\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Plateforme de Résumé Automatique d'Articles d'Al Jazeera**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Mots clés :** BART, PEGASUS, T5, Artificial Intelligence, Text Summarisation

Réalisé par : Encadré par:

**Ibtissam LABYADY Najima DAOUDI**

**Sokhna Mai WANE Ghizlane BOURAHOUAT**

**Mohamed CISSE**

# **Table des matière**

Contents

[**Table des matière** 2](#_Toc154867104)

[Tables des figures 3](#_Toc154867105)

[**Introduction** 4](#_Toc154867106)

[**Conception de notre plateforme** 5](#_Toc154867107)

[**Implementation : visite de notre plateforme** 6](#_Toc154867108)

[**Web Scraping Aljazeera** 7](#_Toc154867109)

[**Modèle BART : Bidirectional + Auto-Regressif** 9](#_Toc154867110)

[**A.** **Définition :** 9](#_Toc154867111)

[**B.** **Architecture BART :** 10](#_Toc154867112)

[**C.** **Evaluation du modèle BART :** 12](#_Toc154867113)

[**Pegasus** 13](#_Toc154867114)

[**1.** **Compréhension de base de PEGASUS** 14](#_Toc154867115)

[**2.** **Architecture de PEGASUS** 14](#_Toc154867116)

[**3.** **Entraînement et Fine-tuning** 15](#_Toc154867117)

[**4.** **Application de Pegasus pour résumer les articles de Al Jazeera** 16](#_Toc154867118)

[**T5: Text-To-Text Transfer Transformer** 18](#_Toc154867119)

[1-Généraliés sur le modèle T5 18](#_Toc154867120)

[2-Implémentation du code 20](#_Toc154867121)

[3- Evaluation du code 21](#_Toc154867122)

[Le modèle le plus performant (Evaluation) 22](#_Toc154867123)

[**Fine Tuning VS Traduction** 23](#_Toc154867124)

[**1-Problématique :** 23](#_Toc154867125)

[Quelle approche est la plus efficace pour générer des résumés dans une langue cible: utiliser la traduction avec un modèle pré-entraîné ou effectuer un fine-tuning sur des données dans la langue cible ? 23](#_Toc154867126)

[2-Traduction 23](#_Toc154867127)

[**3-Fine Tuning du modele T5** 24](#_Toc154867128)

[**4-Implementation du code** 25](#_Toc154867129)

[**5- Résultats et Discussion :** 28](#_Toc154867130)

[Conclusion 30](#_Toc154867131)

[Référence 31](#_Toc154867132)

# Tables des figures

[Figure 1 T5 :Text-to-Text Transfer Transformer 19](#_Toc154867050)

[Figure 2 Traduction du text 24](#_Toc154867051)

[Figure 3 dataset Aljazeera 26](#_Toc154867052)

[Figure 4 Tokenization 27](#_Toc154867053)

[Figure 5 fine tuning T5 Base scores 29](#_Toc154867054)

# **Introduction**

Définition :

La summarization de texte (résumé automatique de texte) est le processus de distillation des informations les plus importantes d'un texte afin de produire une version abrégée pour une tâche spécifique et un utilisateur donné.

Un résumé automatique de texte est une version condensée d'un document textuel, obtenu au moyen de techniques informatiques. La forme la plus connue et la plus visible des condensés de textes est le résumé, représentation abrégée et exacte du contenu d'un document.

Cependant, produire un résumé pertinent et de qualité demande au résumeur (un humain ou un système automatique) l'effort de sélectionner, d'évaluer, d'organiser et d'assembler des segments d'information selon leur pertinence. Bien comprendre et gérer les phénomènes de redondance, cohérence et cohésion est fondamental afin de produire des résumés automatiques humainement crédibles.

Il existe deux principaux types de text summarization, à savoir l'extraction et la génération.

**1. Extraction de Résumé :**

- Principe : L'extraction de résumé implique la sélection des phrases ou parties les plus importantes d'un texte existant pour créer un résumé concis.

- Méthode : Les algorithmes d'extraction analysent le texte source et identifient les phrases clés en se basant sur des critères tels que la fréquence des mots, l'importance des phrases, ou d'autres caractéristiques.

- Avantages : C'est une approche plus simple et plus rapide, souvent utilisée pour extraire l'essentiel d'un document sans générer de nouveaux mots ou concepts.

- Inconvénients : Elle peut ne pas être aussi créative ou flexible que la génération, et il est possible de manquer des informations non explicitement présentes dans le texte source.

