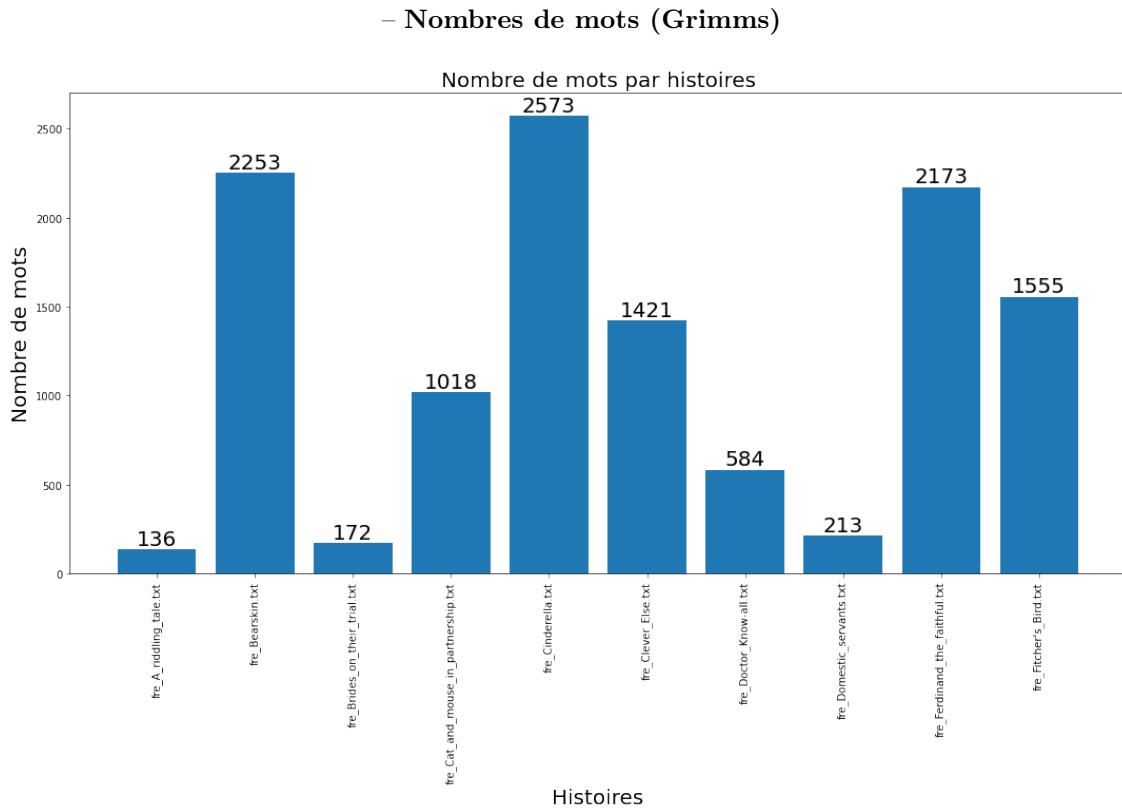
Les images ci-dessous présentent respectivement le nombre de mots par histoire, le nombre de caractères par histoire et la fréquence des mots par histoires. Ayant beaucoup d'histoire

dans les jeux de données, je vous présente ci-dessous un échantillon des 10 premières histoires pour chaque dataset des contes de Grimms et d'Andersen en français.

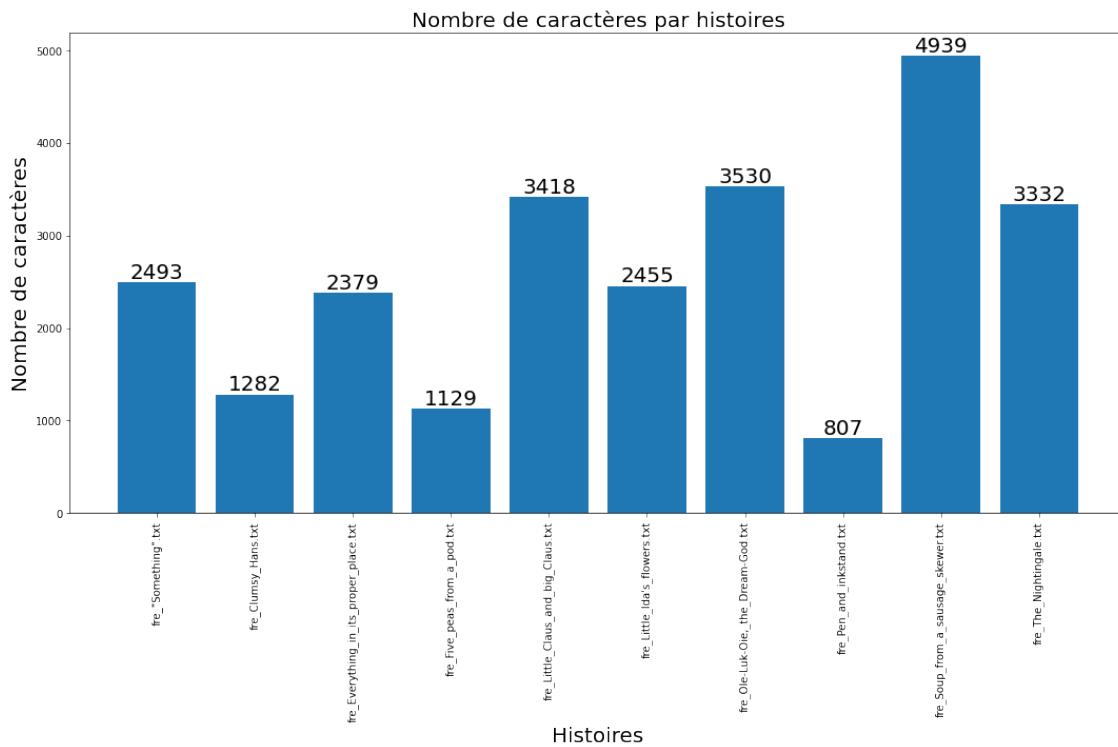
L'analyse exploratoire de l'ensemble des datasets vous est présenté en annexe 9.2 page 34.



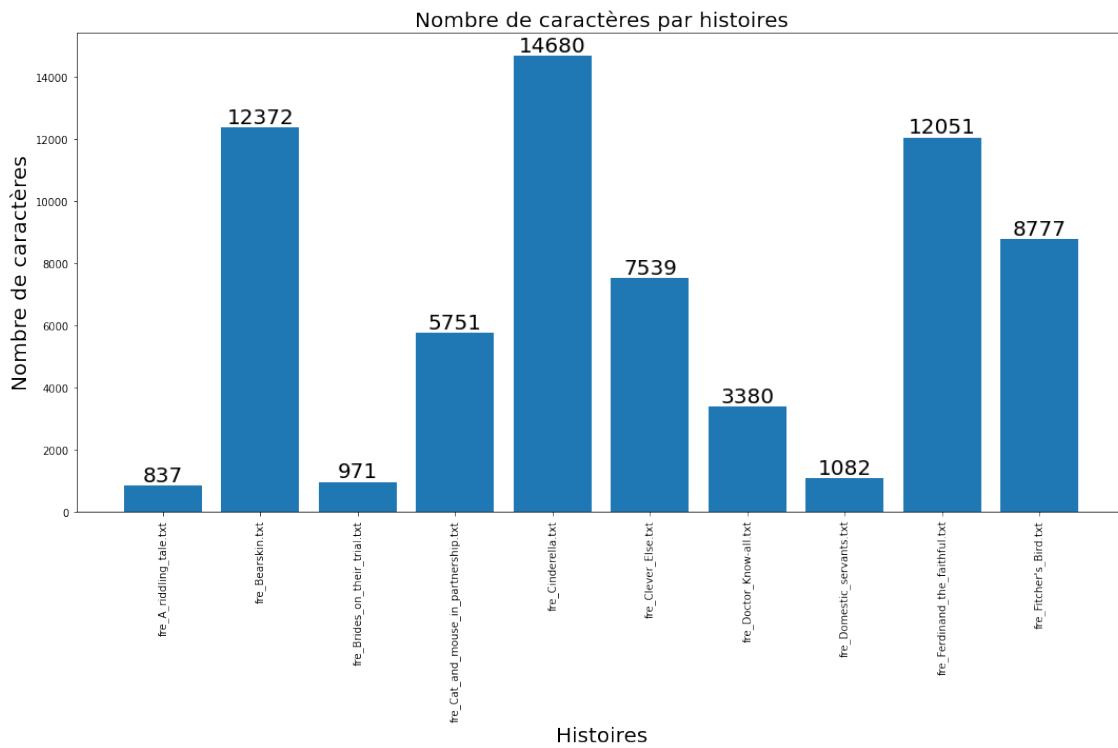
FIGURE 2 – Nuage de mots d'une histoire du dataset des contes de Grimms et des contes d'Andersen en français



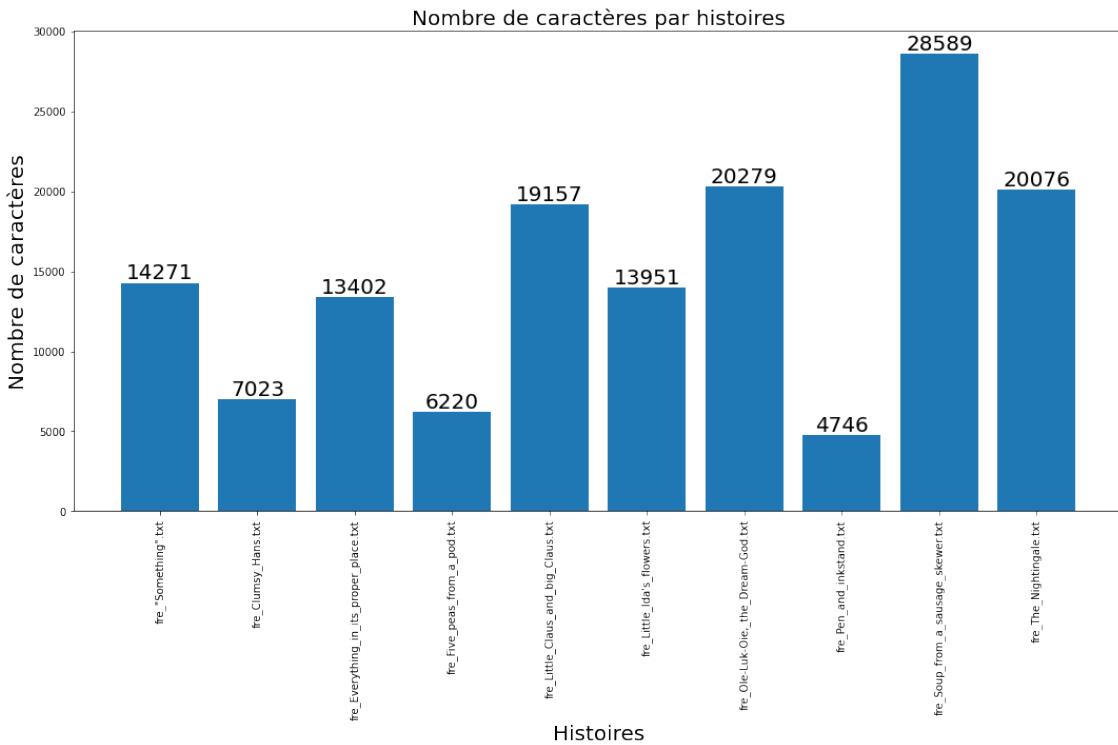
– Nombres de mots (Andersen)



– Nombres de caractères (Grimms)



– Nombres de caractères (Andersen)



5 Choix des modèles d'intelligence artificielle

5.1 Approche

J'ai choisi d'implémenter un modèle LSTM bidirectionnelle seulement sur les corpus de texte en français, et d'effectuer ensuite du transfert learning à l'aide des modèles Transformers. Mon choix, c'est arrêté sur le modèle 'GPT-2' (Open AI) que j'ai implémenté sur mon corpus en français et en anglais. Ce modèle fait partie de la classe des Transformers. (*pour plus d'informations voir le rapport effectué sur la veille*)

5.2 Les modèles choisis

5.2.1 Le modèle LSTM Bidirectionnelle

Afin de modéliser des dépendances à très long terme, il est nécessaire que les réseaux de neurones récurrents maintiennent leurs états sur de longues périodes de temps.

C'est à cela que sert une cellule LSTM (Long Short Term Memory), elle possède une mémoire interne appelée cellule (ou unité). L'unité peut maintenir un état aussi longtemps que nécessaire. L'unité est constituée d'une valeur numérique que le réseau peut contrôler en fonction de la situation.

5.2.2 Nettoyage et pré-traitement des données pour le modèle BiLSTM

À noter : le modèle BiLSTM a été implémenté seulement sur les corpus de texte en français.

La première étape que j'ai effectuée pour le nettoyage des données a été la suppression des caractères spéciaux, doubles espaces, motifs dans le texte tout ce qui n'est pas un mot ou une langue parlée. Je garde les majuscules et la ponctuation, car lors de la génération du texte, je souhaite que le modèle génère un texte avec les majuscules et la ponctuation afin que l'histoire ait du sens.

'Trois femmes avaient été métamorphosées en fleurs et brillaient ainsi dans la campagne. Cependant le charme permettait que l'une d'elles retournât chaque nuit dans sa demeure. Il y avait quelque temps qu'elle subissait cette métamorphose, lorsqu'elle dit à son mari: "L'aurore va paraître, et je devrai te quitter de nouveau pour rejoindre mes compagnes et redevenir, comme elles, fleur des champs; mais si tu arrives aujourd'hui avant midi, et que tu me cueilles, l'enchantement cessera, et je ne le'

FIGURE 3 – Échantillons de texte après avoir effectué le nettoyage

À l'étape suivante, j'ai effectué une tokenisation, étant donné que les méthodes d'apprentissage automatique ne peuvent pas optimiser directement le texte, j'ai besoin d'un moyen de convertir le texte en une représentation numérique. Pour cette tâche, j'utilise des incorporations de mots, qui sont des moyens par lesquels un mot peut être représenté comme un vecteur de nombres.

J'ai donc créer un corpus en fractionnant les données au niveau du caractère '\n', puis en jetons, le nombre de mots est également déterminé, j'ai donc un total de 11518 mots dans le corpus des contes de Grimms, et 12063 pour le corpus des contes d'Andersen.

Le vocabulaire est de 11518 mots et l'index de mots mappe ce vocabulaire à leur représentation numérique ce qui est essentiel pour coder les séquences.

Je dois ensuite créer les données d'entraînement en créant les séquences de n mots comme des vecteurs d'entrée (X) et les mots suivants comme des étiquettes (y).

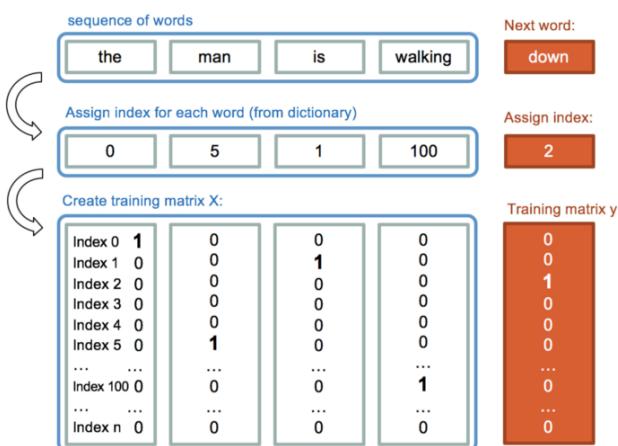


FIGURE 4 – Création de séquences

5.2.3 L'architecture du modèle

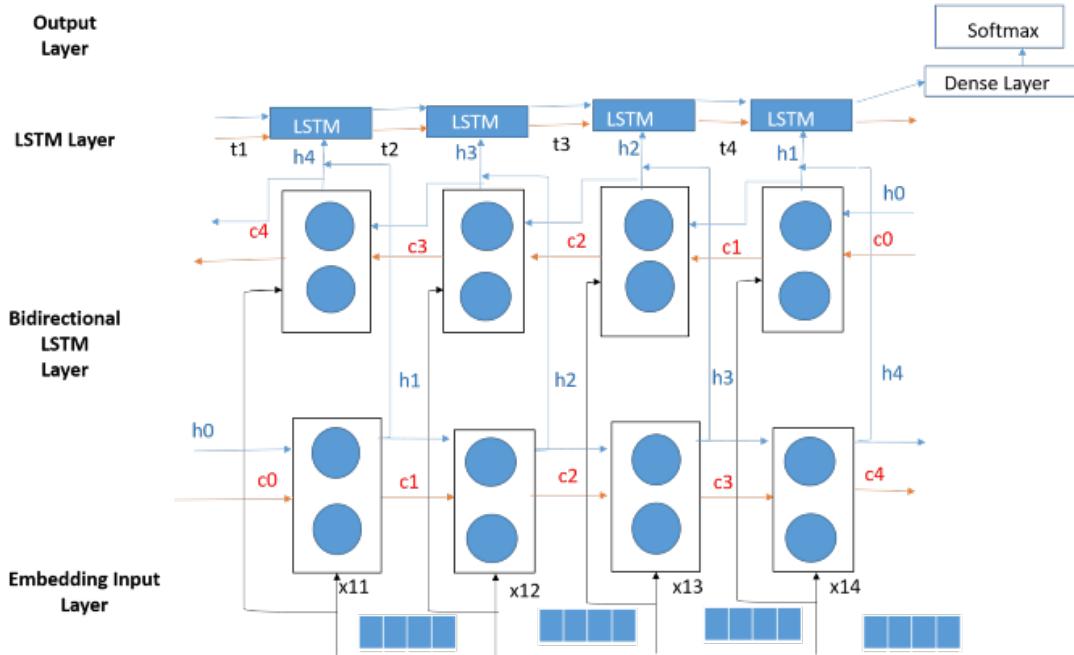


FIGURE 5 – Architecture d'un modèle LSTM Bidirectionnelle

Comme on peut le voir dans l'architecture ci-dessus Les LSTM bidirectionnels peuvent être utilisés pour former deux côtés, au lieu d'un côté de la séquence d'entrée. Le premier de gauche à droite sur la séquence d'entrée et le second dans l'ordre inverse de la séquence d'entrée.

Il fournit un contexte supplémentaire au mot pour s'adapter au bon contexte à partir des mots venant après et avant, cela se traduit par un apprentissage et une résolution plus rapides et complète d'un problème, c'est pour cela que les modèles BiLSTM montrent de très bons résultats.

5.2.4 L'entraînement du modèle

J'ai implémenté un modèle séquentiel, la première couche est une couche d'incorporation de mots (*word-embedding*). Cette couche est essentielle pour la génération de texte, car elle fournit une meilleure compréhension des mots en donnant des valeurs mathématiques similaires pour les mots ayant des significations similaires. Cela aide à générer un texte plus logique.

Ensuite, il y a une pile de couches LSTM bidirectionnelle, elle permet à la fois des informations en avant et en arrière sur la séquence à chaque étape de temps en préservant mieux le contexte.

Une couche de décrochage est ajoutée pour éviter le sur-ajustement, une couche supplémentaire de LSTM bidirectionnelle et une couche plus dense avec activation comme Relu, et un régularisateur pour éviter le sur-ajustement.

Une couche softmax pour la couche de sortie, elle sert à obtenir la probabilité que le mot soit prédit ensuite.

5.2.5 Les performances du modèle

Pour évaluer les performances du modèle, je regarde la métrique d'Accuracy sur le jeu de train et le jeu de test, j'ai choisi volontairement d'arrêter le modèle après avoir effectué 10 epochs, car le modèle commençait à sur-entraîner cela peut être lié au fait que je n'ai pas assez de données pour l'entraînement. En effet, pour qu'un modèle génère un texte de qualité avec une bonne performance, il faut envoyer une quantité non-négligeable de données au modèle par exemple le modèle GPT-2 a été entraîné sur un corpus de texte mesurant 40Go, hors le mien ne mesure que 887,167 octets soit 1,1 Mo pour les contes de grimms et 771 651 octets soit 893 Ko pour les contes d'Andersen en français.

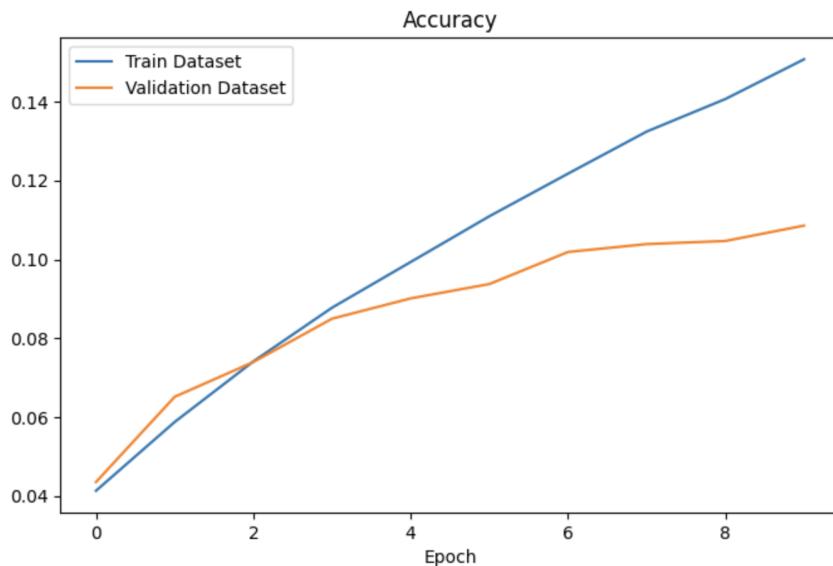


FIGURE 6 – Performance du modèle

5.2.6 Le modèle GPT-2

Le modèle GPT-2 existe en 4 variantes :

J'ai choisi d'entraîner le plus petit modèle, il contient 124M de paramètres, j'ai choisi celui-ci, je n'avais pas les ressources nécessaires pour entraîner les plus grosses variantes, en effet, ces modèles nécessitent d'avoir des ressources en GPU assez importantes pour pouvoir les entraîner. Ces modèles demandent également des ressources de stockage assez volumineuses. La plus petite variante du GPT-2 occupe 500MBs de stockage pour stocker tous ses paramètres. La plus grande variante de GPT-2 est 13 fois plus grande, prenant ainsi plus de 6,5 Go d'espace de stockage.