**2. Génération de Résumé :**

- Principe : La génération de résumé créé un résumé entièrement nouveau en utilisant ses propres mots, phrases et structures de phrases, souvent de manière plus abstraite ou réorganisée par rapport au texte source.

- Méthode : Les modèles de génération, tels que BART, utilisent des approches basées sur des transformers et des techniques d'apprentissage profond pour comprendre le contenu global et générer un résumé complet.

- Avantages : Elle peut être plus expressive et flexible, générant des résumés uniques et bien formulés, parfois même en reformulant le contenu original.

- Inconvénients : Elle peut nécessiter plus de ressources computationnelles, de données d'entraînement et peut être plus complexe à mettre en œuvre.

L'extraction se concentre sur la sélection des parties importantes du texte source, tandis que la génération crée un résumé original en utilisant des modèles linguistiques avancés. Le choix entre les deux dépend des besoins spécifiques de la tâche et des préférences en matière de qualité et de créativité du résumé généré.

La text summarization présente cinq avantages majeurs :

**1. Economie de temps :**

**2. Accès rapide à l'information clé** : Elle facilite une compréhension rapide en extrayant les points principaux, ce qui s'avère particulièrement utile dans des situations où le temps est limité.

**3. Traitement efficace des grandes quantités de données :** Pour les volumes importants de texte, la summarization offre un moyen efficace d'extraire les idées principales sans nécessiter une analyse manuelle détaillée.

**4. Aide à la prise de décision :** En fournissant une vue d'ensemble succincte, elle assiste les individus dans la prise de décisions éclairées en évitant d'être submergés par des détails inutiles.

**5. Automatisation des tâches :** Les techniques automatisées de text summarization permettent de traiter rapidement et efficacement de vastes ensembles de données, ce qui peut être complexe et chronophage avec des approches manuelles.

# **Conception de notre plateforme**

La conception suit les étapes suivantes :

1. Collecte des données :

- Utilisation du web scraping pour extraire les catégories et liens d'un site (Al Jazeera).

- Extraction d'articles à partir des liens obtenus.

2. Prétraitement des données :

- Analyse statistique et exploration des données.

- Utilisation de TF-IDF pour identifier les mots clés les plus pertinents.

3. Création et évaluation des modèles :

- Utilisation du modèle BART pour générer des résumés de texte.

- Évaluation des performances du modèle en utilisant la similarité ROUGE entre les résumés générés et les résumés réels.

4. Amélioration des performances :

- Analyse du sentiment et reconnaissance des parties du discours pour enrichir l'analyse.