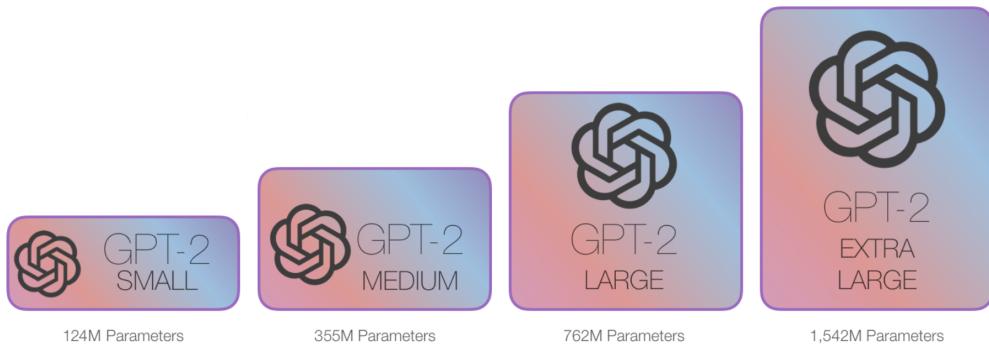


FIGURE 7 – Variantes GPT-2

5.2.7 Nettoyage et pré-traitement des données pour le modèle GPT-2

J'ai effectué le même nettoyage de données pour le modèle GPT-2 et le modèle BiLSTM c'est à dire suppression des caractères spéciaux, doubles espaces, motifs dans le texte tout ce qui n'est pas un mot ou une langue parlée. Je garde les majuscules et la ponctuation, car lors de la génération du texte, je souhaite que le modèle génère un texte avec les majuscules et la ponctuation afin que l'histoire ait du sens. Pour le pré-traitement, ayant effectuer un réglage fin, je n'ai pas eu à effectué de pré-traitement (tokenisation, création des séquences, etc) car cela est fait automatiquement avec la fonction `finetune()`.

5.2.8 L'architecture du modèle

Comme je l'explique dans la veille que j'ai effectuée sur les Transformers, le Transformer basique est composé d'un encodeur et d'un décodeur.

À la différence des Transformers basique, GPT-2 lui est composé seulement de bloc décodeur (la version small que j'utilise est composée de 12 blocs de décodeurs).

La fonction principale du bloc de décodeur est de comprendre la relation entre chaque mot, ce qui permet d'identifier les liens qui relient les mots entre eux. Cela permet au modèle de comprendre quel verbe est associé à quel sujet, quel sujet est associé à quel adjectif, etc. Le GPT-2 peut traiter 1024 jetons. Chaque jeton parcourt tous les blocs décodeurs le long de son propre chemin.

C'est un modèle auto régressif, c'est-à-dire qu'il produit un seul token à la fois, après chaque token produit, le token est ajouter à la séquence d'entrée et cette nouvelle séquence devient l'entrée du modèle pour l'étape suivante.

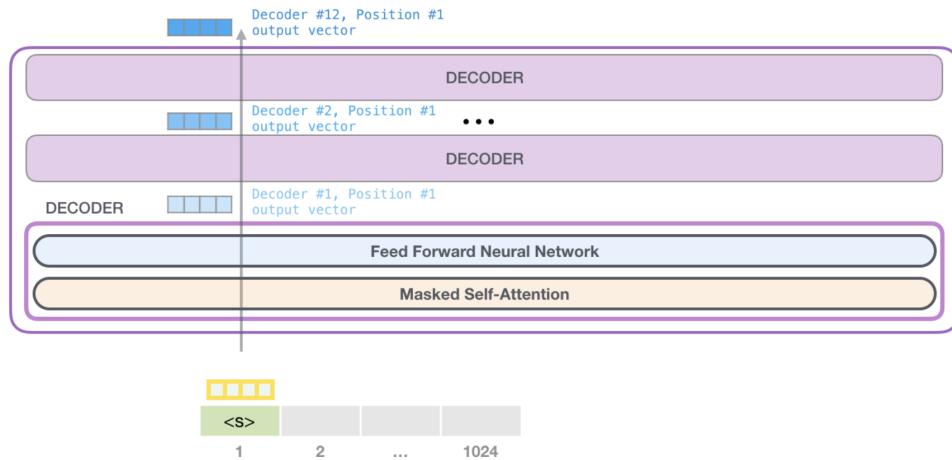


FIGURE 8 – Architecture de GPT-2

Dans l'exemple ci-dessus, le premier bloc traite le premier jeton en le faisant d'abord passer par le processus d'auto-attention, puis par sa couche d'anticipation (feed-forward). Une fois le traitement terminé, le bloc envoie le vecteur résultant pour que le traitement soit effectué par le bloc suivant. Le processus dans chaque bloc est le même, mais chaque bloc a ses propres poids dans l'auto-attention et les sous-couches du réseau neuronal.

GPT-2 utilise une couche d'auto-attention masquée, cette couche masque les tokens futur en bloquant les informations des tokens qui sont à droite de la position à calculer.

L'avantage de la couche d'auto-attention est qu'elle permet à la machine de comprendre les relations entre des mots très éloignés dans une phrase, elle peut donc détecter le contexte des mots afin d'avoir une compréhension globale de la phrase avant de le traiter en le faisant passer par un réseau de neurones.

5.2.9 L'entraînement du modèle

À des fins de réglage fin, puisque je ne peux pas modifier l'architecture, je considère GPT-2 comme une boîte noire, prenant des entrées et fournissant des sorties. L'entrée est une séquence de jetons et la sortie est la probabilité du jeton suivant dans la séquence, et ces probabilités sont utilisées comme poids pour que l'IA sélectionne le jeton suivant dans la séquence.

Le processus général que j'ai appliqué est :

- **Effectuer le réglage fin** : dans cette cellule, je charge l'ensemble de données spécifié et l'entraîne sur un nombre d'étapes que je spécifie (j'ai commencé par 1000 étapes, mais cela n'était pas suffisant pour permettre l'émergence d'un texte distinct, j'ai donc augmenté le nombre d'étapes à 3000). Durant l'entraînement, je regarde la perte d'entraînement moyenne, la valeur absolue de la perte n'a pas vraiment d'importance, car la qualité de sortie du texte est subjective, mais si la perte cesse de baisser, cela veut dire que le modèle a convergé et une formation supplémentaire peut ne pas aider à améliorer le modèle.

Dans cette cellule, je définis également le dossier dans lequel sera enregistrer le checkpoint, et après combien d'étapes, il enregistre le point de contrôle (j'ai choisi un enregistrement toutes les 500 étapes).

- **Chargement du point de contrôle pour générer du texte :** J'ai besoin de charger le point de contrôle enregistré durant les différentes étapes du modèle afin d'utiliser les poids de mon modèle pour effectuer la génération de texte. (Le checkpoint permet l'enregistrement des poids du modèle, cela permet de ré-entraîné le modèle en partant des poids précédemment enregistrés.). Dans mon cas, j'utilise le checkpoint pour effectuer la génération sans avoir à ré-entraîné le modèle.
- **Génération de texte :** Dans cette cellule, j'utilise la fonction `gpt-2.generate()`, qui me permet de générer du texte à partir de mon modèle précédemment entraîné.

Je définis différents paramètres dans cette fonction :

- **Le point de contrôle**
- **Length** (Nombre de jetons à générer (par défaut 1023, le maximum))
- **Le paramètre Top_k** (Limite les suppositions générées aux k suppositions les plus élevées (0 par défaut qui désactive le comportement ; si la sortie générée est super folle, définir top_k=40)
- **Le paramètre Top_p** (Définition d'un seuil qui filtre la valeur de probabilité du prochain mot, bon résultat avec top_p = 0.9
 - bas = garde les mots les plus probables et évite les mots improbables
 - haut = plus créatif)
- **Le paramètre Température** (Uniformise la probabilité du prochain mot, bon résultat avec une température entre 0,7 et 1,0
 - bas = déterministe,
 - haut = plus aléatoire)
- **Le préfixe** Permet de définir le début d'un texte (Le modèle générera un texte en fonction du texte défini dans ce paramètre).

5.2.10 Les performances du modèle

Ayant effectué un réglage fin pour entraîner le modèle, je n'ai pas de métrique de performances, ce modèle étant un générateur de texte, la meilleure métrique d'évaluation, reste l'évaluation humaine.

Dans mon cas, le modèle que j'ai implémenté est destiné à des enfants, j'ai demandé à un groupe d'enfants d'évalué mon modèle, ce groupe était assez diversifié, il était constitué de 5 enfants, Hugo 5 ans, Chaden 6 ans, Mathilde 8 ans, Quentin 9 ans et Théo 12 ans.

Cette diversité m'a permis d'évaluer la qualité du texte en fonction de la compréhension de

lecture de chacun.

L'évaluation ci-dessous porte sur les modèles qui ont été entraînés sur le jeu de données des contes de Grimms en anglais et en français. J'ai demandé aux enfants d'inventer un début d'histoire que nous avons ensuite envoyé au modèle afin qu'il génère la suite. Nous avons généré des histoires en faisant varier la langue, les paramètres et la longueur. Ensuite, j'ai fait lire les histoires générées à chaque enfant afin qu'il évalue le texte en prenant en compte :

- Le contexte de l'histoire dans sa globalité
- la qualité des phrases générées (présence de ponctuation, orthographe, dialogue, ...)

La diversité de l'âge des enfants et leur compréhension de lectures et de langues étant différente, les notes l'ont été également. Voici les histoires générées avec leurs paramètres et les notes qui vont de 1 à 10 attribué par chaque enfant :

Pour la première histoire en français, nous avons défini les paramètres suivants :

- **Top_p** = 0.9
- **Température** = 1.0
- **Longueur de texte** = 50
- **Préfixe** = Il était une fois un prince qui vivait dans son château dans un pays lointain. Un jour, il rencontra une princesse. Elle était si belle qu'il en tomba amoureux. Elle avait de longs et beaux cheveux en or. Mais une méchante femme était très jalouse de la princesse et décida de l'enlever au prince.

Voici l'histoire générée :

Il était une fois un prince qui vivait dans son château dans un pays lointain. Un jour, il rencontra une princesse. Elle était si belle qu'il en tomba amoureux. Elle avait de longs et beaux cheveux en or. Mais une méchante femme était très jalouse de la princesse et décida de l'enlever au prince. «Mère, tu n'iras bien» dit sa peine ; et c'est ma mère que le poêle qui versait toujours beaucoup tropôt d'où elle est

Voici les notes des enfants et leurs appréciations :

- Hugo 5 ans = 2/10 "Ça ne veut rien dire, mais il a quand même écrit quelque chose."
- Chaden 6 ans = 1/10 "L'histoire n'a pas de sens."
- Mathilde 8 ans = 1/10 "Il a inventé des mots. (tropôt)"
- Quentin 10 ans = 2/10 "Il y a du dialogue."
- Théo 12 ans = 2/10 "Ne prends pas en compte le contexte de l'histoire qu'on a écrite."

J'ai additionné toutes les notes et effectuer la moyenne : $8/5 = 1.6$

Mon modèle obtient une moyenne de **1,6 sur 10**

Nous avons généré une seconde histoire en anglais en changeant les paramètres, nous avons

gardé le même début d'histoire que nous avons traduit en anglais :

(Les enfants ne parlant pas anglais, nous avons demandé à un adulte bilingue, de nous traduire le texte.)

- **Top_p** = 1.0
- **Température** = 0.7
- **Longueur de texte** = 100
- **Préfixe** = Once upon a time there was a prince who lived in his castle in a distant land. One day he met a princess. She was so beautiful that he fell in love with her. She had long, beautiful golden hair. But an evil woman was very jealous of the princess and decided to take her away from the prince.

Once upon a time there was a prince who lived in his castle in a distant land. One day he met a princess. She was so beautiful that he fell in love with her. She had long, beautiful golden hair. But an evil woman was very jealous of the princess and decided to take her away from the prince. Singing clean is not for me. " "Oh, woe's me! Then I will sing also for thee, Here in the sanggarden's courtyard, Sings clean, here in the hall's lock. " So the father and daughter set out, and the next morning they sang their song, "Fiddle for me, cock for thee, Here in the sanggarden's courtyard, Sings clean, here in the hall's lock. " But

Voici les notes des enfants et leurs appréciations pour la seconde histoire :

- Hugo 5 ans = 5/10 "C'est mieux."
- Chaden 6 ans = 4/10 "L'histoire est mieux."
- Mathilde 8 ans = 6/10 "L'histoire a plus de sens, mais il invente encore des mots."
- Quentin 10 ans = 7/10 "Il a bien écrit la ponctuation et l'orthographe est pas mal l'histoire à plus de sens."
- Théo 12 ans = 5/10 "Les phrases ont plus de contextes."

J'ai additionné toutes les notes effectuées la moyenne : $27/5 = 5,4$

Mon modèle obtient une moyenne de **5,4 sur 10**.

Conclusion, le modèle n'est pas performant, mais ce qui ressort des évaluations sont que le modèle génère de la ponctuation, que l'orthographe est correct et que les phrases générées ont du contexte, mais que ce contexte n'est pas lié au début de l'histoire écrite par les enfants.

Conclusion de l'histoire en français :

Comme pour le modèle BiLSTM, le fait de n'avoir pas beaucoup de données en entrée, et que le modèle GPT-2 ai été entraîné sur un corpus en anglais, que je ré-entraîne sur un petit corpus en français fait que le modèle n'est pas performant. Pour remédier à cela, il y a plusieurs possibilité :

- envoyer au modèle un très gros corpus de données en français
- utiliser un modèle comme fr-boris de cedille.ai qui a été entraîné sur un corpus en français.

Conclusion de l'histoire en anglais :

Le modèle est plus performant sur la génération de texte en anglais, cela est dû au fait que le modèle a déjà été entraîné sur un corpus de texte en anglais.

Les évaluations pour les modèles des contes d'Andersen en français et en anglais, vous sont présentées en annexe 9.6 page 71.

5.2.11 Modèle fr-boris (Cedille.ai)

5.2.12 Inférence

Le modèle fr-boris étant un modèle très volumineux (22Go) il faut des ressources en TPU assez élevé. Ce modèle est basé sur l'architecture GPT-J et entraîné à l'aide de la base de code mesh-transformer-jax. Il a été formé sur environ 78 milliards de jetons de texte français à partir du jeu de données C4.

Ne pouvant pas effectuer l'entraînement dû au problème de ressources, j'ai choisi de faire le l'inférence en l'exécutant via la base de code kingoflolz/mesh-transformer-jax qui se trouve sur le repository Github de GPT-J.

Je ne ré-entraîne donc pas le modèle sur mes données, mais utilise les poids du modèles fr-boris pour faire de l'inférence.(L'inférence consiste ici à rendre le modèle entraîné autonome et à lui permettre d'identifier, de traiter et d'analyser seul des résultats à partir d'exemples possédant un grand nombre de caractéristiques.).

Pour ce faire, j'ai cloné le repository Github de Cedille afin de récupérer le point de contrôle (check-point) et j'ai téléchargé le tokenizer fr-boris via la bibliothèque Huggingface.

Ensuite, il faut créer le réseau et charger les paramètres (ce sont les paramètres de GPT-J) en effet, fr-boris étant basé sur le modèle GPT-J, il contient exactement les mêmes paramètres.

J'effectue ensuite l'inférence et génère du texte en ajoutant à la fonction les paramètres suivant :

- **Le contexte** ;
- **Le paramètre Top-p** ;
- **La température** ;
- **La longueur du texte**.

Ci-dessous, je vous présente les résultats de deux générations de texte, l'une a été faite en fonction du contexte définit dans la fonction.

Pour la première, j'ai demandé au modèle de me générer une histoire de super héros en saisissant "**Voici une histoire de super-héros**" en définissant les paramètres suivant :

- **Top-p = 1**
- **Température = 0.7**

- La longueur du texte = 100

Voici les résultats :

Voici une histoire de super-héros :

Superman n'est pas super, tout le monde le sait. Il est juste moyen.
Il est moche et c'est très compliqué à croire quand on le voit.
Mais il a aussi un super pouvoir!
C'est celui de faire tomber la pluie quand il pleut trop. Surtout quand il pleut très fort, comm

Pour l'autre exemple, j'ai défini un texte de départ "Il était une fois un crabe mignon que ses parents avaient appelé le petit Pince. Le petit Pince était d'une beauté à couper le souffle. Il avait six petites pattes rose bonbon parfaitement articulées comme tous les crabes de cocotier bien nés, mais sa carapace était spéciale."

J'ai définit les paramètres suivant :

- Top-p = 0.7
- Température = 1.0
- La longueur du texte = 75

Voici les résultats :

Il était une fois un crabe mignon que ses parents avaient appelé le petit Pince. Le petit Pince était d'une beauté à couper le souffle. Il avait six petites pattes rose bonbon parfaitement articulées comme tous les crabes de cocotier bien nés, mais sa carapace était spéciale. Elle était d'un rose plus vif que le reste de son corps, et ses pinces étaient plus courtes et plus pointues. Ses grands yeux étaient de la même couleur que sa carapace. Le petit Pince avait une coquille qui n'

Conclusion : C'est à ce jour le modèle le plus performant sur les données en français. Il peut générer du texte en fonction d'une demande faite par l'utilisateur avec un thème défini par exemple "Voici une histoire de Noël", ou la suite d'un texte écrit par l'utilisateur.

Le script de modèle cédille, vous est présentée en annexe 9.5 page 69.

6 L'application

6.1 La base de données

Afin de stocker les différents types d'informations nécessaires au fonctionnement de l'application, j'ai implémenté une base de données SQL avec le système de gestion de bases de données relationnelles PostgreSQL.

La base de données comprend deux tables :

- Une table **User** : elle contient les informations personnelles de l'utilisateur, l'id, le nom, le prénom, l'email, le mot de passe hasher
- Une table **Histoire** : elle contient l'id et les histoires générées.

Les différentes informations de la base de données peuvent être visualisées dans l'image ci-dessous grâce à la représentation en diagramme ou **MCD**.

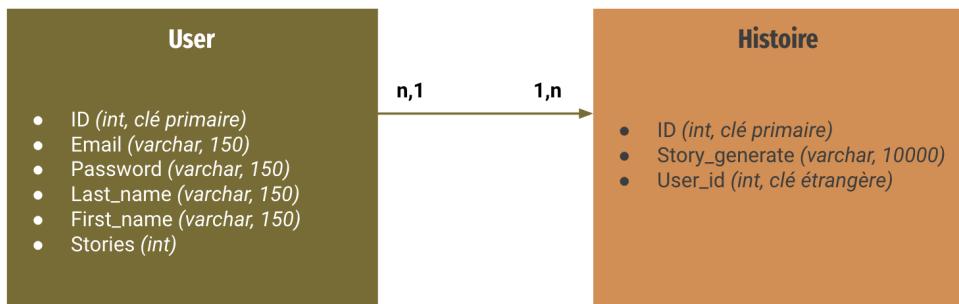


FIGURE 9 – Représentation MCD

Dans l'image ci-dessus, nous visualisons les deux tables qui ont une relation "OneToMany" avec les cardinalités n,1 et 1,n, les cardinalités se traduisent par la présence de clés étrangères user_id dans la table histoire :

- La cardinalité **n,1** : User peut générer et récupérer une à plusieurs histoires.
- la cardinalité **1,n** : les histoires appartiennent à un seul User.

6.1.1 Implémentation de la base de données

J'ai fait le choix de gérer la base de données via un ORM : SQLAlchemy. Cette solution me permet de manipuler la base relationnelle via une abstraction sous forme de classes.

Voici comment les tables ont été implémentés via SQLAlchemy dans le fichier models.py :

```

class histoire(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    story_generate = db.Column(db.String(10000))
    #story = db.Column(db.String(1000))
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))


class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(150), unique=True)
    password = db.Column(db.String(150))
    last_name = db.Column(db.String(150))
    first_name = db.Column(db.String(150))
    storys = db.relationship('histoire')

```

Voici un tableau récapitulant les requêtes SQL qui seront misent en œuvre dans l'application et leur transposition via SQLAlchemy :

Requête	SQL	Transposition via SQLAlchemy
Ajouter un nouvel utilisateur dans la table User	INSERT INTO User VALUES (first_name, last_name, email password);	<pre> new_user = User(email=email, first_name=first_name, password=generate_password_hash(password1, method='sha256')) db.session.add(new_user) db.session.commit() </pre>
Vérifier qu'un email est déjà présent dans la table User	SELECT email FROM User WHERE email LIKE email;	User.query.filter_by(email=email).first()
Ajouter une prédiction dans la table histoire	INSERT INTO histoire VALUES (story_generate, user_id);	<pre> new_histoire = histoire(story_generate=text, user_id=user_id) db.session.add(new_histoire) db.session.commit() </pre>
Récupérer toutes les prédictions réalisées par l'utilisateur connecté	SELECT story_generate FROM histoire WHERE user_id	Story = histoire.query.all()

A noter : Dans le cadre de l'ajout de l'utilisateur, j'ai implémenté le hachage du mot de passe.

6.2 Le développement de l'application

J'ai choisi de développer l'application à l'aide de Flask qui est un framework open-source de développement web en Python. L'application tend à fournir une interface et une base de

données simples, ce framework est largement suffisant pour répondre au besoin exprimé.

6.2.1 Le parcours utilisateur

Le parcours de l'utilisateur commence sur la page d'inscription qui lui propose de créer un compte via un formulaire ou s'il en a déjà un, lui propose de se connecter.

Si l'utilisateur choisit la création d'un nouveau compte, il doit renseigner :

- Nom
- Prénom
- Email
- Mot de passe

Une fois le compte créé, l'utilisateur est redirigé vers la page de connection.

Si le formulaire comprend des erreurs lors de la saisie, un message sur l'erreur en question lui est transmis, l'utilisateur doit saisir à nouveau les informations sans erreurs dans le formulaire afin de valider son inscription.

Après l'authentification, il obtient l'accès à la page d'accueil (seuls les utilisateurs authentifiés peuvent accéder à l'application.) sur cette page d'accueil, il peut accéder via un bouton ou un menu déroulant à la page qui lui permet de générer du texte..

Une fois sur la page de génération de texte, il peut :

- Saisir un texte à l'endroit indiqué ;
- Modifier les paramètres ;
- Générer du texte en effectuant un clic sur le bouton générer.

Après avoir effectué le clic, un loader s'affiche, indiquant que la prédiction est en cours, une fois la prédiction effectuée, la page de génération s'affiche à nouveau avec le texte prédit (Le texte est automatiquement enregistré en base de données.).

Il peut ensuite accéder à la page mon compte, cette page lui permet de supprimer toutes les histoires qu'il a générées.

Les utilisateurs peuvent se déconnecter et être automatiquement redirigés vers la page de connexion une fois leur navigation sur le site terminé.

L'image ci-dessous vous présente le parcours utilisateur :

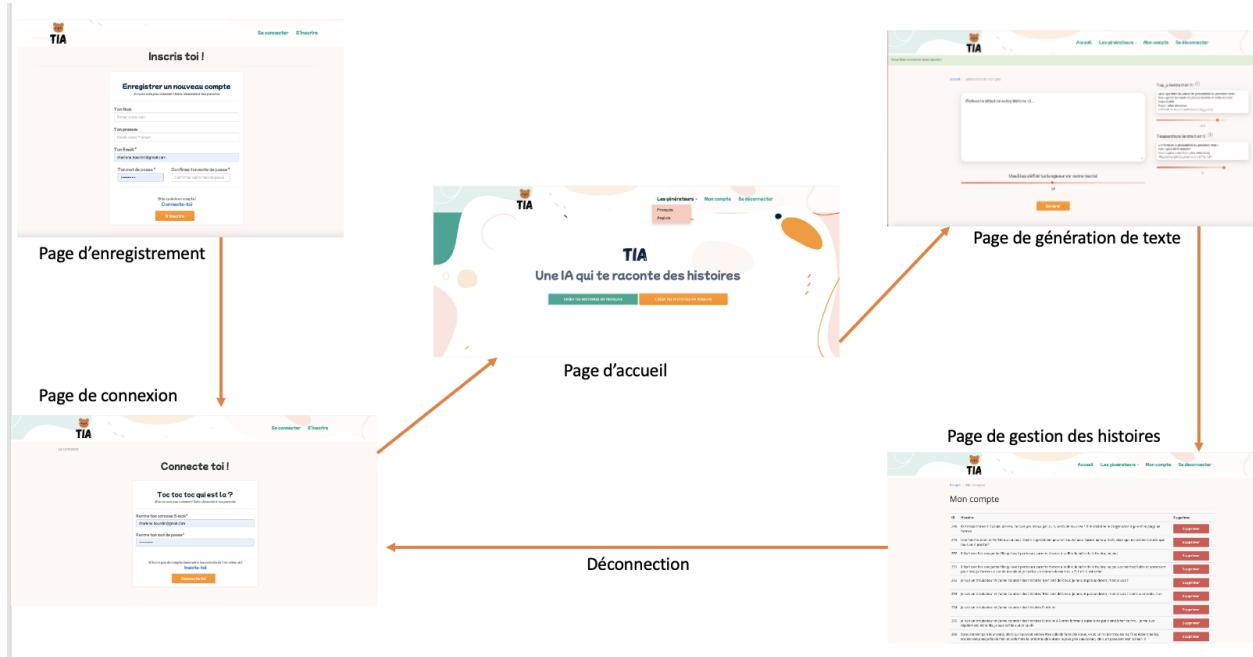


FIGURE 10 – Parcours utilisateur

6.2.2 L'architecture de l'application

L'image ci-dessous présente l'architecture de l'application :

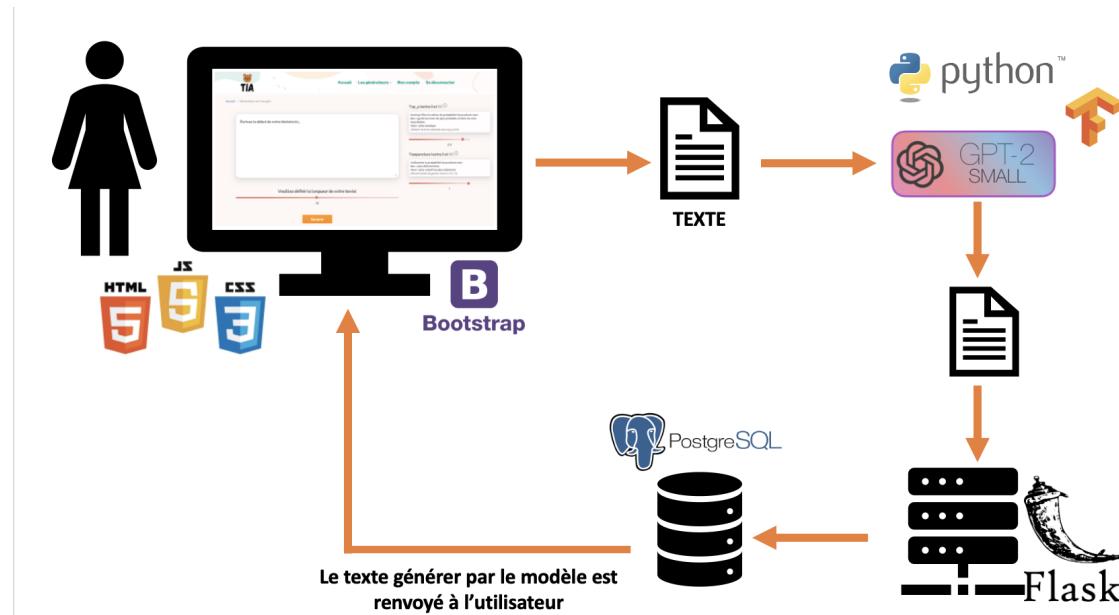


FIGURE 11 – Architecture de l'application

6.2.3 Cration de l'interface

L'interface (cote front-end) a te construite en HTML, CSS, JS et Bootstrap (ensemble d'outils qui contient des codes HTML et CSS, des formulaires, boutons et autres lments interactifs dj mis en forme).

Pour communiquer et transmettre les informations au serveur, les mthodes GET et POST ont te utilises. HTTP dfinit un ensemble de mthodes de requtes qui indiquent l'action que l'on souhaite raliser sur la ressource indique.

- La mthode GET demande une reprsentation de la ressource spcifie. (utilise uniquement pour la rcupration de donnes)
- La mthode POST est utilise pour envoyer une entit vers la ressource indique.

L'ensemble des interfaces vous est prsent en annexe 9.7 page 73.

6.2.4 Cration du back-end de l'application

Cte back-end, Flask un framework open-source de dveloppement web en Python a te utilis comme web-serveur.

Voici la liste des briques fonctionnelles implmentes dans le back-end de l'application :

- La gestion des routes lies aux vues du front-end.
- L'implmentation de la base de donnes via l'ORM SQLAlchemy
- L'implmentation d'un systme de Login via la librairie Flask_login
- L'implmentation d'un systme de monitoring via la librairie flask_monitoringdashboard

Voici l'arbre du dossier de l'application :

```
.  
|   └── modele  
|       |   └── run_grimms_en_3000  
|       |   └── run_grimms_french_3000  
|       |   └── __init__.py  
|       |   └── generate_english.py  
|       |   └── generate.py  
|       |   └── generate_french.py  
|       └── website  
|           └── static  
|               |   └── css  
|               |   └── fonts  
|               |   └── images  
|               |   └── js  
|               └── templates  
|                   |   └── signin.html  
|                   |   └── signup.html  
|                   |   └── base.html  
|                   |   └── base_logout.html  
|                   |   └── base_generate.html  
|                   |   └── admin.html  
|                   |   └── index.html  
|                   |   └── generate_english.html  
|                   |   └── generate_french.html  
|               └── __init__.py  
|       └── auth.py  
|       └── manage.py  
|       └── views.py  
|       └── config.cfg  
|       └── flask_monitoringdashboard.db  
|       └── main.py
```

FIGURE 12 – Arborescence de l'application d'interface graphique

6.2.5 Gestion des sessions utilisateur

La librairie Flask Login permet de déterminer les algorithmes de login, de log out et de sign up. Elle permet également de déterminer les pages nécessitant un login de la part d'un utilisateur. J'ai complété le dispositif par un password haché en sha256 implémenté grâce à la librairie **werkzeug**.

Le module **login_user** permet de faire apparaître un message d'erreur lorsque les utilisateurs ne sont pas enregistrés. Cela permet de gérer aussi des erreurs ou des fautes de frappe lors de la connexion.

Le tableau ci-dessous présente la liste des routes de l'application, leur utilisation et la nécessité d'être authentifié pour pouvoir y accéder. :

Route	Usage	Méthode	Nécessité Authentification
/	Racine de l'application	GET	OUI
/sign_up	Inscription	GET / POST	NON
/signin	Connection	GET / POST	NON
/generate_grimms_french	Génération de texte en français sur le thème des contes des frères Grimms	GET / POST	OUI
/generate_grimms_english	Génération de texte en anglais sur le thème des contes des frères Grimms	GET / POST	OUI
/Account	Gestion des histoires	GET / POST	OUI

6.3 Les tests unitaires

Afin de valider le bon fonctionnement de l'application, j'ai implémenté des tests unitaires.

Le test unitaire est un procédé permettant de s'assurer du fonctionnement correct d'une partie déterminée d'un logiciel ou d'une portion d'un programme.

Il s'agit pour le programmeur de tester un module, indépendamment du reste du programme, ceci afin de s'assurer qu'il répond aux spécifications fonctionnelles et qu'il fonctionne correctement en toutes circonstances.

Voici le récapitulatif des tests unitaires implémentés

Test	Unités à tester
L'appel des routes renvoie un code HTTP 200	Routes : <ul style="list-style-type: none"> • / (Utilisateur non authentifié) • /sign_up • /signin • /generate_frenc • /generate_english • /account
Présence d'un message d'alerte attendu dans le code HTML	Routes : <ul style="list-style-type: none"> • /sign_up • /signin
Test de redirection effective	<ul style="list-style-type: none"> • Après l'enregistrement d'un nouveau compte, redirection vers la page de connexion. • Après la connexion à son compte, redirection vers la page d'accueil

Le script des tests unitaires vous est présenté en annexe 9.8 page 79.

6.4 Le monitoring

J'ai déterminé les éléments critiques de l'ensemble regroupant l'interface graphique appliquant la matrice d'évaluation des risques.



FIGURE 13 – Matrice d'évaluation des risques

Risque	Gravité	Probabilité	Score
Pas de réponse de l'API	Fatal	Probable	8
Interface graphique de répond pas	Fatal	Probable	8
Erreur d'enregistrement en BDD	Grave	Probable	4
Trop de requêtes sur l'API	Très grave	Peu probable	3
Trop de requêtes en BDD	Grave	Peu probable	2

Afin de suivre l'apparition de ces risques, j'ai implémenté un système de monitoring en charge de journalisé l'activité de l'application et générer des alertes le cas-échéant. J'ai choisi d'utiliser le module **flask_monitoringdashboard** pour sa simplicité.

Ce service permet de journalisé les erreurs et les trier selon plusieurs filtres : Pour accéder à la page de monitoring, il faut utiliser la route **/dashboard**, elle nous amène sur la vue d'ensemble qui contient un tableau avec des statistiques générales pour chaque

point de terminaison de l'application.

La colonne Niveau de surveillance est utilisée pour définir le niveau de surveillance pour chaque point de terminaison :

Présentation du tableau de bord						
	Nombre de visites	Durée médiane des requêtes (ms)				
Point final	Aujourd'hui	Les 7 derniers jours	Globalement	Dernière demande	Niveau de surveillance *	
vues.story_grimms_french	sept	34 ¹	582	il y a 4 heures	0 1 2 3	
vues.accueil	12	20	278	il y a 46 minutes	0 1 2 3	
vues.histoire	3	5	201	il y a 5 heures	0 1 2 3	
auth.login	21	31	152	Il ya 1 heure	0 1 2 3	
vues.effacer	0	0	151	Il ya 1 semaine	0 1 2 3	
vues.harry_francais	3	13	101	il y a 5 heures	0 1 2 3	
auth.sign_up	6	9	52	Il ya 1 heure	0 1 2 3	
auth.logout	sept	11	36	il y a 2 heures	0 1 2 3	
vues.index	0	0	13	3 semaines plus tôt	0 1 2 3	
vues.rapport	0	0	0		0 1 2 3	

[Précédent](#) [Suivant](#)

Le dashboard de flask peut collecter le nombre d'utilisation, les durées des requêtes (performance), des informations sur des requêtes lentes (valeurs aberrantes) et les performances par ligne de code (profileur)

Le niveau de surveillance :

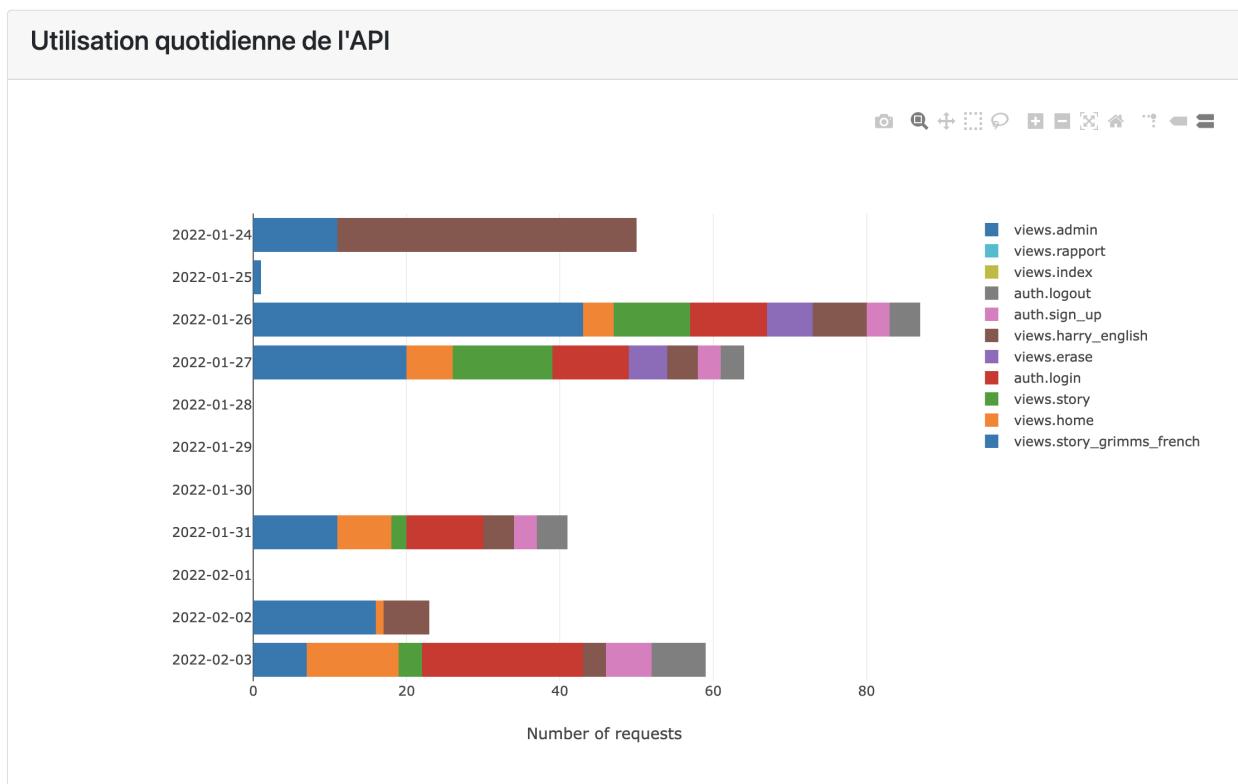
- **Niveau 0** : désactivé, ne collecte que le temps depuis la dernière demande
- **Niveau 1** : Niveau 0 + Utilisation et performance
- **Niveau 2** : Niveau 1 + Valeurs aberrantes
- **Niveau 3** : Niveau 2 + Profileur

Il y a également des graphiques intégrés pour tous les points finaux (avec un niveau de surveillance == 1)

- **Utilisation horaire de l'API** : ce graphique montre le nombre total d'appels (tous les points de terminaison) par heure pour tous les jours de l'intervalle sélectionné. L'intervalle est sélectionné avec les champs Date de début et Date de fin dans les options du graphique.

- Utilisation de l'API multi-version : ce graphique montre le nombre de résultats pour les points de terminaison sélectionnés dans les versions d'application sélectionnées. Le champ APP_VERSION dans la configuration détermine la version actuelle.
- **Utilisation quotidienne de l'API** : ce graphique montre le nombre de visites chaque jour pour chaque point de terminaison dans l'intervalle sélectionné. L'intervalle est sélectionné avec les champs Date de début et Date de fin dans les options du graphique.
- **Performances de l'API** : ce graphique montre la durée de toutes les demandes, y compris la durée médiane, pour les points de terminaison sélectionnés.
- **Rapports** : sur cette page, des graphiques comparant les performances des applications entre deux périodes peuvent être générés. Les performances sont comparées en termes de latence moyennes et de distribution du code d'état.

Le graphique ci-dessous présente l'utilisation quotidienne de l'API du 24 janvier 2022 au 3 février 2022 :



Dans ce graphique, on constate que le 26 janvier, l'API a été le plus utilisé avec un nombre de requêtes dépassant les 80. «views.story_grimms_french» est le point de terminaison qui a subit le plus de requêtes ce jour là.

L'ensemble des graphiques vous est présenté en annexe 0.9 page 81.

6.4.1 Alerting

Afin d'être alerté en cas de dysfonctionnement de l'application, j'ai mis en place un système d'alerte par e-mail en utilisant le module de journalisation inclus dans python. J'utilise SMTPHandler qui me permet de créer un serveur de messagerie en configurant cet objet au niveau de la surveillance ERROR. J'ai également créé un FileHandler configuré au même niveau de surveillance, cet objet écrira des journaux dans le fichier error.txt en cas d'erreur dans le backend Flask.

```
file_handler = FileHandler('logs/error.txt')
file_handler.setLevel(ERROR)
app.logger.addHandler(file_handler)
mail_handler = SMTPHandler(
    mailhost='127.0.0.1',
    fromaddr='server@domain.com', #adresse du serveur
    toaddrs=['charlene.hourdin@gmail.com'],
    subject='Application Error'
)
mail_handler.setLevel(ERROR)
mail_handler.setFormatter(Formatter(
    '[%(asctime)s] %(levelname)s in %(module)s: %(message)s'
))
app.logger.addHandler(mail_handler)

dashboard.bind(app)
```

7 Les axes d'améliorations

Voici une liste des tâches à effectué :

- Ajouter l'inférence du modèle fr-boris à l'application.
- Faire l'inférence du modèle GPT-J pour les données en anglais et l'ajouter à l'application.
- Modifier la base de données en ajoutant une table pour y stocker les histoires générées dans la langue anglaise.
- Modifier la page mon compte en permettant à l'utilisateur de trier les histoires par langue.
- Effectuer du Transfert-learning avec le modèle fr-boris de Cedille.ai sur mes données en français.
- Effectuer du Transfert-learning avec le modèle GPT-J sur mes données en anglais.