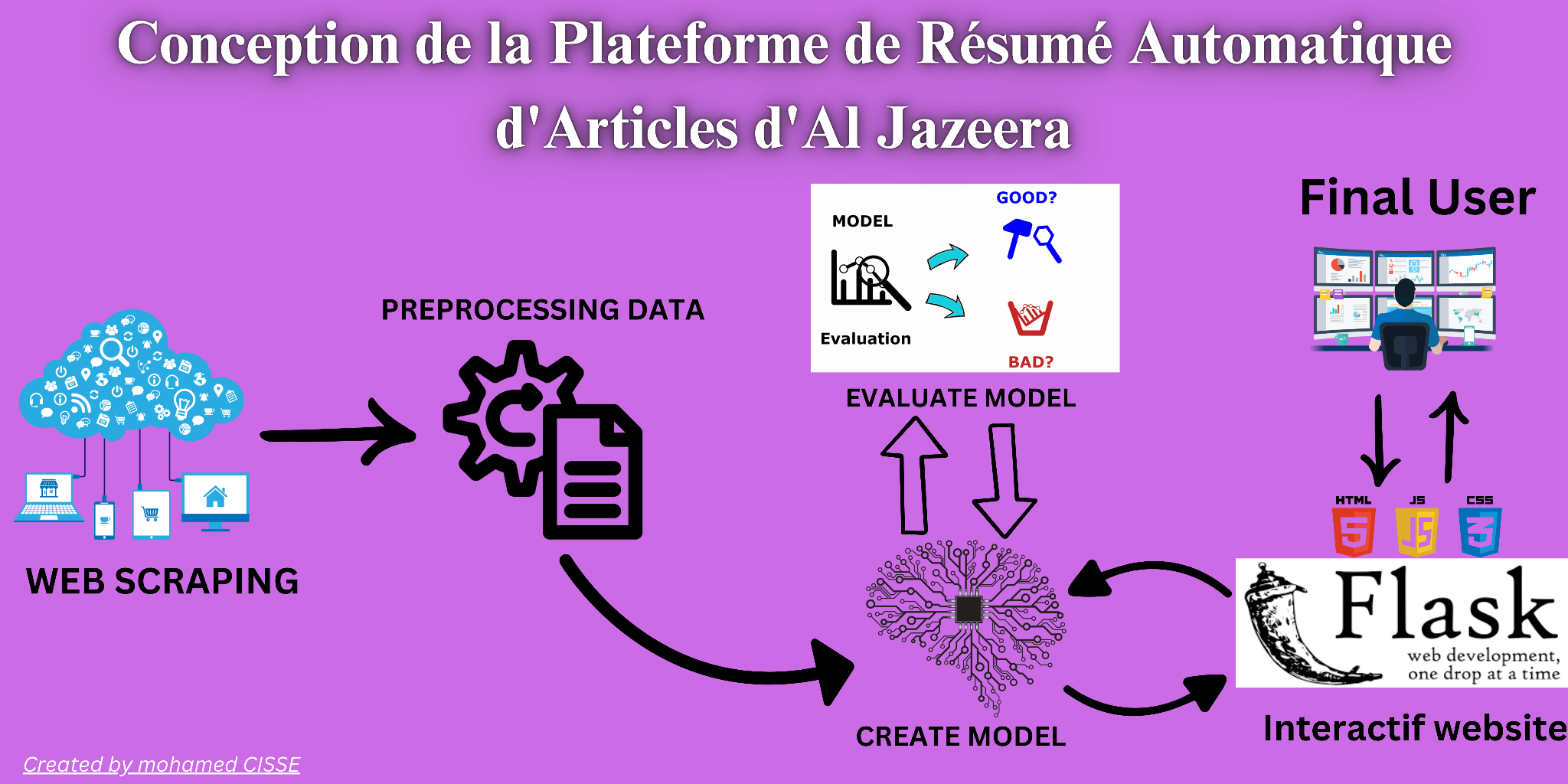
5. Enregistrement du modèle pour le déploiement :

- Sauvegarde du tokenizer et du modèle pour une utilisation future.

6. Déploiement :

- Utilisation du modèle et du tokenizer pour générer des résumés de nouveaux articles, y compris des sources différentes (Wikipédia).

Pour aller plus loin, vous pourriez envisager d'ajouter une interface utilisateur pour faciliter l'utilisation du modèle par des utilisateurs non techniques. Vous pourriez également explorer d'autres métriques d'évaluation et techniques d'amélioration des performances, en fonction des besoins spécifiques de votre projet.

Ci-joint une image de la conception de notre projet :