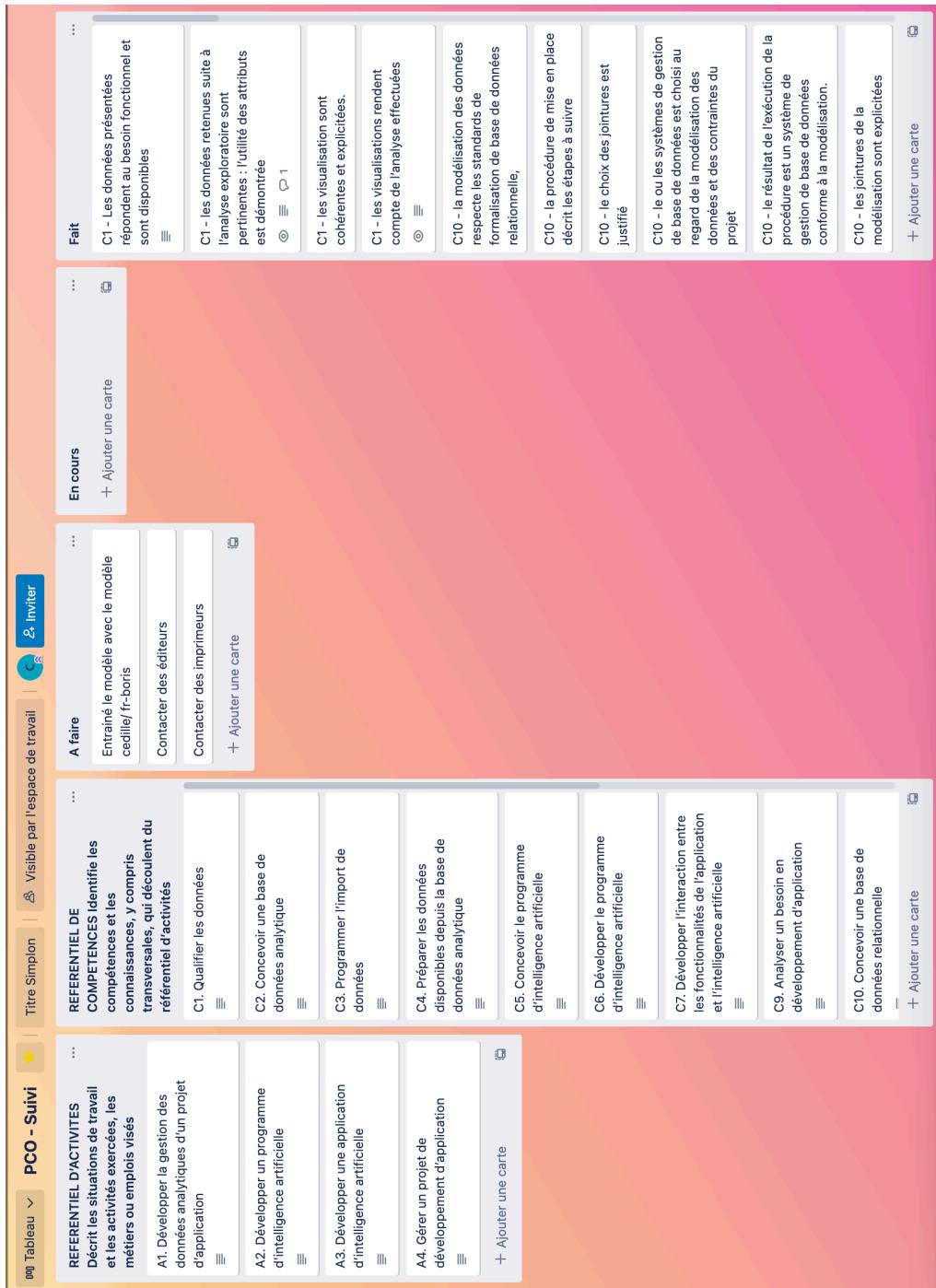
8 Conclusion

Ce projet est passionnant, mais cela reste un sujet R&D complexe. J'ai rencontré des problèmes de ressources de calcul (GPU et TPU), que vais essayer de solutionner dans les mois à venir afin de pouvoir améliorer mes modèles et proposés une application performante aux utilisateurs.

Dans cette attente, je vais déployer l'application en utilisant l'inférence **fr-boris** et **GPT-J** en y ajoutant une page pour la politique de confidentialité et pour les conditions d'utilisation en y référençant le travail des développeurs de **Cedille.ai** et **EleutherAI**.

9 Annexe

9.1 Gestion de projet - Tableau de bord Trello



9.2 Analyse exploratoire

```
In [43]: import pandas as pd
import numpy as np
import string, os
import matplotlib.pyplot as plt
import spacy
import re
import glob

from wordcloud import WordCloud
from PIL import Image
```

```
In [44]: from google.colab import drive # Se connecter à une google drive
drive.mount('/content/drive', force_remount=True)

Mounted at /content/drive
```

```
In [45]: grimms = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/Conte-de-fée'))
```

```
In [46]: andersen = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/andersen'))
```

```
In [47]: grimms_en = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/Grimm'))
```

```
In [48]: andersen_en = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/Andersen'))
```

Fonction Wordcloud des mots les plus fréquent dans chaque histoire

```
In [49]: def plotWordFrequency(input):
    text=""
    for book in input:
        with open(book) as f:
            print('Reading', book.split('/')[-1])
            text += f.read()
        #create the wordcloud object
        exclude_mots = ['d', 'du', 'de', 'la', 'des', 'le', 'et', 'est', 'elle']
        wordcloud = WordCloud(background_color='white', max_words=50).generate(text)
        #display the cloud
        plt.imshow(wordcloud)
        plt.axis('off')
        plt.show()
```

Mots les plus fréquent

Conte grimms en français

```
In [50]: grimms_wordcloud_fr = plotWordFrequency(grimms)
```

Reading fre_A_riddling_tale.txt



Reading fre_Bearskin.txt



Reading fre_Brides_on_their_trial.txt



Reading fre_Cat_and_mouse_in_partnership.txt



Reading fre_Cinderella.txt



Reading fre_Clever_Else.txt

A word cloud for the French story 'Le Loup et la Chèvre'. The words are arranged in a shape that suggests a wolf's head and a goat's head facing each other. The text includes: du en elle ventre se chevreaux est petite, et il le la de Mais le loup dans pas une, enfants sa pierres nous patte maman forest un, que comme pour la vieille était voix loup petite, l'un tout que maison assier la vieille était loup petite, dit que loup loup loup loup.

Reading fre_The_wonderful_musician.txt

A word cloud for the French story 'Le Géant'. The words are arranged in a shape of a giant's head. The text includes: que tu paure voilà forêt qui ce un dit le je te qu'il fois, au levraut force pas temps loup musicien là puis deux, d'un prit si compagnons son ménétrier lui que je, renard sa avec du le musicien une air.

Reading fre_The_young_giant.txt

A word cloud for the French story 'Tom Thumb'. The words are arranged in a shape of a small person's head. The text includes: qu'il elle pas je lui veux tout Non sa tu bien encore qui vous pour si géant, comme jeune sur avec du deux mais pour, ce son sur Le fermier deux fut mais, coup Quand fils avait plus dit une chevaux.

Reading fre_Tom_Thumb.txt

A word cloud for the French story 'Trusty John'. The words are arranged in a shape of a hand. The text includes: la dans lui monde vous sur que qui tout édu ne son un pas été sa cetu une bien dans le je Mais avec d'une se père il petit où nous dit leur moi qu' par plus Tom est Pouce.

Reading fre_Trusty_John.txt

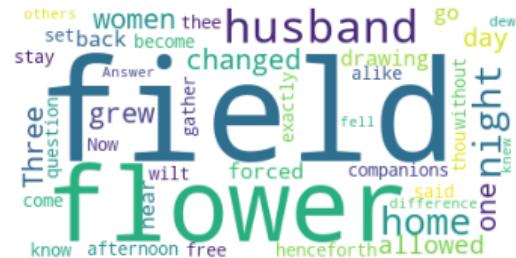
A word cloud for the French story 'Le Roi et la Fidèle Jean'. The words are arranged in a shape of a hand. The text includes: je était tout d'or pour sur chambre lui Mais pas avait quand il qui roi le roi ne fit si vous au tu encore dans la princesse que plus son fidèle Jean elle sa une fut le fidèle.

Conte Grimms en anglais

In [51]:

```
grimms_wordcloud_en = plotWordFrequency(grimms_en)
grimms_wordcloud_en
```

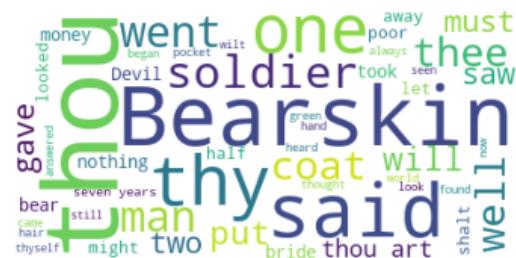
Reading eng_A_riddling_tale.txt



Reading eng_All-kinds-of-fur_(Allerleirauh).txt



Reading eng_Bearskin.txt



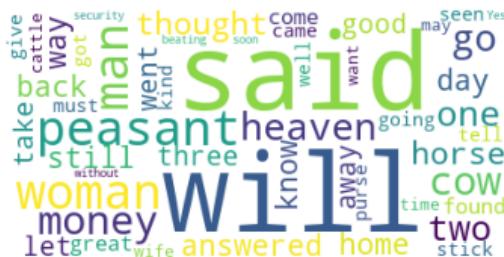
Reading eng_Brides_on_their_trial.txt



Reading eng_Brother_Lustig.txt



Reading eng_Wise_folks.txt



Conte d'Andersen en français

In [52]:

```
andersen_wordcloud_fr = plotWordFrequency(andersen)
andersen_wordcloud_fr
```

Reading fre_Something.txt



Reading fre_Clumsy_Hans.txt



Reading fre_Everything_in_its_proper_place.txt

de du le il
 leur pas dans une grande
 là lui avec vieille fut
 C'était sa place mais tout C'est
 pour maître sur où son bien
 ne plus au ce en comme bien
 avait elle se flûte château précepteur
 gardeuse d'oies saule

Reading fre_Five_peas_from_a_pod.txt

1 e et i la dans je
 dit pois elle cosse se
 une mère était veillée lui
 un qui avec son soleil
 jour petit Dieu
 ne au son sur
 mais de plus qu'il bien avec
 la fenêtre plus sa moi malade avait
 que de il que la petite petite fille

Reading fre_Little_Claus_and_big_Claus.txt

1 e et i la dans je
 sade la petit Claus tout si
 avait comme moi lui le sac au en
 plus Se te tu pour qu'il répondit
 que dans ne mais ce bien pour
 un pas dit je du dans la dans le
 pas qui la stécria était dans la femme
 qui la grand Claus le paysan son la femme

Reading fre_Little_Ida's_flowers.txt

de 1 e et i la
 tout dans ne elle leur quand
 plus Sophie avait là Ida
 étaient nous sur poupée
 mais ce une un grande
 que qui que bien tu était la petite
 pas et se la fleur sont aussi l'étudiant

Reading fre_Ole-Luk-Oie,_the_Dream-God.txt

et e la un la
 en est nous lui de la au une leur dans le
 Ferme plus aussi dans là une leur dans le
 du bien vous tout petit pour toute ne si
 Ferme l'œil Hjalmar que dit voilà qui
 tous avec comme mais avait était pas je

Reading fre_Pen_and_inkstand.txt



Conte d'Andersen en anglais

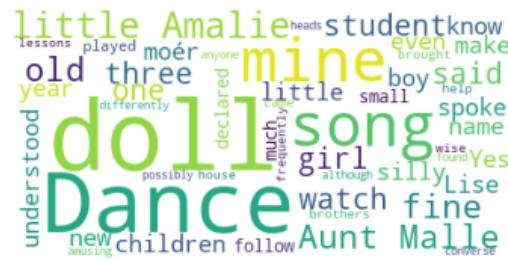
In [53]:

```
andersen_wordcloud_en = plotWordFrequency(andersen_en)
andersen_wordcloud_en
```

Reading eng_ "Beautiful".txt



Reading eng_ "Dance,_dance,_doll_of_mine!".txt



Reading eng_ "Something".txt



Reading eng_A_cheerful_temper.txt



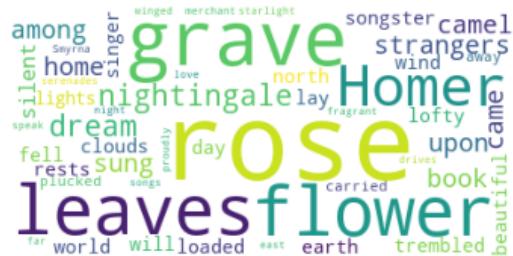
Reading eng_A_leaf_from_heaven.txt



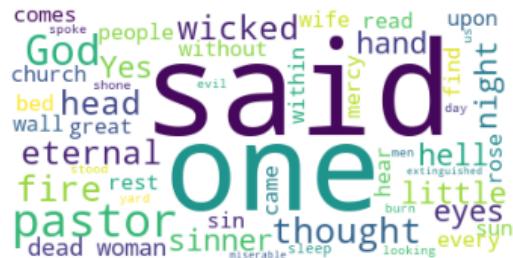
Reading eng_A_picture_from_the_ramparts.txt



Reading eng_A_rose_from_Homer's_grave.txt



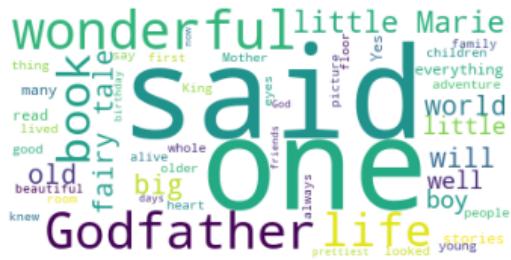
Reading eng_A_story.txt



Reading eng_A_story_from_the_sand_dunes.txt



Reading eng_What_the_whole_family_said.txt



Reading eng_Which_was_the_happiest?.txt



Creation des dataframes

```
In [54]: def create_df(input):
    story = []
    name = []
    for book in input:
        with open(book) as f:
            text = f.read()
            story.append(text)
        name.append(book.split('/')[-1])
    df = pd.DataFrame({'story':story, 'name_story':name})
    return df
```

```
In [55]: df_grimms_en = create_df(grimms_en)
```

```
In [56]: df_andersen_en = create_df(andersen_en)
```

```
In [57]: df_grimms_fr = create_df(grimms)
```

```
In [58]:
```

```
df_andersen_fr = create_df(andersen)
```

Fonction moyenne de caractère par histoire

In [59]:

```
def count_char(books):
    caractere = []
    name = []
    for book in books:
        with open(book) as f:
            text = f.read()
            caractere.append(len(text))
            name.append(book.split('/')[-1])
    df = pd.DataFrame({'nb_caractere':caractere, 'histoire':name})
    return df
```

In [60]:

```
def plot_characters(x,y):
    plt.figure(figsize=(15,10))
    plt.bar(x, y)
    plt.xlabel('Histoires', fontsize=20)
    plt.xticks(rotation=90)
    plt.ylabel('Nombre de caractères', fontsize=20)
    plt.title('Nombre de caractères par histoires', fontsize=20)
    for index,data in enumerate(y):
        plt.text(x=index , y =data+1 , s=f'{data}' , fontdict=dict(fontsize=20))
    plt.tight_layout()
    plt.show()
```

Fonction de nombres de mots par histoire

In [61]:

```
def count_word(books):
    words = []
    name = []
    for book in books:
        with open(book) as f:
            text = f.read()
            words.append(len(text.split()))
            name.append(book.split('/')[-1])
    df = pd.DataFrame({'nb_mots':words, 'histoire':name})
    return df
```

In [62]:

```
def plot_words(x,y):
    plt.figure(figsize=(15,10))
    plt.bar(x, y)
    plt.xlabel('Histoires', fontsize=20)
    plt.xticks(rotation=90)
    plt.ylabel('Nombre de mots', fontsize=20)
    plt.title('Nombre de mots par histoires', fontsize=20)
    for index,data in enumerate(y):
        plt.text(x=index , y =data+1 , s=f'{data}' , fontdict=dict(fontsize=20))
    plt.tight_layout()
    plt.show()
```

Création des dataframe avec le nombre de

mots et de caractere

In [63]:

```
df_chars_andersen_en = count_char(andersen_en)
df_chars_andersen_en
```

Out[63]:

	nb_caractere	histoire
0	13980	eng_"Beautiful".txt
1	1976	eng_"Dance,_dance,_doll_of_mine!".txt
2	14599	eng_"Something".txt
3	8385	eng_A_cheerful_temper.txt
4	5932	eng_A_leaf_from_heaven.txt
...
151	28704	eng_What_old_Johanne_told.txt
152	6706	eng_What_one_can_invent.txt
153	10670	eng_What_the_old_man_does_is_always_right.txt
154	5439	eng_What_the_whole_family_said.txt
155	8295	eng_Which_was_the_happiest?.txt

156 rows × 2 columns

In [64]:

```
df_chars_grimms_en = count_char(grimms_en)
df_chars_grimms_en
```

Out[64]:

	nb_caractere	histoire
0	654	eng_A_riddling_tale.txt
1	11225	eng_All-kinds-of-fur_(Allerleirauh).txt
2	10279	eng_Bearskin.txt
3	812	eng_Brides_on_their_trial.txt
4	21610	eng_Brother_Lustig.txt
...
196	15843	eng_The_young_giant.txt
197	8571	eng_Thumbling_as_journeyman_(Thumbling's_Trave...
198	11514	eng_Tom_Thumb.txt
199	16486	eng_Trusty_John.txt
200	8071	eng_Wise_folks.txt

201 rows × 2 columns

In [65]:

```
df_words_andersen_en = count_word(andersen_en)
df_words_andersen_en
```

Out[65]:

	nb_mots	histoire
0	2538	eng_"Beautiful".txt
1	379	eng_"Dance,_dance,_doll_of_mine!".txt
2	2789	eng_"Something".txt
3	1587	eng_A_cheerful_temper.txt
4	1120	eng_A_leaf_from_heaven.txt
...
151	5472	eng_What_old_Johanne_told.txt
152	1275	eng_What_one_can_invent.txt
153	2083	eng_What_the_old_man_does_is_always_right.txt
154	1022	eng_What_the_whole_family_said.txt
155	1587	eng_Which_was_the_happiest?.txt

156 rows × 2 columns

In [66]:

```
df_words_grimms_en = count_word(grimms_en)
df_words_grimms_en
```

Out[66]:

	nb_mots	histoire
0	131	eng_A_riddling_tale.txt
1	2177	eng_All-kinds-of-fur_(Allerleirauh).txt
2	1988	eng_Bearskin.txt
3	155	eng_Brides_on_their_trial.txt
4	4181	eng_Brother_Lustig.txt
...
196	3097	eng_The_young_giant.txt
197	1622	eng_Thumbling_as_journeyman_(Thumbling's_Trave...
198	2287	eng_Tom_Thumb.txt
199	3145	eng_Trusty_John.txt
200	1598	eng_Wise_folks.txt

201 rows × 2 columns

In [67]:

```
df_chars_andersen_fr = count_char(andersen)
df_chars_andersen_fr
```

Out[67]:

	nb_caractere	histoire
0	14271	fre_"Something".txt
1	7023	fre_Clumsy_Hans.txt
2	13402	fre_Everything_in_its_proper_place.txt

nb_caractere	histoire
3	fre_Five_peas_from_a_pod.txt
4	fre_Little_Claus_and_big_Claus.txt
5	fre_Little_Ida's_flowers.txt
6	fre_Ole-Luk-Oie,_the_Dream-God.txt
7	fre_Pen_and_inkstand.txt
8	fre_Soup_from_a_sausage_skewer.txt
9	fre_The_Nightingale.txt
10	fre_The_angel.txt
11	fre_The_bell.txt
12	fre_The_bottle_neck.txt
13	fre_The_brave_tin_soldier.txt
14	fre_The_butterfly.txt
15	fre_The_daisy.txt
16	fre_The_darning-needle.txt
17	fre_The_elderbush.txt
18	fre_The_emperor's_new_suit.txt
19	4978 fre_The_farm-yard_cock_and_the_weather-cock.txt
20	fre_The_fir_tree.txt
21	fre_The_flax.txt
22	fre_The_flying_trunk.txt
23	fre_The_garden_of_paradise.txt
24	14783 fre_The_gardener_and_the_noble_family.txt
25	6230 fre_The_happy_family.txt
26	3585 fre_The_jumper.txt
27	12243 fre_The_last_dream_of_the_old_oak.txt
28	4995 fre_The_little_match-seller.txt
29	44610 fre_The_little_mermaid.txt
30	4694 fre_The_moneybox.txt
31	19169 fre_The_neighbouring_families.txt
32	18734 fre_The_old_house.txt
33	13803 fre_The_old_street_lamp.txt
34	2020 fre_The_princess_and_the_pea.txt
35	9101 fre_The_puppet-show_man.txt
36	6568 fre_The_races.txt
37	24399 fre_The_shadow.txt
38	9048 fre_The_shepherdess_and_the_sweep.txt
39	5736 fre_The_shirt-collar.txt

	nb_caractere	histoire
40	9857	fre_The_silver_shilling.txt
41	4399	fre_The_snail_and_the_rosebush.txt
42	63286	fre_The_snow_queen.txt
43	10259	fre_The_snowman.txt
44	8850	fre_The_swineherd.txt
45	13727	fre_The_tinder-box.txt
46	13486	fre_The_toad.txt
47	37048	fre_The_travelling_companion.txt
48	16131	fre_The_ugly_duckling.txt
49	31360	fre_The_wild_swans.txt
50	22980	fre_Thumbelina.txt
51	10338	fre_What_happened_to_the_thistle.txt
52	9300	fre_What_the_old_man_does_is_always_right.txt

In [68]:

```
df_chars_grimms_fr = count_char(grimms)
df_chars_grimms_fr
```

Out[68]:

	nb_caractere	histoire
0	837	fre_A_riddling_tale.txt
1	12372	fre_Bearskin.txt
2	971	fre_Brides_on_their_trial.txt
3	5751	fre_Cat_and_mouse_in_partnership.txt
4	14680	fre_Cinderella.txt
...
110	5593	fre_The_wolf_and_the_seven_young_kids.txt
111	6631	fre_The_wonderful_musician.txt
112	15381	fre_The_young_giant.txt
113	11207	fre_Tom_Thumb.txt
114	15298	fre_Trusty_John.txt

115 rows × 2 columns

In [69]:

```
df_words_andersen_fr = count_word(andersen)
df_words_andersen_fr
```

Out[69]:

	nb_mots	histoire
0	2493	fre_"Something".txt
1	1282	fre_Clumsy_Hans.txt
2	2379	fre_Everything_in_its_proper_place.txt

nb_mots	histoire
3	fre_Five_peas_from_a_pod.txt
4	fre_Little_Claus_and_big_Claus.txt
5	fre_Little_Ida's_flowers.txt
6	fre_Ole-Luk-Oie,_the_Dream-God.txt
7	fre_Pen_and_inkstand.txt
8	fre_Soup_from_a_sausage_skewer.txt
9	fre_The_Nightingale.txt
10	fre_The_angel.txt
11	fre_The_bell.txt
12	fre_The_bottle_neck.txt
13	fre_The_brave_tin_soldier.txt
14	fre_The_butterfly.txt
15	fre_The_daisy.txt
16	fre_The_darning-needle.txt
17	fre_The_elderbush.txt
18	fre_The_emperor's_new_suit.txt
19	885 fre_The_farm-yard_cock_and_the_weather-cock.txt
20	fre_The_fir_tree.txt
21	fre_The_flax.txt
22	fre_The_flying_trunk.txt
23	fre_The_garden_of_paradise.txt
24	fre_The_gardener_and_the_noble_family.txt
25	fre_The_happy_family.txt
26	fre_The_jumper.txt
27	fre_The_last_dream_of_the_old_oak.txt
28	fre_The_little_match-seller.txt
29	fre_The_little_mermaid.txt
30	fre_The_moneybox.txt
31	fre_The_neighbouring_families.txt
32	fre_The_old_house.txt
33	fre_The_old_street_lamp.txt
34	fre_The_princess_and_the_pea.txt
35	fre_The_puppet-show_man.txt
36	fre_The_races.txt
37	fre_The_shadow.txt
38	fre_The_shepherdess_and_the_sweep.txt
39	fre_The_shirt-collar.txt

	nb_mots	histoire
40	1723	fre_The_silver_shilling.txt
41	762	fre_The_snail_and_the_rosebush.txt
42	11202	fre_The_snow_queen.txt
43	1848	fre_The_snowman.txt
44	1537	fre_The_swineherd.txt
45	2480	fre_The_tinder-box.txt
46	2377	fre_The_toad.txt
47	6425	fre_The_travelling_companion.txt
48	2866	fre_The_ugly_duckling.txt
49	5416	fre_The_wild_swans.txt
50	4062	fre_Thumbelina.txt
51	1798	fre_What_happened_to_the_thistle.txt
52	1698	fre_What_the_old_man_does_is_always_right.txt

In [70]:

```
df_words_grimms_fr = count_word(grimms)
df_words_grimms_fr
```

Out[70]:

	nb_mots	histoire
0	136	fre_A_riddling_tale.txt
1	2253	fre_Bearskin.txt
2	172	fre_Brides_on_their_trial.txt
3	1018	fre_Cat_and_mouse_in_partnership.txt
4	2573	fre_Cinderella.txt
...
110	991	fre_The_wolf_and_the_seven_youthful_kids.txt
111	1171	fre_The_wonderful_musician.txt
112	2855	fre_The_young_giant.txt
113	2054	fre_Tom_Thumb.txt
114	2767	fre_Trusty_John.txt

115 rows × 2 columns

Création du dataframe global

In [71]:

```
df_grimms_en['nb_caractere'] = df_chars_grimms_en['nb_caractere']
df_grimms_en['nb_mots'] = df_words_grimms_en['nb_mots']

df_andersen_en['nb_caractere'] = df_chars_andersen_en['nb_caractere']
df_andersen_en['nb_mots'] = df_words_andersen_en['nb_mots']
```

```
In [72]: df_grimms_fr['nb_caractere'] = df_chars_grimms_fr['nb_caractere']
df_grimms_fr['nb_mots'] = df_words_grimms_fr['nb_mots']

df_andersen_fr['nb_caractere'] = df_chars_andersen_fr['nb_caractere']
df_andersen_fr['nb_mots'] = df_words_andersen_fr['nb_mots']
```

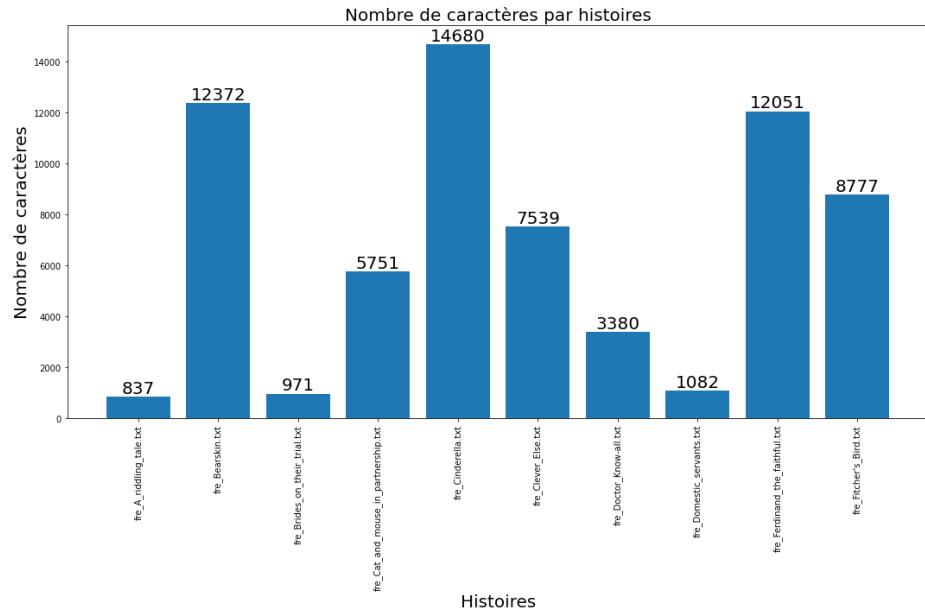
Statistique descriptives

Graphiques nombres de caractères par histoire

```
In [73]: # Echantillons des 10 premières histoires
samples_grimms_fr = df_grimms_fr[:10]
samples_andersen_fr = df_andersen_fr[:10]
samples_grimms_en = df_grimms_en[:10]
samples_andersen_en = df_andersen_en[:10]
```

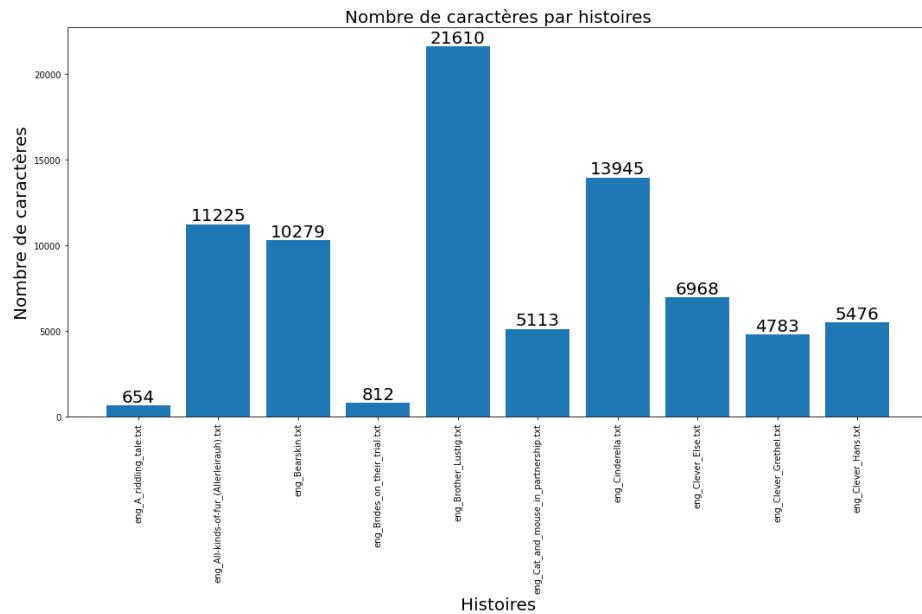
Nombre de caractères Grimms français

```
In [74]: num_chars_grimmm_fr = plot_characters(samples_grimms_fr['name_story'],samples_
```



Nombre de caractères Grimms anglais

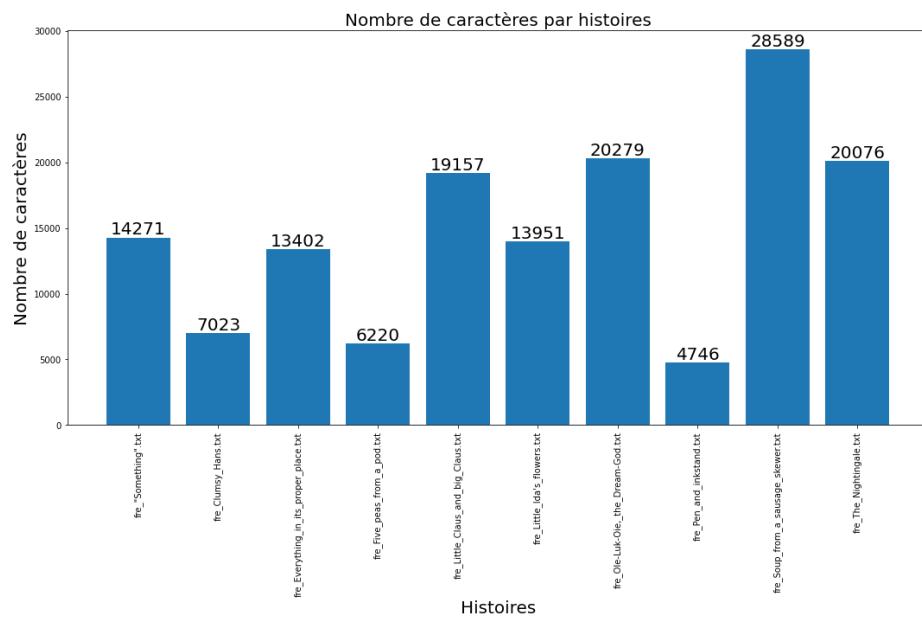
```
In [75]: num_chars_grimmm_en = plot_characters(samples_grimms_en['name_story'],samples_
```



Nombre de caractères Andersen français

In [76]:

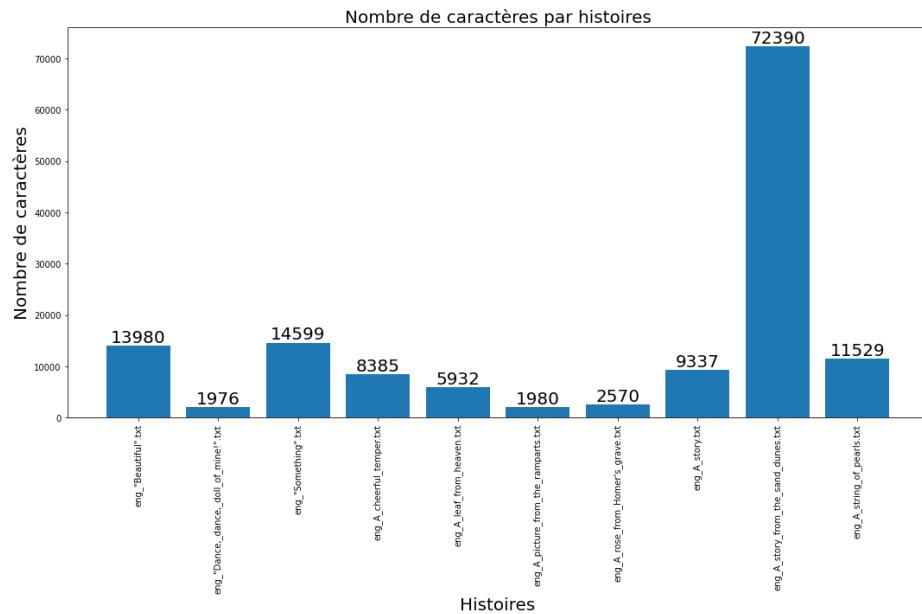
```
num_chars_andersen_fr = plot_characters(samples_andersen_fr[ 'name_story' ], samples_andersen_fr[ 'text' ])
```



Nombre de caractère Andersen anglais

In [77]:

```
num_chars_andersen_en = plot_characters(samples_andersen_en[ 'name_story' ], samples_andersen_en[ 'text' ])
```

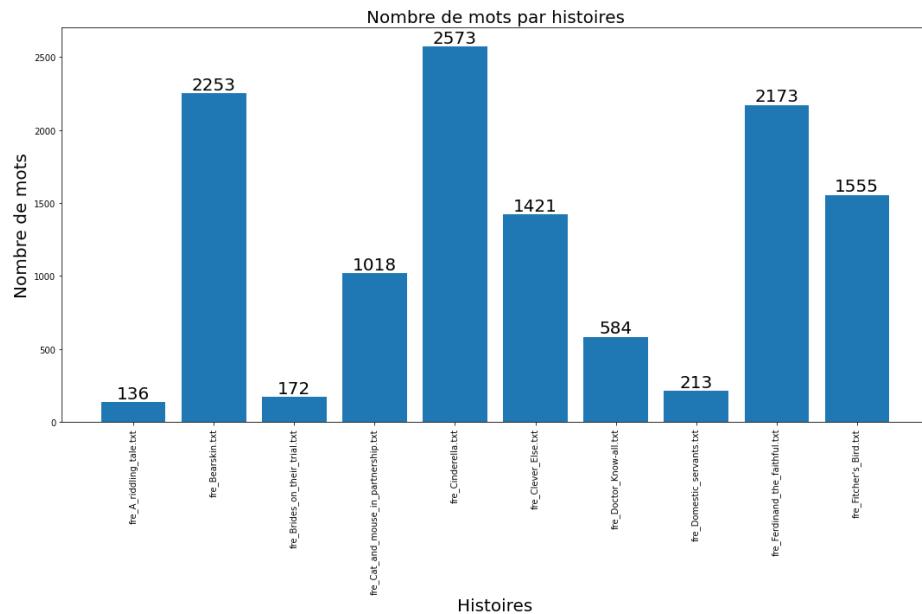


Graphiques nombres de mots par histoire

Nombre de mots Grimms français

In [78]:

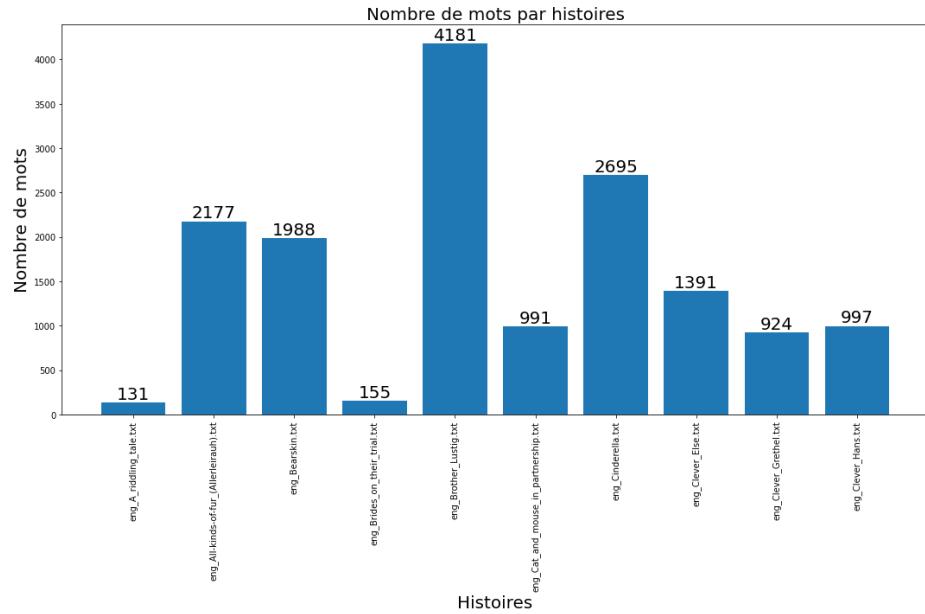
```
num_words_grimmm_fr = plot_words(samples_grimms_fr['name_story'], samples_grimmm)
```



Nombre de mots Grimms anglais

In [79]:

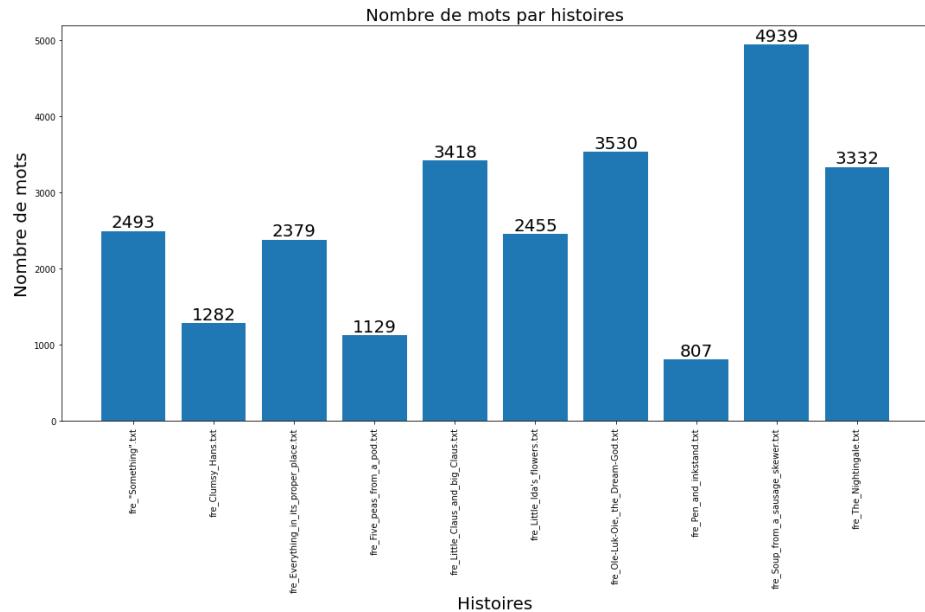
```
num_words_grimmm_en = plot_words(samples_grimms_en['name_story'], samples_grimms_en['text'])
```



Nombre de mots Andersen français

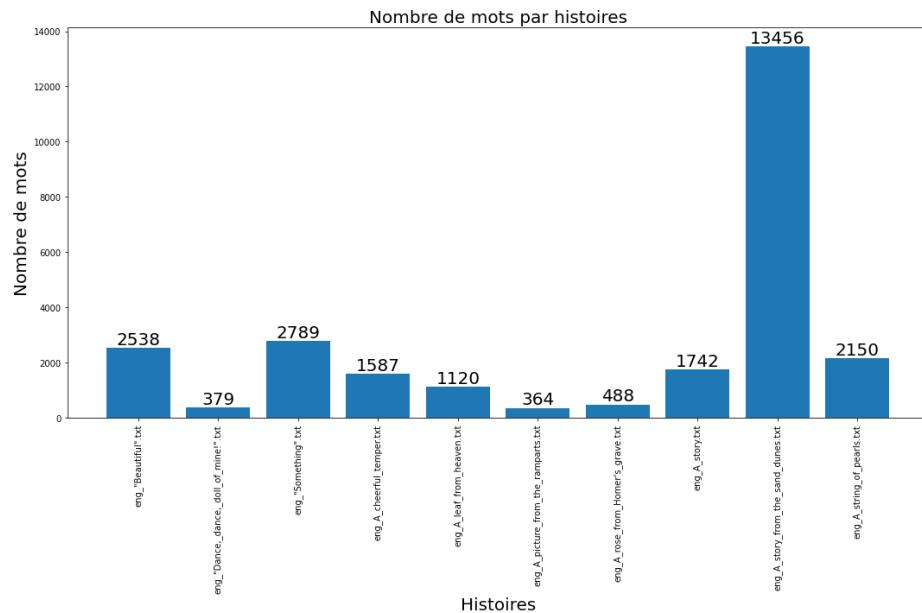
In [80]:

```
num_words_andersen_fr = plot_words(samples_andersen_fr['name_story'], samples_andersen_fr['text'])
```



Nombre de mots Andersen anglais

In [81]: `num_words_andersen_en = plot_words(samples_andersen_en['name_story'],samples_andersen_en['nb_mots'])`