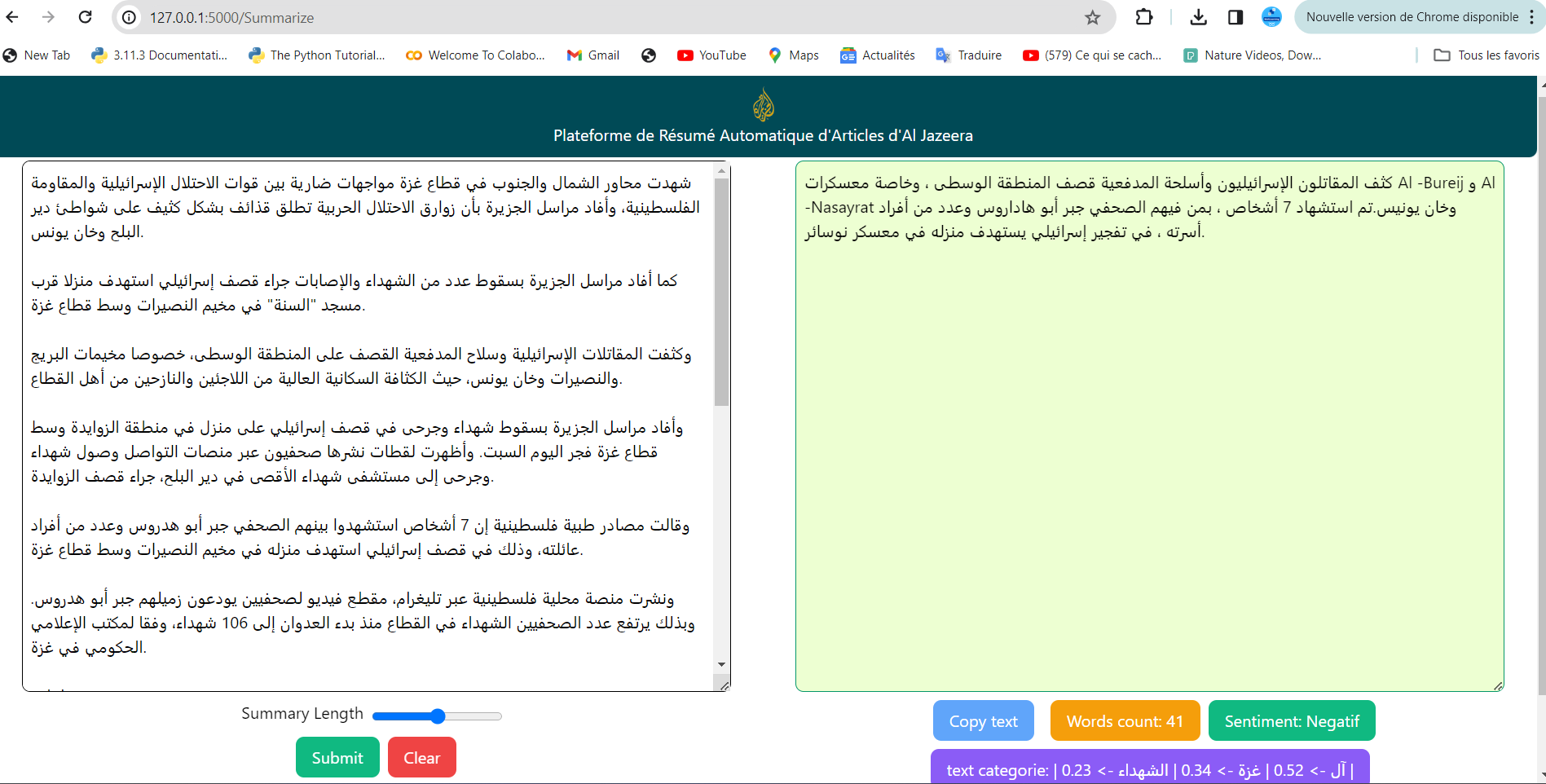
# **Implementation : visite de notre plateforme**

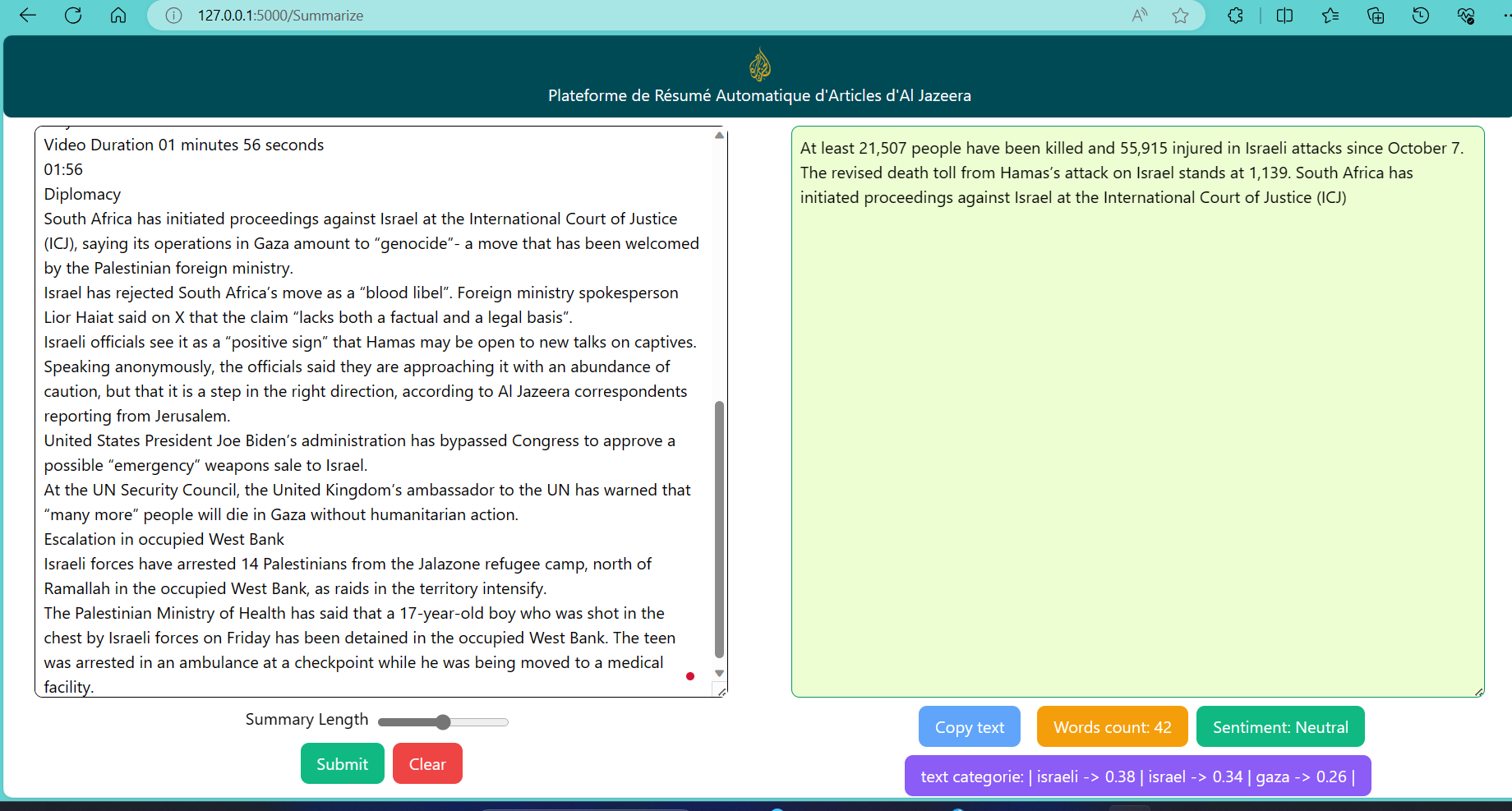
Notre interface web est une page HTML qui interagit avec le framework Flask pour déployer notre modèle de text summarization. Elle se compose de deux onglets : l'un dédié au texte original et l'autre à la génération du résumé. Grâce à la librairie Google Translate, notre plateforme peut être adaptée à toutes les langues, permettant ainsi la génération de résumés indépendamment de la langue du texte.