Statistique descriptive des datasets

Andersen Français

In [82]: `df_andersen_fr.describe()`

Out[82]:

	nb_caractere	nb_mots
count	53.000000	53.000000
mean	14223.943396	2483.924528
std	11167.577593	1957.670042
min	2020.000000	350.000000
25%	6871.000000	1209.000000
50%	10338.000000	1848.000000
75%	18734.000000	3251.000000
max	63286.000000	11202.000000

Andersen anglais

In [83]: `df_andersen_en.describe()
print("Statistique descriptives des contes d'Andersen en anglais: \n")
df_andersen_en.describe()`

Statistique descriptives des contes d'Andersen en anglais:

Out[83]:

	nb_caractere	nb_mots
count	156.000000	156.000000
mean	14177.288462	2656.25641
std	14777.788292	2742.70214
min	1916.000000	364.00000
25%	5238.500000	978.50000
50%	9014.500000	1733.50000
75%	17818.500000	3313.50000
max	101291.000000	18673.00000

Grimms français

In [84]:

```
df_grimms_fr.describe()
```

Out[84]:

	nb_caractere	nb_mots
count	115.000000	115.000000
mean	7525.339130	1347.669565
std	5077.688742	913.776267
min	803.000000	136.000000
25%	3329.500000	593.500000
50%	6799.000000	1230.000000
75%	10635.500000	1953.500000
max	21650.000000	3879.000000

Grimms en anglais

In [85]:

```
df_grimms_en.describe()
#print("Statistique descriptives des contes de Grimms en anglais: \n")
```

Out[85]:

	nb_caractere	nb_mots
count	201.000000	201.000000
mean	7148.532338	1385.651741
std	5555.426148	1075.649933
min	654.000000	131.000000
25%	3023.000000	585.000000
50%	6095.000000	1156.000000
75%	10279.000000	1988.000000
max	44500.000000	8575.000000

9.3 Modèle BiLSTM + Nettoyage

```
In [1]: !pip install keras
Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-packages
(2.7.0)

In [2]:
import tensorflow as tf
import numpy as np
import string
import datetime
import random
import re
import glob
import matplotlib.pyplot as plt
import tensorflow.keras.utils as ku

from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional, Dropout
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import regularizers
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split

from google.colab import drive # Se connecter a google drive
```

```
In [3]:
from google.colab import drive # Se connecter a google drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
In [4]: !nvidia-smi
```

```
/bin/bash: nvidia-smi: command not found
```

```
In [5]: grimms = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/Conte-d&gt;
```

```
In [6]: andersen = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/ander&gt;
```

Nettoyer le texte

Cette étape consiste à nous assurer que nous nous débarrassons de toutes les ponctuations, doubles espaces, etc., de tout ce qui n'est pas un mot ou une langue parlée avant de le tokeniser. J'ai fait deux cycles de nettoyage tout en vérifiant que le texte a été nettoyé selon les normes requises. Je laisse le caractère de nouvelle ligne ('\n') parce que c'est ce sur quoi nous allons diviser les données pour créer un corpus à l'étape suivante.

```
In [20]:
import re
def clean_text(text):
    pattern = '(page|Page|PAGE)(\s+|\s+)([0-9]+)(.*$)'
    cleaned = re.sub('\s$', '', text, flags=re.MULTILINE)
    p = re.compile(pattern, re.MULTILINE)
```

```
cleaned = p.sub(" ", cleaned)
return cleaned
```

```
In [ ]:
text=""
for book in grimms:
    with open(book) as f:
        print('Reading', book.split('/')[-1])
        temp = f.read()
        text = text + clean_text(temp)
#text = text.lower()
```

Tokenize

Je créer un corpus en fractionnant les données au niveau du caractère '\n', puis en jetons le corpus. À partir de la sortie, je peux voir que j'ai un total de 33945 mots dans le corpus, c'est-à-dire que le vocabulaire est de 33945 mots et que l'index de mots mappe ce vocabulaire à leur représentation numérique qui est essentielle pour coder les séquences.

```
In [ ]:
# Tokenisation

corpus = text.split('\n')

tokenizer = Tokenizer()
tokenizer.fit_on_texts(corpus)

total_words = len(tokenizer.word_index) + 1

print(total_words)
print(tokenizer.word_index)
```

Création de séquences n_gram

C'est là que nous allons ligne par ligne dans le corpus et les convertissons en séquences. Nous itérons ensuite sur chaque séquence et ajoutons les deux premiers jetons, puis les trois premiers jetons, puis les quatre premiers jetons... jusqu'à la longueur de la séquence à la liste input_seq. Nous remplissons ensuite chaque élément de input_seq à la longueur de l'élément avec la longueur maximale.

Ensuite, nous divisons chaque élément de la liste input_seq par tout sauf le dernier pris comme prédicteur et le dernier jeton comme étiquette. De cette façon, nous avons créé nos prédicteurs et nos labels et sommes prêts à adapter les données.

```
In [ ]:
# création de séquences n_gram
input_seq = []
for line in corpus:
    token_list = tokenizer.texts_to_sequences([line])[0]
    for i in range(1, len(token_list)):
        ngram_seq = token_list[:i+1]
        input_seq.append(ngram_seq)

# séquences de pad
max_seq_len = max([len(x) for x in input_seq])
input_seq = np.array(pad_sequences(input_seq, maxlen=max_seq_len, padding='pr
```

```
# créer des prédicteurs et des labels
# je divise chaque élément de la liste input_seq
#sauf le dernier qui est pris comme prédicteur et le dernier jeton comme label
xs, labels = input_seq[:, :-1], input_seq[:, -1]
```

```
In [ ]:
print(len(input_seq))
print(len(input_seq[1]))
```

```
In [ ]:
x_train, x_test, y_train, y_test = train_test_split(xs, labels, test_size=0.2,
```

```
In [ ]:
y_train.shape
```

La modélisation

La première couche est une couche d'incorporation qui est essentielle pour la génération de texte car elle fournit une meilleure compréhension des mots en donnant des valeurs mathématiques similaires pour les mots ayant des significations similaires. Cela nous aidera à générer un texte plus logique.

Je suis allé avec une pile de couche LSTM bidirectionnelle car elle permet à la fois des informations en arrière et en avant sur la séquence à chaque étape de temps en préservant mieux le contexte. Vous pouvez également expérimenter une couche LSTM unidirectionnelle. Après quelques expériences, j'ai introduit une couche d'abandon entre eux, ce qui a entraîné une légère augmentation de la précision.

La sortie du LSTM, c'est-à-dire la couche cachée, est ensuite entrée dans une couche dense avec une activation softmax produisant des probabilités de mot entre 0 et 1.

Le résumé du modèle montre cette architecture de modèle et le nombre de paramètres totaux et entraînables. Nous sommes maintenant prêts à nous entraîner.

```
In [ ]:
# Modelling

model = Sequential([
    Embedding(total_words, 64, input_length=max_seq_len-1),
    Bidirectional(LSTM(256, return_sequences=True)),
    Dropout(0.6),
    Bidirectional(LSTM(256)),
    Dense(total_words/2, activation='relu', kernel_regularizer='l2'),
    Dense(total_words, activation='softmax')
])

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()
```

Entraînement

Comme mentionné précédemment, j'ai formé le modèle sur Google Colab pour accéder à la puissance du GPU et je l'ai fait sur plusieurs sessions de 20 époques en enregistrant les

poids à l'aide de Model Checkpoint à chaque époques. Même avec un GPU, chaque époque prenait environ 115 secondes. La durée totale de la formation était de quelques heures tout en surveillant constamment la précision et la perte.

```
In [ ]: print('train : ', X_train.shape)
print('test : ', X_test.shape)
```



```
In [ ]: cp_path = '/content/drive/My Drive/projet-TIA/checkpoint/andersen_french'
#cp_path = "drive/My Drive/projet-TIA/checkpoint"
#epoch = 20
# Model checkpoint

cp = tf.keras.callbacks.ModelCheckpoint(cp_path,
                                         save_weights_only=True,
                                         save_freq='epoch',
                                         verbose=1)

# Load weights
model.load_weights(cp_path)

# Train
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test,y_test))
```



```
In [ ]: import os
os.listdir(cp)
```



```
In [ ]: # Sauvegarder le modèle
!mkdir -p saved_model
model.save('/content/drive/My Drive/projet-TIA/saved_model/generated_text_BiLSTM.h5')
```

Performance train

```
In [ ]: model.evaluate(X_train, y_train)
```

Performance test

```
In [ ]: model.evaluate(X_test, y_test)
```

Précision du tracé

Un excellent moyen de surveiller la précision et la perte consiste à tracer leurs graphiques entre les sessions de formation pour vérifier l'amélioration du modèle. Dans la capture d'écran des graphiques de la dernière séance d'entraînement ci-dessous, nous pouvons voir que le modèle cesse de converger autour d'une précision de 86%, c'est-à-dire que la dernière séance d'entraînement a entraîné une amélioration de la précision de 0,1 à 0,2%.

```
In [ ]: plt.figure(figsize=(8, 5), dpi=100)
plt.title("Loss")
plt.plot(history.history['loss'], label="Train Dataset")
plt.plot(history.history['val_loss'], label="Validation Dataset")
plt.xlabel("Epoch")
```

```
plt.legend()
plt.show()
```

```
In [ ]:
plt.figure(figsize=(8, 5), dpi=100)
plt.title("Accuracy")
plt.plot(history.history['accuracy'], label="Train Dataset")
plt.plot(history.history['val_accuracy'], label="Validation Dataset")
plt.xlabel("Epoch")
plt.legend()
plt.show()
```

Générer du texte

Nous arrivons maintenant à la partie où nous commençons à générer des histoires inspirées par notre choix d'histoire. Cependant, nous devons fournir un texte de départ ou un point de départ et le nombre de mots qu'il devrait prédire après la semence.

Pour ce faire, le texte d'origine doit être tokenisé et complété, puis prédire le jeton suivant à l'aide du modèle entraîné qui est ensuite reconvertis en mot et imprimé avec le texte d'origine.

Cela demande un peu d'expérimentation. Par exemple, pour la première ligne, j'ai défini le texte de départ sur «Il était une fois» et le nombre de mots à prédire comme 6. Le résultat était «le monde est devenu noir devant mes yeux». Un peu sombre mais pas mal. Sur cette base, j'ai défini le texte de départ de la ligne suivante comme «j'étais» et le nombre de mots sur 4. Le modèle a prédit «j'étais dans les égouts». Cela correspond à l'ambiance sombre.

J'ai continué de la même manière pour les deux lignes suivantes et voici à quoi ressemblent les quatre premières lignes:

le monde est devenu noir devant mes yeux j'étais dans les égouts en descendant la rivière Rollin de montagne pourquoi ne m'entends-tu pas pleurer

Comme nous avons supprimé toutes les ponctuations et les avons rendues minuscules, le texte généré est dans le même format. Il est intéressant de voir le modèle générer des mots comme «ya» au lieu de «vous» et «rouler / pleurer» au lieu de «rouler / pleurer» pour suivre le style d'écriture de Bob Dylan.

```
In [ ]:
# importer le modèle
new_model = tf.keras.models.load_model("/content/drive/MyDrive/projet-TIA/sav")
```

```
In [ ]:
def generate_text(seed_text, next_words, new_model, max_sequence_len):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_seq_len-1, padding='post')
        predict = model.predict(token_list)
        classes = np.argmax(predict, axis=1)
        output_word = ""
        for word, index in tokenizer.word_index.items():
            if index == classes:
                output_word = word
                break
```

```
    seed_text += " " + output_word
return seed_text
```

```
In [ ]: print(generate_text("Il était une fois un prince qui vivait dans son château et il aimait les princesses. Un jour, il a rencontré une jeune fille dans la forêt et il a été émerveillé par sa beauté. Il a alors décidé de l'épouser. Le mariage a été célébré dans le château avec une grande fête. Depuis ce jour, le prince et la princesse sont heureux et ils ont beaucoup d'enfants. Ils vivent tous ensemble dans le château et ils sont très contents."))
```

```
In [ ]:
```

9.4 Modèle GPT-2

```
In [ ]: %tensorflow_version 1
#!/usr/bin/env python
#pip install tensorflow==1.13.1
!pip install -q gpt-2-simple
```

```
In [ ]: !git clone https://github.com/aquadzn/gpt2-french.git
```

```
In [ ]: #!pip install tensorflow-gpu==1.15.2 # Pour utiliser tensorflow.contrib, il
#!pip install tensorflow-estimator==2.1.0
#from tensorflow.python.compiler.tensorrt import trt_convert as trt
import gpt_2_simple as gpt2
import glob
import nltk

from datetime import datetime
from google.colab import files
```

```
In [ ]: gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
    print('Not connected to a GPU')
else:
    print(gpu_info)
```

```
In [ ]: gpt2.download_gpt2(model_name="355M")
```

```
In [ ]: gpt2.download_gpt2(model_name="124M")
```

```
In [ ]: import os
model_name = "124M"
if not os.path.isdir(os.path.join("models", model_name)):
    print(f"Downloading {model_name} model...")
    gpt2.download_gpt2(model_name=model_name)
```

```
In [ ]: gpt2.mount_gdrive()
```

Conte de grimms en français

```
In [ ]: import glob
```

```
In [ ]: grimms = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/Conte-de-Grimm/*.txt'))
```

```
In [ ]: import re
def clean_text(text):
    pattern = '(page|Page|PAGE)(\s+|\s+)([0-9]+)(.*)$'
    cleaned = re.sub('\s+', ' ', text, flags=re.MULTILINE)
    p = re.compile(pattern, re.MULTILINE)
```

```

cleaned = p.sub(" ", cleaned)
return cleaned

```

```

In [ ]:
text=""
for book in grimms:
    with open(book) as f:
        print('Reading', book.split('/')[-1])
        temp = f.read()
        text = text + clean_text(temp)

```

```

In [ ]:
with open("/content/drive/My Drive/Cleaned_grimms_french.txt", 'w') as f:
    f.writelines(text)

```

```

In [ ]:
grimms_clean = "Cleaned_grimms_french.txt"

```

```

In [ ]:
gpt2.copy_file_from_gdrive(grimms_clean)

```

Entrainement du modèle

```

In [ ]:
sess = gpt2.start_tf_sess()

gpt2.finetune(sess,
              dataset=grimms_clean,
              model_name='124M',
              steps=3000,
              restore_from='fresh',
              checkpoint_dir = "drive/My Drive/projet-TIA/gpt2/checkpoint/",
              run_name='run_grimms_french-3000',
              sample_every=100,
              save_every=500
              )

```

```

In [ ]:
gpt2.copy_checkpoint_to_gdrive(run_name='/content/drive/MyDrive/projet-TIA/gp

```

Générer du texte

```

In [ ]:
sess = gpt2.start_tf_sess()

```

```

In [ ]:
gpt2.reset_session(sess)
del sess
sess = gpt2.start_tf_sess()

```

```

In [ ]:
%%time
gpt2.load_gpt2(sess, run_name='/content/drive/MyDrive/projet-TIA/gpt2/checkpo

```

```

In [ ]:
gpt2.generate(sess,
              run_name='/content/drive/MyDrive/projet-TIA/gpt2/checkpoint/run_

```

```

length=50,
top_k=40,
top_p=1.0,
temperature=0.7,
prefix="Dans des temps très anciens, alors qu'il pouvait encore
)

```

Conte d'Andersen en français

```

In [ ]: andersen = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/anderse
In [ ]:
import re
def clean_text(text):
    pattern = '(page|Page|PAGE)(\s+|\s+)([0-9]+)(.*)$'
    cleaned = re.sub('\s$', '', text, flags=re.MULTILINE)
    p = re.compile(pattern, re.MULTILINE)
    cleaned = p.sub(" ", cleaned)
    return cleaned

In [ ]:
text=""
for book in andersen:
    with open(book) as f:
        print('Reading', book.split('/')[-1])
        temp = f.read()
        text = text + clean_text(temp)

In [ ]: with open("/content/drive/My Drive/Cleaned_andersen_french.txt", 'w') as f:
        f.writelines(text)

In [ ]: andersen_clean = "Cleaned_andersen_french.txt"

In [ ]: gpt2.copy_file_from_gdrive(andersen_clean)

```

Entrainement du modèle

```

In [ ]: sess = gpt2.start_tf_sess()

gpt2.finetune(sess,
              dataset=andersen_clean,
              model_name=model_name,
              steps=3000,
              restore_from='fresh',
              checkpoint_dir = "drive/My Drive/projet-TIA/gpt2/checkpoint/",
              run_name='run_andersen_french_3000',
              sample_every=100,
              save_every=500
            )

In [ ]: gpt2.copy_checkpoint_to_gdrive(run_name='/content/drive/MyDrive/projet-TIA/gp

```

Générer du texte

```
In [ ]: sess = gpt2.start_tf_sess()

In [ ]: gpt2.reset_session(sess)
        del sess
        sess = gpt2.start_tf_sess()

In [ ]: %%time
gpt2.load_gpt2(sess, run_name='/content/drive/MyDrive/projet-TIA/gpt2/checkpo
>Loading checkpoint /content/drive/MyDrive/projet-TIA/gpt2/checkpoint/run_ander
sen_french_3000/model-3000
INFO:tensorflow:Restoring parameters from /content/drive/MyDrive/projet-TIA/gp
t2/checkpoint/run_andersen_french_3000/model-3000
CPU times: user 2.61 s, sys: 846 ms, total: 3.46 s
Wall time: 6.63 s

In [ ]: gpt2.generate(sess,
                    run_name='/content/drive/MyDrive/projet-TIA/gpt2/checkpoint/run_
length=20,
top_k=40,
top_p=0.9,
temperature=1.0,
prefix="Il était une fois un prince qui voulait épouser une pri
)
```

Grimms en anglais

```
In [ ]: import glob

In [ ]: grimms_en = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/Grim
)

In [ ]: import re
def clean_text(text):
    pattern = '(page|Page|PAGE)(\s+|\s+)([0-9]+)(.*)$'
    cleaned = re.sub('\s$', '', text, flags=re.MULTILINE)
    p = re.compile(pattern, re.MULTILINE)
    cleaned = p.sub(" ", cleaned)
    return cleaned

In [ ]: text=""
for book in books:
    with open(book) as f:
        print('Reading', book.split('/')[-1])
        temp = f.read()
        text = text + clean_text(temp)

In [ ]: with open("/content/drive/My Drive/Cleaned_grimms_en.txt", 'w') as f:
        f.writelines(text)
```

```
In [ ]: grimms_en_clean = "Cleaned_grimms_en.txt"
```

```
In [ ]: gpt2.copy_file_from_gdrive(grimms_en_clean)
```

Entrainement du modèle

```
In [ ]: sess = gpt2.start_tf_sess()

gpt2.finetune(sess,
              dataset=grimms_en_clean,
              model_name='124M',
              steps=3000,
              restore_from='fresh',
              checkpoint_dir = "drive/My Drive/projet-TIA/gpt2/checkpoint/",
              run_name='run_grimms_en-3000',
              sample_every=100,
              save_every=500
              )
```

```
In [ ]: gpt2.copy_checkpoint_to_gdrive(run_name='/content/drive/MyDrive/projet-TIA/gp
```

Générer du texte

```
In [ ]: sess = gpt2.start_tf_sess()
```

```
In [ ]: gpt2.reset_session(sess)
del sess
sess = gpt2.start_tf_sess()
```

```
In [ ]: %%time
gpt2.load_gpt2(sess, run_name='/content/drive/MyDrive/projet-TIA/gpt2/checkpo
```

```
In [ ]: gpt2.generate(sess,
                  run_name='/content/drive/MyDrive/projet-TIA/gpt2/checkpoint/run_3000',
                  length=100,
                  top_k=40,
                  top_p=0.9,
                  temperature=1.0,
                  prefix="Once upon a time there was a princess with golden hair"
                  )
```

```
In [ ]:
```

Andersen en anglais

```
In [ ]: import glob
```

```
In [ ]: andersen_en = sorted(glob.glob('/content/drive/My Drive/projet-TIA/dataset/An
```

```
In [ ]: import re
def clean_text(text):
    pattern = '(page|Page|PAGE)(\s+|\s+)([0-9]+)(.*)$'
    cleaned = re.sub('\s$', '', text, flags=re.MULTILINE)
    p = re.compile(pattern, re.MULTILINE)
    cleaned = p.sub(" ", cleaned)
    return cleaned
```

```
In [ ]: text=""
for book in andersen_en:
    with open(book) as f:
        print('Reading', book.split('/')[-1])
        temp = f.read()
        text = text + clean_text(temp)
```

```
In [ ]: with open("/content/drive/My Drive/Cleaned_andersen_en.txt", 'w') as f:
    f.writelines(text)
```

```
In [ ]: andersen_en_clean = "Cleaned_andersen_en.txt"
```

```
In [ ]: gpt2.copy_file_from_gdrive(andersen_en_clean)
```

Entrainement du modèle

```
In [ ]: sess = gpt2.start_tf_sess()

gpt2.finetune(sess,
              dataset=grimms_en_clean,
              model_name='124M',
              steps=3000,
              restore_from='fresh',
              checkpoint_dir = "drive/My Drive/projet-TIA/gpt2/checkpoint/",
              run_name='run_andersen_en-3000',
              sample_every=100,
              save_every=500
              )
```

```
In [ ]: gpt2.copy_checkpoint_to_gdrive(run_name='/content/drive/MyDrive/projet-TIA/gp
```

Générer du texte

```
In [ ]: sess = gpt2.start_tf_sess()

gpt2.reset_session(sess)
del sess
```

```
sess = gpt2.start_tf_sess()

In [ ]:
%%time
gpt2.load_gpt2(sess, run_name='/content/drive/MyDrive/projet-TIA/gpt2/checkpo

In [ ]:
gpt2.generate(sess,
              run_name='/content/drive/MyDrive/projet-TIA/gpt2/checkpoint/run_1',
              length=100,
              top_k=40,
              top_p=0.9,
              temperature=1.0,
              prefix="Once upon a time there was a princess with golden hair"
            )
```

9.5 Modèle fr-boris (Cedille.ai)

Inférence fr-boris (Cedille.ai)

Ce notebook montre comment exécuter le [modèle GPT-J-6B](#). Voir le lien pour plus de détails sur le modèle, y compris les mesures d'évaluation et les crédits.

Installer les dépendances

```
In [ ]: apt install gzip
apt install zstd

In [ ]: # Récuperer les poids du modèles fr-boris
time wget "https://storage.googleapis.com/cedille-public/models/boris/step_15"
time tar -zxf step_150000.tar.gz

curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
sudo apt-get install git-lfs
git lfs install

# Cloner le repository kingoflolz/mesh-transformer-jax
git clone https://github.com/kingoflolz/mesh-transformer-jax.git

# Installer les dépendances
pip install -r mesh-transformer-jax/requirements.txt
pip install mesh-transformer-jax/ jax==0.2.12 tensorflow==2.5.0
```

Setup Model

```
In [ ]: import os
import requests
from jax.config import config

colab_tpu_addr = os.environ['COLAB_TPU_ADDR'].split(':')[0]
url = f'http://{colab_tpu_addr}:8475/requestversion/tpu_driver0.1_dev20210607'
requests.post(url)

# The following is required to use TPU Driver as JAX's backend.
config.FLAGS.jax_xla_backend = "tpu_driver"
config.FLAGS.jax_backend_target = "grpc://" + os.environ['COLAB_TPU_ADDR']

In [ ]: import time
import glob
import re

import jax
from jax.experimental import maps
import numpy as np
import optax
import transformers

from transformers import AutoModelForCausalLM, AutoTokenizer, top_k_top_p_filter
from mesh_transformer.checkpoint import read_ckpt_lowmem
from mesh_transformer.sampling import nucleus_sample
from mesh_transformer.transformer_shard import CausalTransformer
```

```
In [ ]:
params = {
    "layers": 28,
    "d_model": 4096,
    "n_heads": 16,
    "n_vocab": 50400,
    "norm": "layernorm",
    "pe": "rotary",
    "pe_rotary_dims": 64,
}

"seq": 2048,
"cores_per_replica": 8,
"per_replica_batch": 1,
}

per_replica_batch = params["per_replica_batch"]
cores_per_replica = params["cores_per_replica"]
seq = params["seq"]

params["sampler"] = nucleaus_sample

# ici, nous "supprimons" les paramètres de l'optimiseur du modèle (car nous n
params["optimizer"] = optax.scale(0)

mesh_shape = (jax.device_count() // cores_per_replica, cores_per_replica)
devices = np.array(jax.devices()).reshape(mesh_shape)

maps.thread_resources.env = maps.ResourceEnv(maps.Mesh(devices, ('dp', 'mp')))

tokenizer = AutoTokenizer.from_pretrained("Cedille/fr-boris")
```

Création du réseau et chargement des paramètres (prend environ 5 minutes)

```
In [ ]:
total_batch = per_replica_batch * jax.device_count() // cores_per_replica

network = CausalTransformer(params)

network.state = read_ckpt_lowmem(network.state, "/content/step_150000/", devic
network.state = network.move_xmap(network.state, np.zeros(cores_per_replica))
```

```
In [ ]: !python to_hf_weights.py --help
```

Execution du modèle

Le premier échantillon prend environ une minute en raison de la compilation.

Jouer avec les différents paramètres d'échantillonnage (top_p et temp), ainsi que la longueur des générations (gen_len, provoque une recompilation lorsqu'il est modifié).

On peut modifier d'autres éléments tels que per_replica_batch dans les cellules précédentes pour modifier le nombre de générations effectuées en parallèle. Un lot plus important a une latence plus élevée mais un débit plus élevé lorsqu'il est mesuré en jetons générés

9.6 Evaluation Andersen

Évaluation des contes d'Andersen en français et en anglais :

L'évaluation ci-dessous porte sur les modèles qui ont été entraînés sur le jeu de données des contes d'Andersen en anglais et en français.

J'ai demandé aux enfants d'inventer un début d'histoire que nous avons ensuite envoyé au modèle afin qu'il génère la suite.

Nous avons générée des histoires en faisant varier la langue, les paramètres et la longueur. Ensuite, j'ai fait lire les histoires générées à chaque enfant afin qu'il évalue le texte en prenant en compte :

- Le contexte de l'histoire dans sa globalité
- La qualité des phrases générées (présence de ponctuation, orthographe, dialogue, ...)

La diversité de l'âge des enfants et leur compréhension de lectures et de langues étant différente, les notes l'ont été également.

Voici les histoires générées avec leurs paramètres et les notes qui vont de 1 à 10 attribué par chaque enfant :

Pour la première histoire en français, nous avons défini les paramètres suivants :

- **Top_p = 0.7**
- **Température = 0.8**
- **Longueur de texte = 40**
- **Préfixe** = Il était une fois une jolie licorne du nom de Mathilde qui avait un mignon petit frère. Il était gentil et elle l'adorait. Mais aujourd'hui, Mathilde aurait bien aimé avoir sa maman pour elle toute seule car c'était son anniversaire.

Voici l'histoire générée :

Il était une fois une jolie licorne du nom de Mathilde qui avait un mignon petit frère. Il était gentil et elle l'adorait. Mais aujourd'hui, Mathilde aurait bien aimé avoir sa maman pour elle toute seule car c'était son anniversaire. A la manche lui, tâche de fins, tâche de bons sur ces petites forces, et pour la plus jeune serviteur, à l'inst

Voici les notes des enfants et leurs appréciations :

- **Hugo 5 ans** = 0/10 « C'est nul »
- **Chaden 6 ans** = 1/10 « Il y a des fautes de français « **la plus jeune serviteur.** » »
- **Mathilde 8 ans** = 1/10 "ça n'a aucun sens"
- **Quentin 10 ans** = 0/10 "...."
- **Théo 12 ans** = 1/10 "Sans commentaires"

J'ai additionné toutes les notes et effectuer la moyenne : **3/5 = 0.6**

Mon modèle obtient une moyenne de **0,6 sur 10**

Nous avons généré une seconde histoire en anglais en changeant les paramètres, nous avons gardé le même début d'histoire que nous avons traduit en anglais :

Les enfants ne parlent pas anglais, nous avons demandé à un adulte bilingue, de nous traduire le texte.

- **Top_p = 0.9**
- **Température = 1.0**
- **Longueur de texte = 100**
- **Préfixe = Once upon a time there was a pretty unicorn named Mathilde who had a cute little brother. He was nice and she loved him. But today, Mathilde would have liked to have her mother all to herself because it was her birthday.**

Voici l'histoire générée :

Once upon a time there was a pretty unicorn named Mathilde who had a cute little brother. He was nice and she loved him. But today, Mathilde would have liked to have her mother all to herself because it was her birthday. They had set out with their gold and silver on the greatest journey that they had ever been on, and now that they had everything they were looking for. "Now we are together again," said the shoemaker; and, without speaking a word, he spread out his handkerchiefs on the ground, and there they stood them, looking for any one who might help them. He went up to them and said, "Well, how have you got all this far?" - "Oh,

Voici les notes des enfants et leurs appréciations pour la seconde histoire :

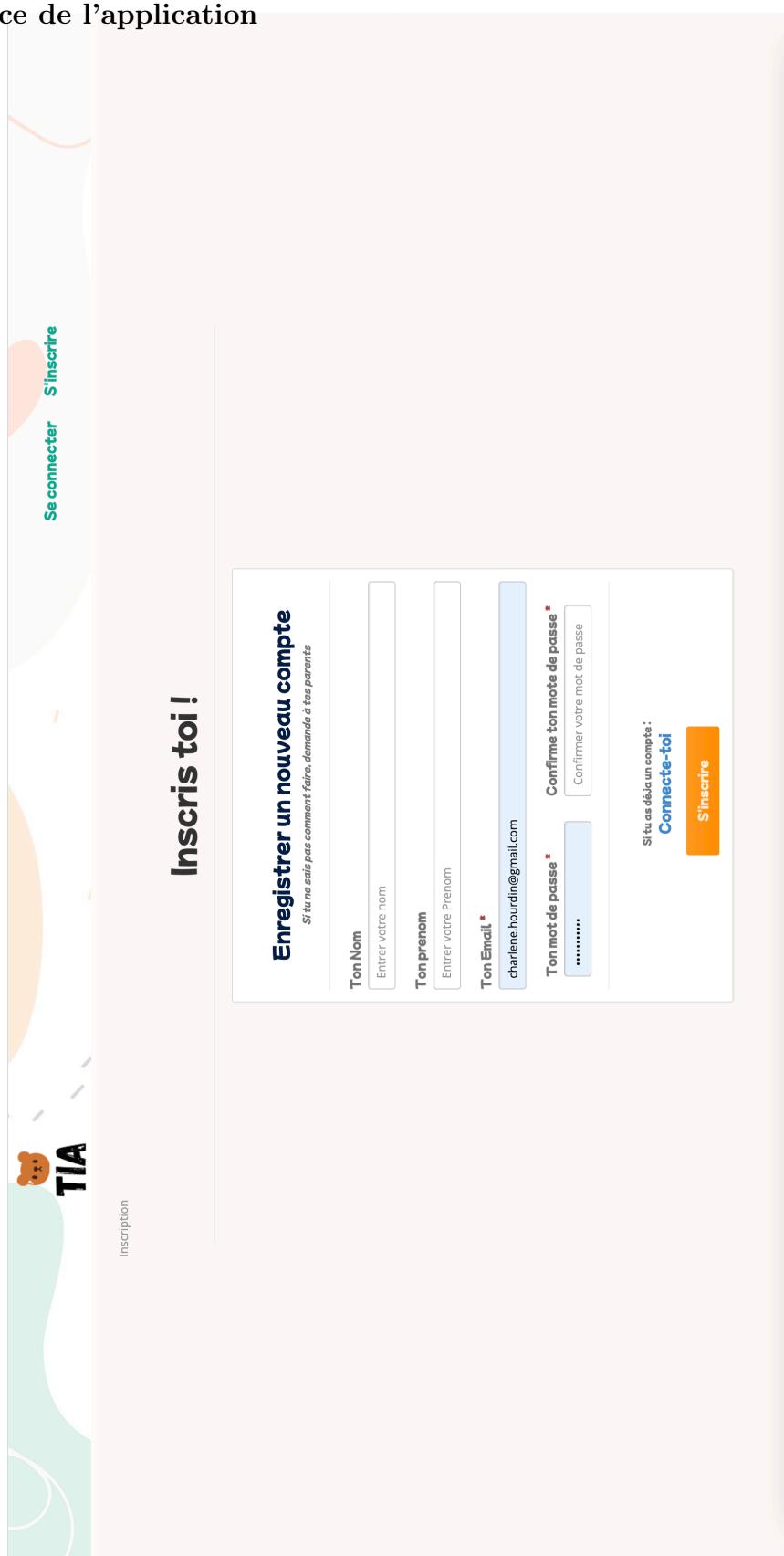
- **Hugo 5 ans = 6/10** « Il a écrit une autre histoire, mais elle est bien »
- **Chaden 6 ans = 5/10** « la suite ne correspond pas au début ».
- **Mathilde 8 ans = 6/10** « L'histoire qu'il a créé est mieux mais ça n'a rien à voir avec l'histoire que nous avons créer. »
- **Quentin 10 ans = 7/10** « les phrases sont bien écrites »
- **Théo 12 ans = 4/10** « pas de sens avec le début de l'histoire par contre il y a de vraie dialogue »

J'ai additionné toutes les notes effectuées la moyenne : **28/5 = 5,6**

Mon modèle obtient une moyenne de **5,6 sur 10**.

En conclusion : La conclusion est identique à celle présenté en page 16 du rapport.

9.7 Interface de l'application



The screenshot shows the TIA website's login page. The background features a light blue gradient with white wavy lines and a small orange bear icon. On the left, there is a vertical sidebar with the word "TIA" in large letters and a smaller "Se connecter" link. The main content area has a light gray background with a central white rectangular form.

Connecte-toi !

Toc toc toc qui est là ?

S'il t'a pas comment faire, demande à tes parents

Rentrez ton adresse E-mail*
charlene.houdini@gmail.com

Rentrez ton mot de passe*
.....

Si tu n'as pas de compte demande à tes parents de t'en créer un !

Inscris-toi

Connecte-toi

Suivez-moi

Contact

06 83 71 45 23
charlene.houdini@gmail.com

A propos

En reconversion professionnelle ...
....

in

The screenshot shows the TIA website interface. On the left, there's a sidebar with a green background featuring a cartoon bear icon, the text "Vous êtes connecté avec succès !", and navigation links: "Les générateurs", "Mon compte", and "Se déconnecter". The main content area has a white background with a large orange oval and abstract shapes. The title "TIA" is prominently displayed, followed by the subtitle "Une IA qui te raconte des histoires". Below this are two buttons: "CRÉER TES HISTOIRES EN FRANÇAIS" (in green) and "CRÉER TES HISTOIRES EN ANGLAIS" (in orange). To the right, a detailed view of a generated story titled "TIA, un générateur d'histoire" is shown. The story text is in French and discusses Natural Language Generation (NLG). At the bottom of the page, there are sections for "Contact", "Suivez-moi", "À propos", and a footer with links like "Actualité", "Générateur français", "Générateur anglais", "Contact", and "Mon compte".

TIA, un générateur d'histoire

TIA est un modèle NLG (Natural language generation) qui génère des histoires en anglais ou en français en fonction du thème choisi

Le NLG, qu'est-ce que c'est ?

La génération automatique de textes (NLG, pour Natural language generation) est un domaine de l'intelligence artificielle (IA) qui vise à produire du contenu (ici du texte) ou un discours comparable à celui des humains à partir d'un ensemble de données. La génération de texte est une tâche complexe puisque l'on cherche à produire des textes qui se rapprochent le plus possible du contenu d'une source donnée, sans nécessairement être cohérent avec celle de la source. La génération automatique de texte présente des difficultés traditionnelles dans la production de textes. Les mots sont souvent mal choisis et le style général peut être tronqué ou erroné (par exemple, en utilisant un style familier au lieu d'un style académique, ou en utilisant des mots qui ne sont pas appropriés, etc.

Contact
charline.bourdin@gmail.com
06 83 71 45 23

Suivez-moi
in

À propos
En reconversion professionnelle...
....

Création © 2021. Houelle Chaudie. Design by genmotive

The screenshot displays the TIA website's interface, which includes a navigation bar with "Accueil", "Les générateurs", "Mon compte", and "Se déconnecter". Below the navigation is a large orange "TIA" logo.

Top_P (entre 0 et 1): (1)

Seuil qui filtre la valeur de probabilité du prochain mot :
Bas = garde les mots les plus probable et éteint les mot improbable.
Haut = plus exotique
(Obtient de bons résultats avec top_p=0.9)

Temperature (entre 0 et 1): (1)

Uniformise la probabilité du prochain mot :
Bas = plus déterministe.
Haut = plus créatif (ou plus aléatoire)
(Recommendé de garder entre 0.1 et 1.0)

Veuillez définir la longueur de votre texte: 50

Generer

Contact
06 83 71 45 23
charlene.hourdin@gmail.com

Suivez-moi
in C

A propos
En reconversion professionnelle....
....

Copyright © 2021, Hourdin Charlène. Designed by gettemplate

Accueil | Générateur français | Générateur anglais | Contact | Mon compte

The screenshot shows the TIA story generator interface with the following configuration settings:

- Top_P (0 - 1):** A slider set at 0.7. A tooltip explains: "Threshold that filters the probability value of the next word: low = keep the most probable words and avoid the improbable words; high = more exotic words" (Outputs good results on a dataset with top_p = 0.9).
- Temperature (0 - 1):** A slider set at 0.7. A tooltip explains: "Standardizes the probability of the next word: low = more deterministic; high = more creative or more random" (Recommended to keep between 0.7 and 1.0).
- Please define the length of your text:** A slider set at 50.

At the bottom right, there is a "Generator" button.

Suivez-moi
in

Contact
06 83 71 45 23
charline.hourdin@gmail.com

A propos
En conversion professionnelle....
....

Copyright © 2021, Hourdin Charlène. Designed by getemplate

[Accueil](#) | [Générateur français](#) | [Générateur anglais](#) | [Contact](#) | [Mon compte](#)

 TIA
[Accueil](#)
[Les générateurs](#) •
[Mon compte](#)
[Se déconnecter](#)

[Accueil](#) / Mon compte
[Mon compte](#)

Mon compte

ID	Histoire	Supprimer
356	Il était une fois le dissasne au bord d'un marl des pommes rouges et le dissasne se mit à filer, mais pour le craqueur s'en retourna à cheval et demanda : -	Supprimer
357	Ils saisirent tous deux et dirent: "Que vas-tu devenir pousser les connaissances ma venir un chaudron devant le premier ou de fils ouvert!"	Supprimer
358	Il était une fois une jolie licorne du nom de Mathilde qui avait un mignon petit frère. Il était gentil et elle l'adorait. Mais aujourd'hui, Mathilde aurait bien aimé avoir sa maman pour elle toute seule car c'était son anniversaire, ul ôté dans la cave annoncée, et qu'il fut améné pour elle durant le tour, chacun fut d'accepter leur livre. «J'en suis	Supprimer
359	Il était une fois une jolie licorne du nom de Mathilde qui avait un mignon petit frère. Il était gentil et elle l'adorait. Mais aujourd'hui, Mathilde aurait bien aimé avoir sa maman pour elle toute seule car c'était son anniversaire. Alors maître petite. La petite robe se trouva en courant d'un meutrier. Quelle a porté sous à genêtrier	Supprimer
360	Il était une fois une jolie licorne du nom de Mathilde qui avait un mignon petit frère. Il était gentil et elle l'adorait. Mais aujourd'hui, Mathilde aurait bien aimé avoir sa maman pour elle toute seule car c'était son anniversaire. A la manche lui tâche de fins, tâche de bons sur ces petites forces, et pour la plus jeune serviteur, à l'instant.	Supprimer
361	Once upon a time there was a pretty unicorn named Mathilde who had a cute little brother. He was nice and he loved him. But today, Mathilde would have liked to have her mother all to herself because it was her birthday. They had set out with their gold and silver on the greatest journey that they had ever been on, and now that they had everything they were looking for. "Now we are together again," said the shoemaker, and, without speaking a word, he spread out his handkerchiefs on the ground, and there they stood them, looking for any one who might help them. He went up to them and said, "Well, how have you got all this far?" "Oh,	Supprimer

[Contact](#)
[Suivez-moi](#)
[A propos](#)

[in](#) 

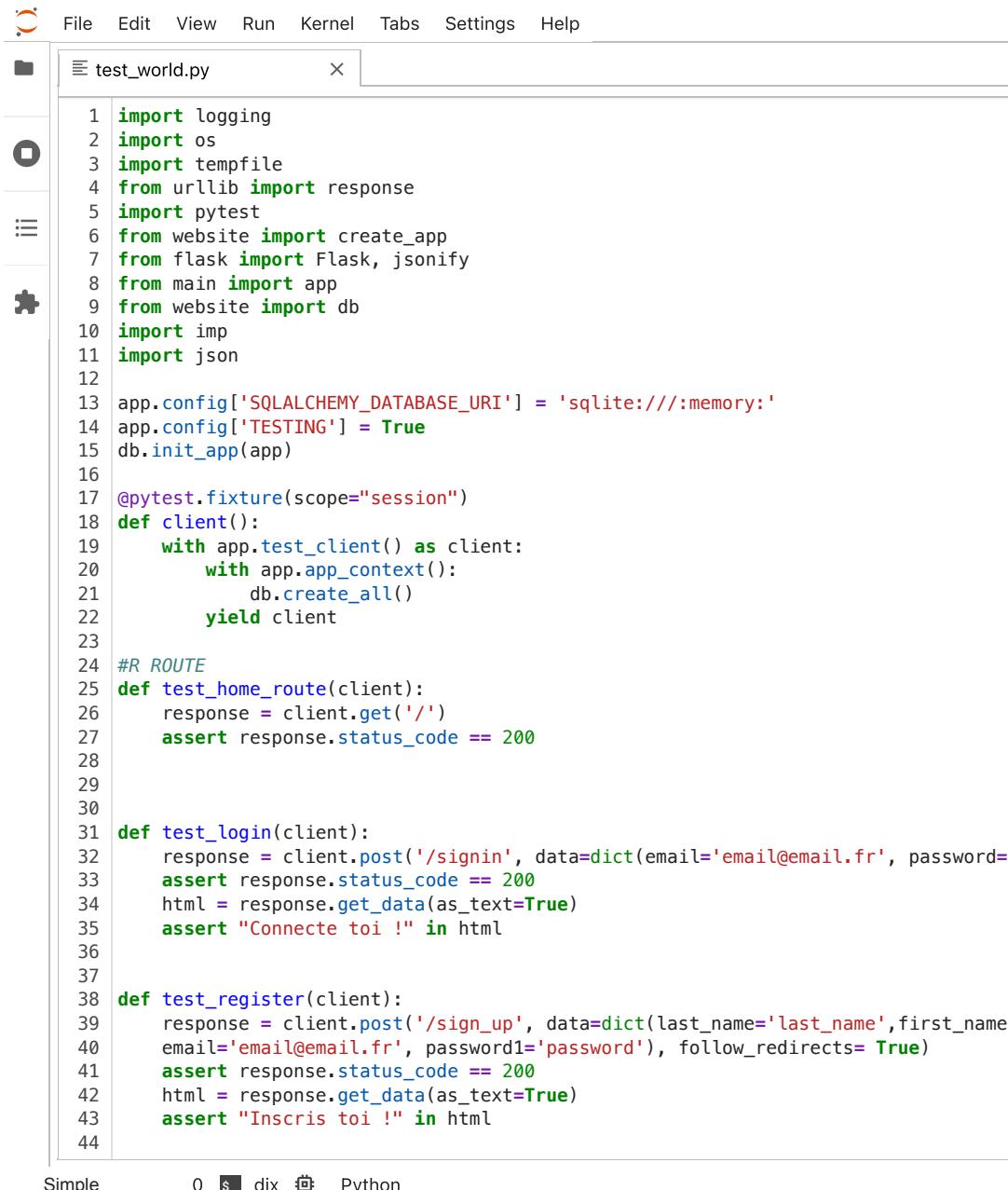
En réconversion professionnelle....