Nous avons choisi le modèle BART, reconnu pour sa performance, comme modèle principal pour la text summarization sur notre plateforme. Une fonctionnalité importante est la possibilité d'ajuster la taille du résumé généré à l'aide de la fonction SUMMARY LENGTH.

Une fois que le résumé est généré, les utilisateurs ont la possibilité de le copier grâce à la fonctionnalité "Copy Text" (bouton). En plus de la text summarization, notre plateforme propose d'autres fonctionnalités utiles, notamment le "Word Count", l'analyse de sentiment, et la catégorisation du texte (avec trois catégories proposées pour une meilleure classification).

Ci-joint, vous trouverez une capture d'écran de notre plateforme de text summarization :





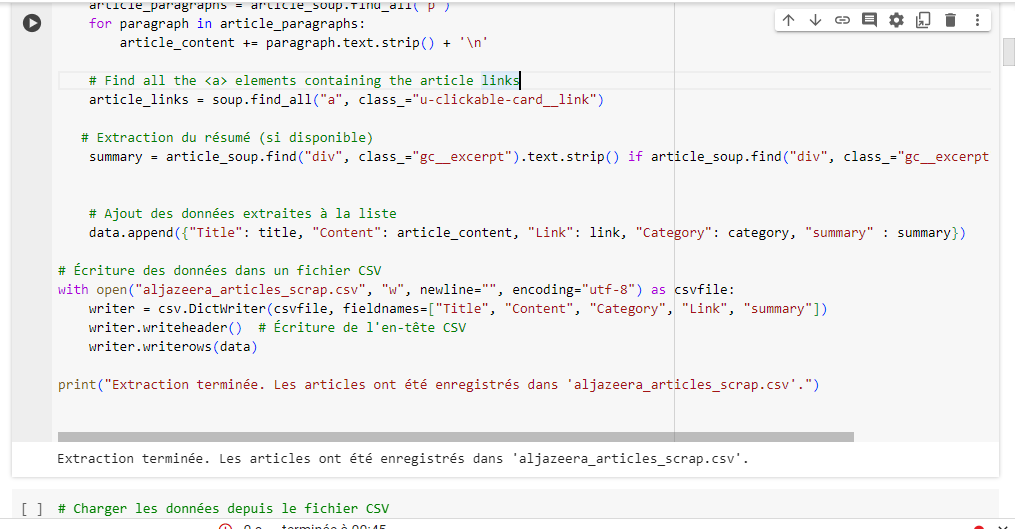
# **Web Scraping Aljazeera**

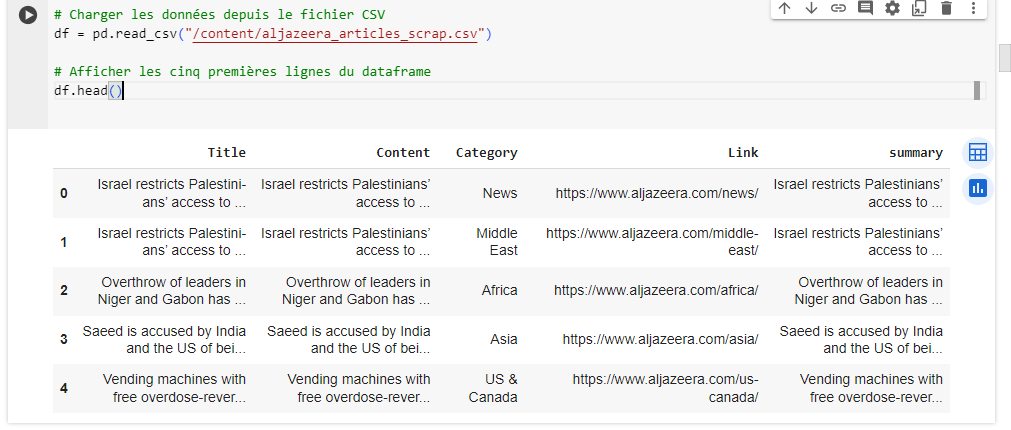
* **Scrapping du site Al Jazeera**

l' objectif est d'extraire des données du site Al Jazeera en anglais les bibliothèques BeautifulSoup et requests pour récupérer et analyser le contenu HTML des pages web. En utilisant la bibliothèque requests, il obtient le contenu HTML de la page Al Jazeera spécifiée dans la variable url. Ensuite, à l'aide de BeautifulSoup, le code analyse ce contenu pour trouver les éléments contenant les liens vers les articles.

Pour chaque lien d'article trouvé, le script récupère le titre, le contenu textuel, le lien, la catégorie et le résumé. Il parcourt les liens des articles, récupère le contenu HTML de chaque article et en extrait les détails associés.

Ces détails sont stockés dans une liste de dictionnaires où chaque dictionnaire représente les informations d'un article spécifique, y compris la catégorie à laquelle il appartient. Enfin, le script écrit ces données dans un fichier CSV nommé "aljazeera\_articles\_scrap.csv".

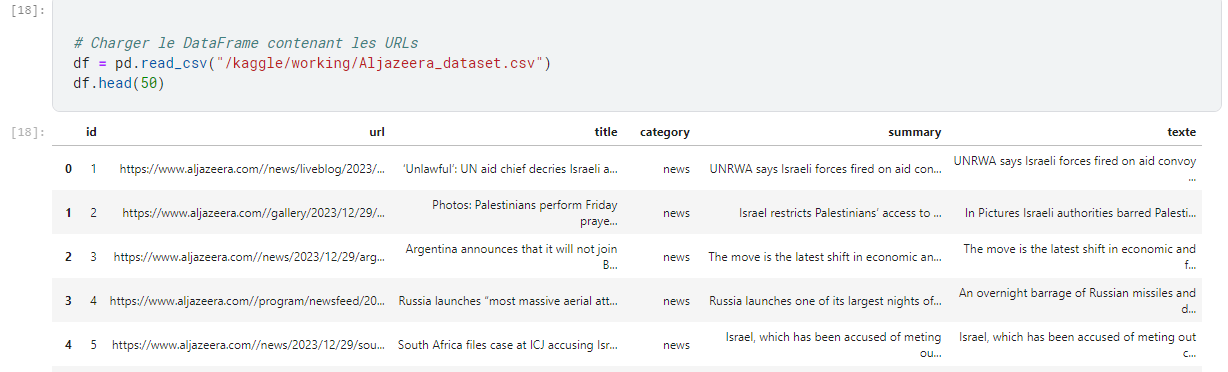
****

****

Dans le cadre du processus de collecte de données, notre objectif était d'extraire des informations du site web Al Jazeera. Nous avons réussi à l’aide **BeautifulSoup** à extraire un ensemble complet de données comprenant les champs suivants pour les articles en anglais : "url" (URL de l'article), "titre" (titre de l'article), "catégorie" (catégorie), "texte" (texte de l'article). Le DataFrame résultant comporte 320 entrées, fournissant ainsi une collection de données exhaustive pour une analyse approfondie ou un traitement ultérieur.

Parallèlement, dans le même processus de collecte de données, nous avons étendu notre extraction pour inclure des articles en arabe. Le DataFrame correspondant à cette langue compte désormais 350 entrées, offrant une base de données étendue pour des analyses ou des traitements ultérieurs.

Ces données, collectées à partir de deux langues différentes, enrichissent notre ensemble de données, permettant ainsi une exploration et une utilisation variées dans le cadre de projets d'analyse ou de traitement de texte.



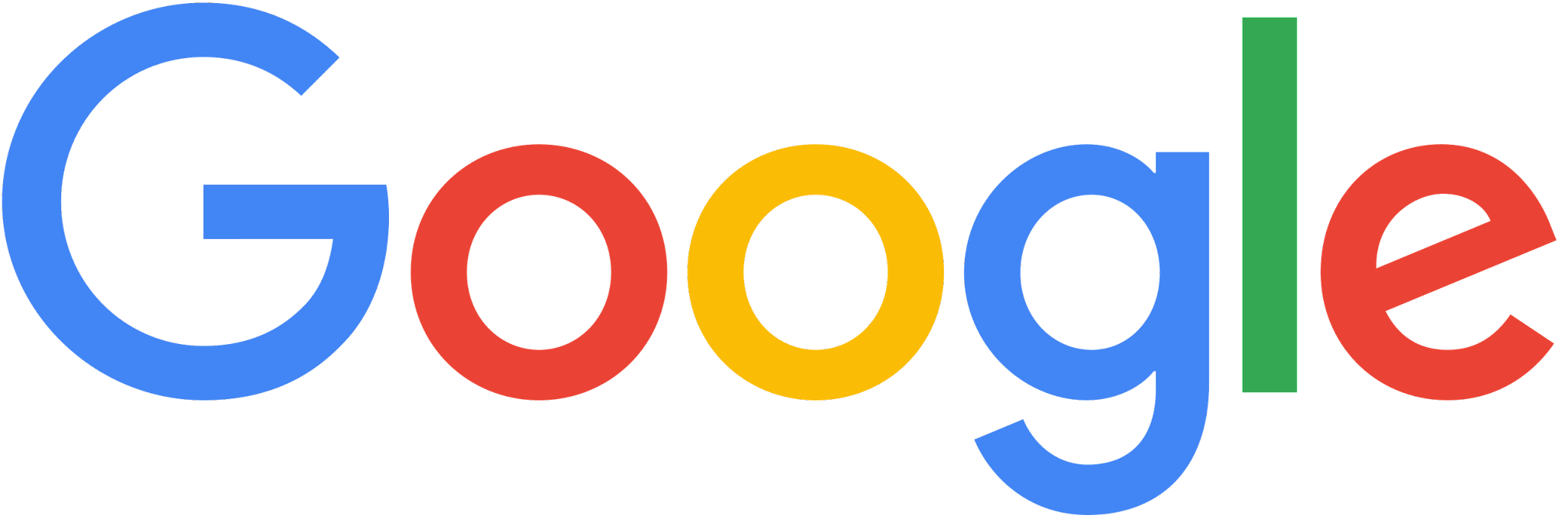


# **Modèle BART : Bidirectional + Auto-Regressif**

## **Définition :**

Le modèle BART est un modèle séquence-à-séquence entraîné en tant qu'autoencodeur de débruitage, capable de réaliser des tâches telles que la traduction automatique, la réponse aux questions, la summarization de texte, la classification de séquences, et d'autres applications spécifiques après un finetuning sur des jeux de données adaptés.

BART est bidirectionnel, ce qui signifie qu'il prend en compte le contexte à la fois à gauche et à droite d'un mot. Cela permet au modèle de mieux comprendre le contexte global du texte. De plus, il est auto-régressif, ce qui signifie qu'il génère du texte de manière séquentielle, un élément à la fois.

BART**=**BERT **+** GPT

La nature bidirectionnelle et auto-encodeur de BERT est un atout considérable pour les tâches en aval, telles que la classification, où l'information contextuelle de l'ensemble de la séquence est cruciale. Cependant, cela peut être moins adapté aux tâches de génération de texte, où la dépendance entre les mots générés est importante.

D'un autre côté, l'approche unidirectionnelle et autorégressive de GPT excelle dans la génération de texte, car chaque mot généré dépend strictement des mots précédemment générés. Cependant, elle peut être moins performante pour les tâches nécessitant une compréhension globale de la séquence, comme la classification.