...

[Accueil](#) | Générateur français | Générateur anglais | Contact | Mon compte

Copyright © 2021, Houdin Charlène. Designed by genemplate.

9.8 Test Unitaire



```

File Edit View Run Kernel Tabs Settings Help
└── test_world.py
1 import logging
2 import os
3 import tempfile
4 from urllib import response
5 import pytest
6 from website import create_app
7 from flask import Flask, jsonify
8 from main import app
9 from website import db
10 import imp
11 import json
12
13 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///:memory:'
14 app.config['TESTING'] = True
15 db.init_app(app)
16
17 @pytest.fixture(scope="session")
18 def client():
19     with app.test_client() as client:
20         with app.app_context():
21             db.create_all()
22     yield client
23
24 #R ROUTE
25 def test_home_route(client):
26     response = client.get('/')
27     assert response.status_code == 200
28
29
30 def test_login(client):
31     response = client.post('/signin', data=dict(email='email@email.fr', password=''))
32     assert response.status_code == 200
33     html = response.get_data(as_text=True)
34     assert "Connecte toi !" in html
35
36
37 def test_register(client):
38     response = client.post('/sign_up', data=dict(last_name='last_name', first_name='',
39 email='email@email.fr', password1='password'), follow_redirects=True)
40     assert response.status_code == 200
41     html = response.get_data(as_text=True)
42     assert "Inscrис toi !" in html
43
44

```

Simple

0 \$ dix Python

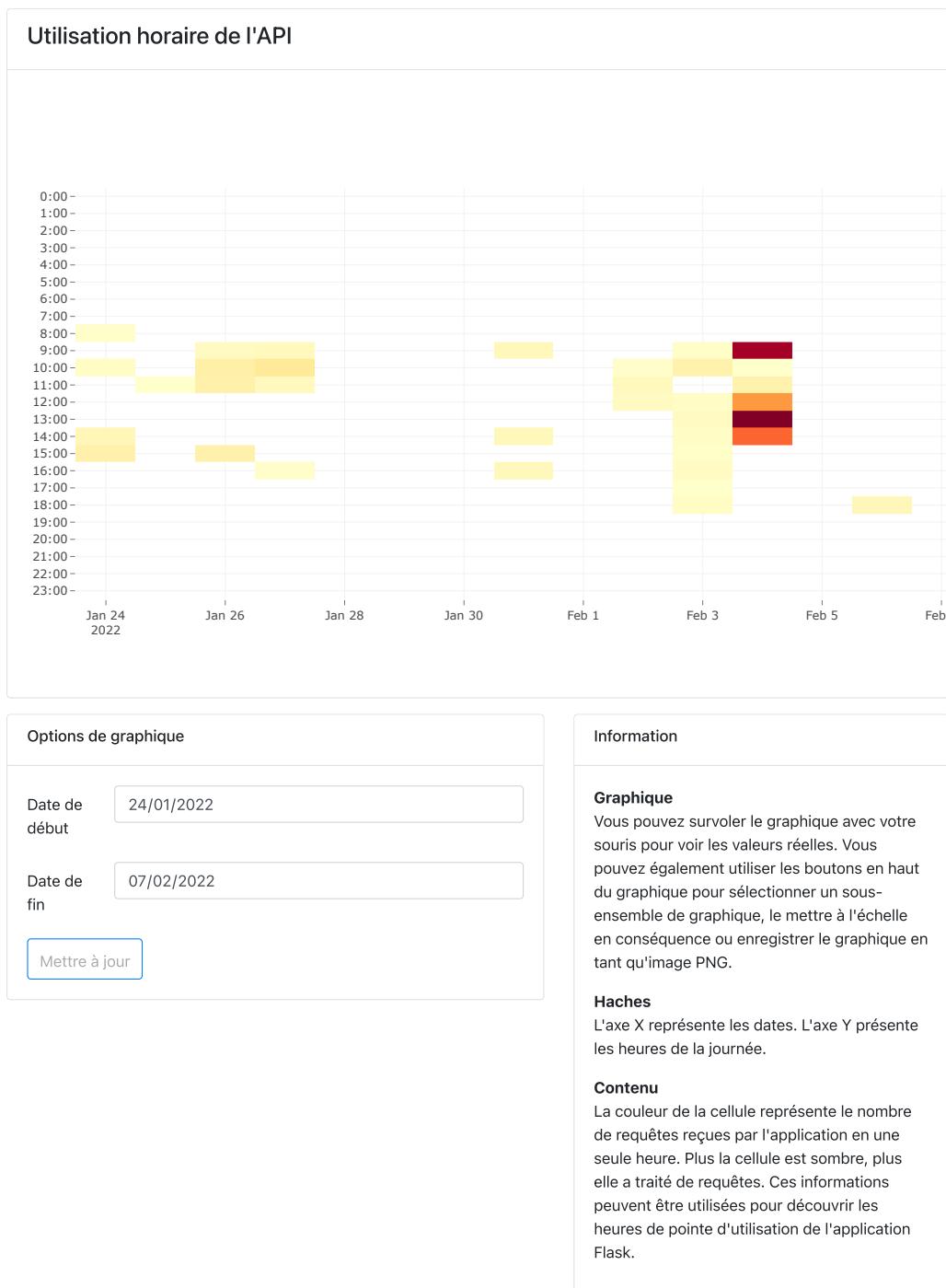
The screenshot shows a Jupyter Notebook interface with a single code cell containing a Python test script. The notebook has a toolbar at the top with icons for File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The left sidebar shows a file tree with 'test_world.py' selected. The main area contains the following code:

```
45
46 def test_generate_french(client):
47     response = client.get('/generate_grimms_french', follow_redirects=True)
48     assert response.status_code == 200
49     html = response.get_data(as_text=True)
50     assert "Générateur en Français" in html
51
52 def test_generate_english(client):
53     response = client.get('/generate_grimms_english', follow_redirects=True)
54     assert response.status_code == 200
55     html = response.get_data(as_text=True)
56     assert "Generator in English" in html
57
58
59 def test_account(client):
60     response = client.get('/account')
61     assert response.status_code == 200
62     html = response.get_data(as_text=True)
63     assert "Mon compte" in html
64
```

At the bottom of the notebook, there are buttons for Simple, 0, \$, dix, and Python.

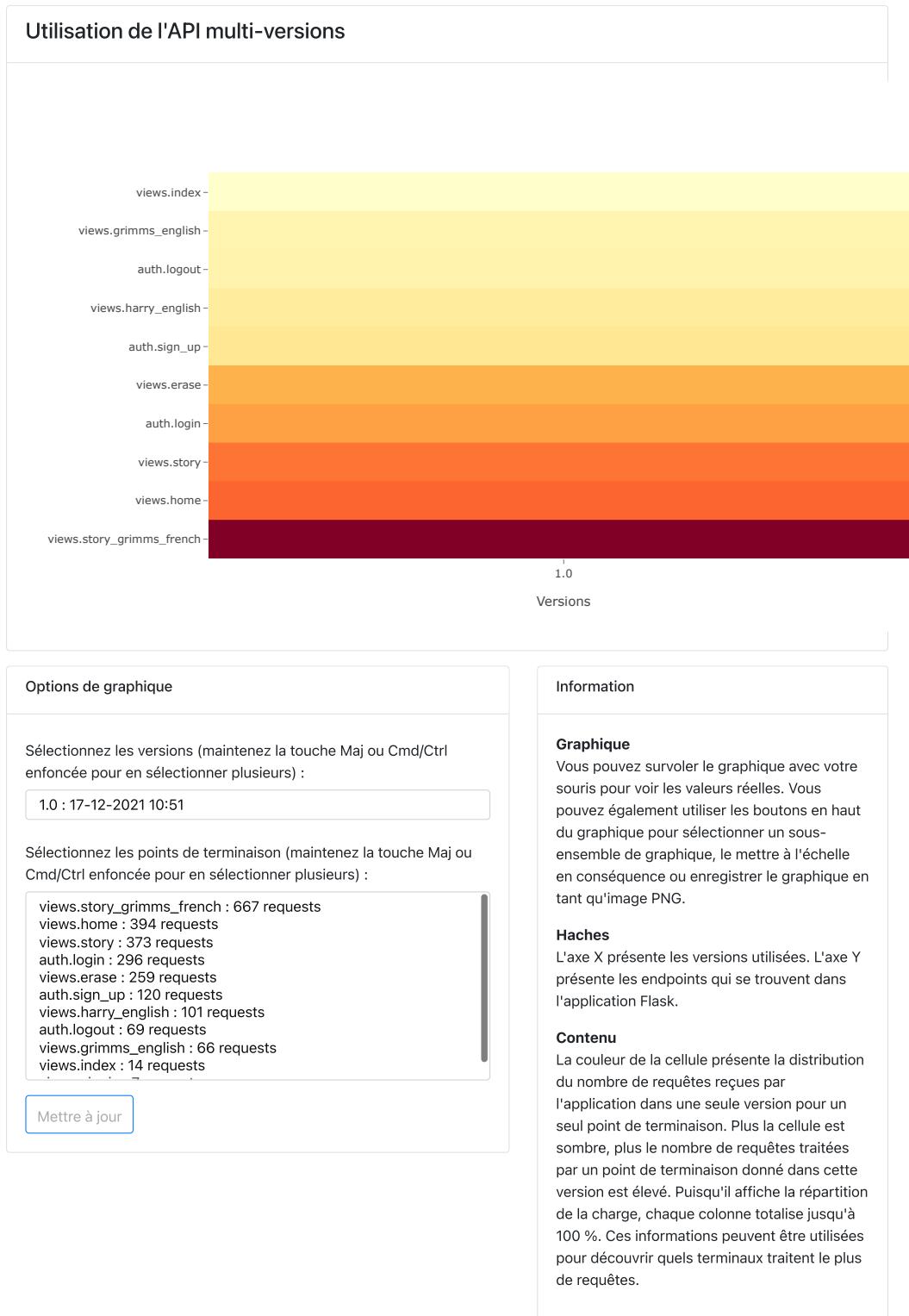
9.9 Monitoring

9.9.1 Utilisation horaire de l'API

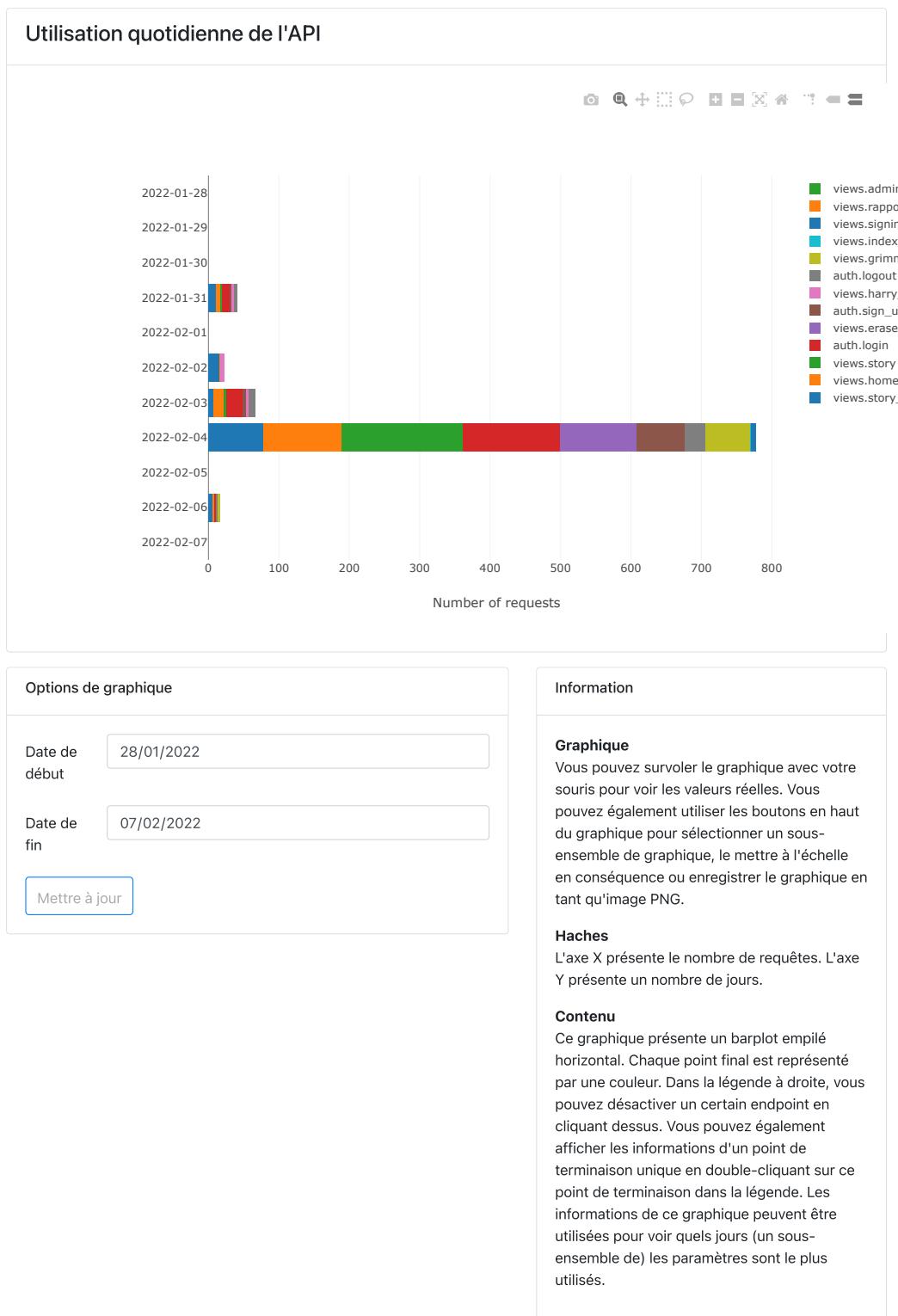


Cliquez [ici](#) pour lire la documentation avancée.

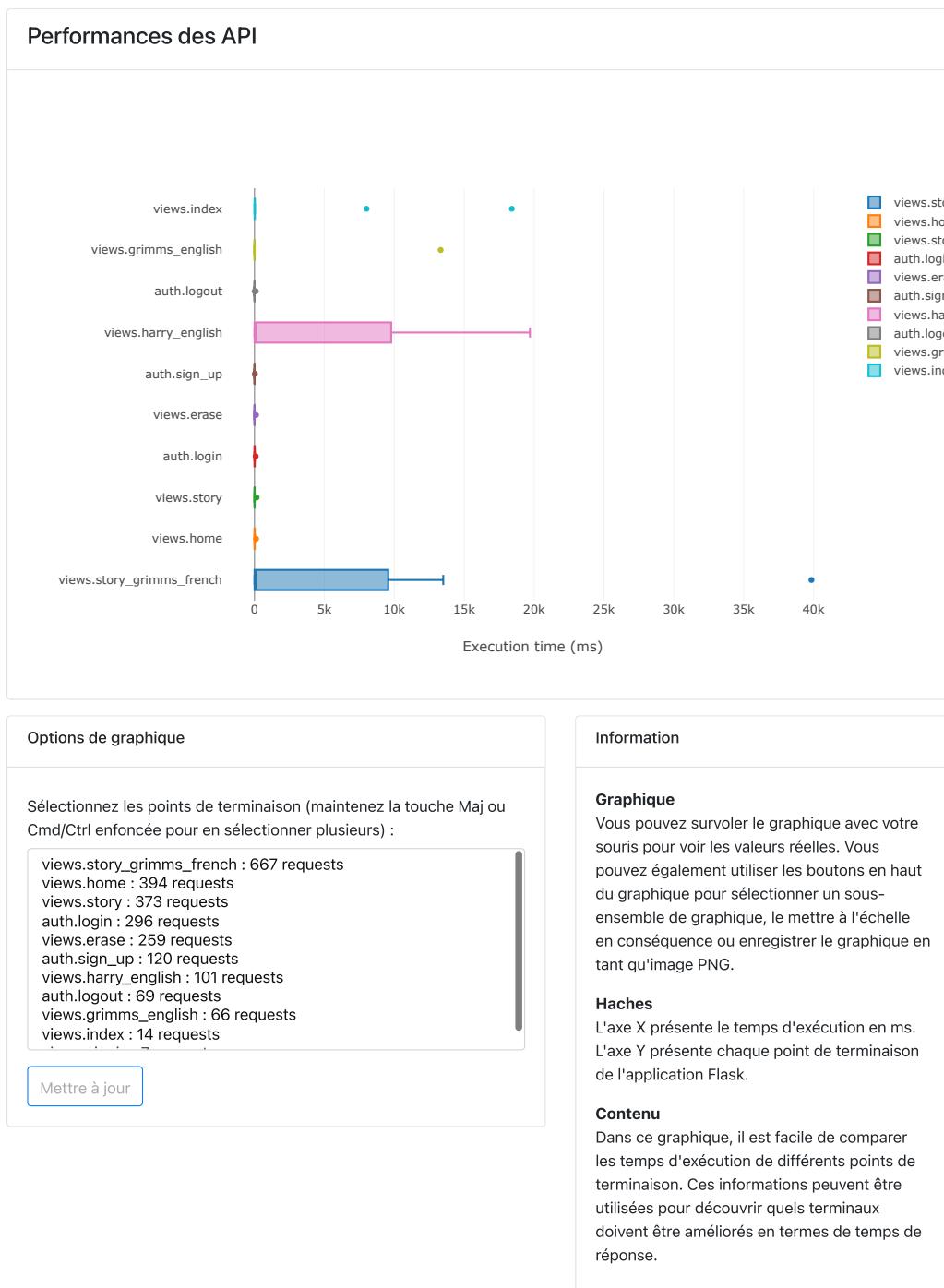
9.9.2 Utilisation de l'API multi-versions



9.9.3 Utilisation quotidienne de l'API



9.9.4 Performances des API



Cliquez [ici](#) pour lire la documentation avancée.

9.9.5 Rapport

Rapports

Générez un rapport pour obtenir plus d'informations sur l'évolution des performances de votre service.

07 février contre 06 février Semaine 6 vs Semaine 5 février vs janvier Comparer les commits Personnalisé

01/02/2022 00:00	28/02/2022 23:59:59,999
------------------	-------------------------

Par rapport à

01/01/2022 00:00	31/01/2022 23:59:59,999
------------------	-------------------------

Afficher uniquement les résultats significatifs

vues.story_grimms_french

La latence médiane a diminué de 35,9 %
De 20,3 ms à 13,0 ms

vues.home

La latence médiane a augmenté de 356,3 %
De 2,3 ms à 10,7 ms

vues.histoire

La latence médiane a augmenté de 53,8 %
De 4,8 ms à 7,4 ms

auth.login

La latence médiane a augmenté de 1 070,5 %
De 1,2 ms à 14,0 ms

Distribution du code d'état modifiée

9.9.6 Rapport

Code d'état	
200	79.4% -> 86.7%
302	20.6% -> 13.3%

views.erase

Median latency increased by 34.1%
From 2.8ms to 3.8ms

[Details](#)

auth.sign_up

Median latency increased by 240.8%
From 0.8ms to 2.7ms

[Details](#)

Distribution of status code changed

Status code	
200	100.0% -> 47.3%
500	0.0% -> 41.9%
302	0.0% -> 10.8%

auth.logout

Median latency decreased by 13.0%
From 2.0ms to 1.7ms

[Details](#)

Cliquez [ici](#) pour lire la documentation avancée.