BART, en tant que modèle qui combine des éléments de BERT et de GPT, offre une solution équilibrée. Il maintient la nature auto-encodeur de BERT tout en adoptant une approche autorégressive pour la génération de texte. Ainsi, BART bénéficie des avantages des deux architectures, en faisant le pont entre les tâches de classification nécessitant une compréhension globale et les tâches de génération de texte exigeant une dépendance séquentielle.

BART est le meilleur des deux mondes, offrant une flexibilité et une performance équilibrées pour une gamme étendue de tâches de summarisation.

## **Architecture BART :**

Le modèle BART (Bidirectional and Auto-Regressive Transformers) se distingue dans le contexte de la text summarization par plusieurs caractéristiques clés qui le rendent particulièrement adapté à cette tâche.

1. Spécifiquement conçu pour la génération de texte : BART a été pré-entraîné en mettant l'accent sur la génération de texte, ce qui le rend bien adapté aux tâches telles que la text summarization. Cette focalisation spécifique sur la génération de texte lui confère une capacité accrue à capturer les nuances et la structure du langage.

2. Bidirectionnel et auto-régressif : L'aspect bidirectionnel de BART lui permet de prendre en compte le contexte à la fois avant et après chaque mot, améliorant ainsi la compréhension globale du texte. De plus, son caractère auto-régressif signifie qu'il génère du texte de manière séquentielle, un pas à la fois, ce qui est particulièrement utile pour la génération de résumés cohérents.

3. Architecture Transformer : Basé sur l'architecture Transformer, BART exploite la puissance des mécanismes d'attention pour gérer efficacement les dépendances à long terme dans le texte. Cette architecture a prouvé son efficacité dans diverses tâches de traitement du langage naturel.

4. Disponibilité dans la bibliothèque Hugging Face Transformers : BART est accessible via la bibliothèque transformers de Hugging Face, facilitant ainsi son intégration dans des projets spécifiques. Cette disponibilité simplifie le processus de mise en œuvre et d'utilisation de modèles pré-entraînés, y compris BART, pour des applications de text summarization.

5. Paramètres pré-entraînés : BART a bénéficié d'un entraînement préalable sur d'énormes quantités de données, lui conférant une compréhension générale de la langue. Cette pré-entraînement approfondi est particulièrement avantageux pour des tâches spécifiques telles que la text summarization, où une compréhension contextuelle fine est cruciale.

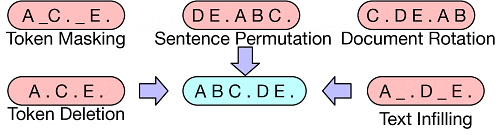
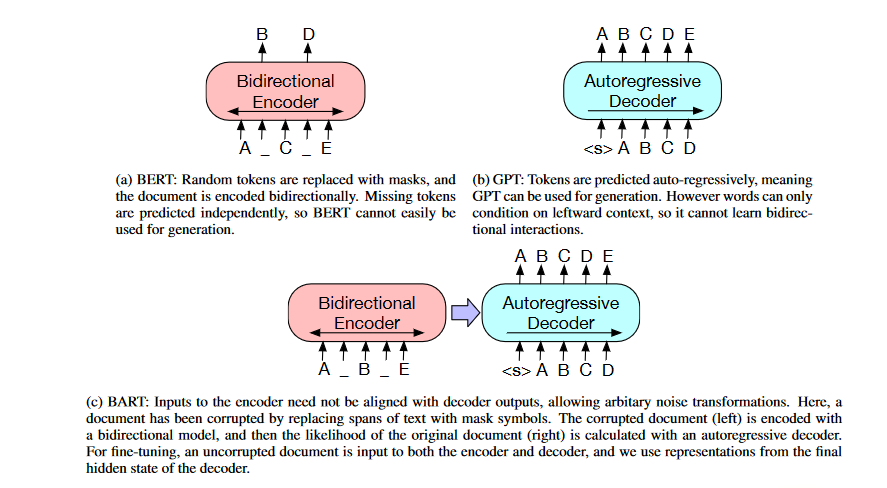
Quant au choix d'un modèle génératif pour la text summarization, il repose sur des considérations pertinentes :

- Génération de nouvelles phrases : Les modèles génératifs, comme BART, peuvent créer de nouvelles phrases pour le résumé, introduisant ainsi une variété et une originalité.

- Créativité : Les modèles génératifs sont plus créatifs et peuvent introduire de nouvelles formulations et idées, enrichissant le résumé.

- Fidélité au corpus de texte : Ils restent fidèles au corpus de texte d'origine, préservant l'intégrité du contenu original.

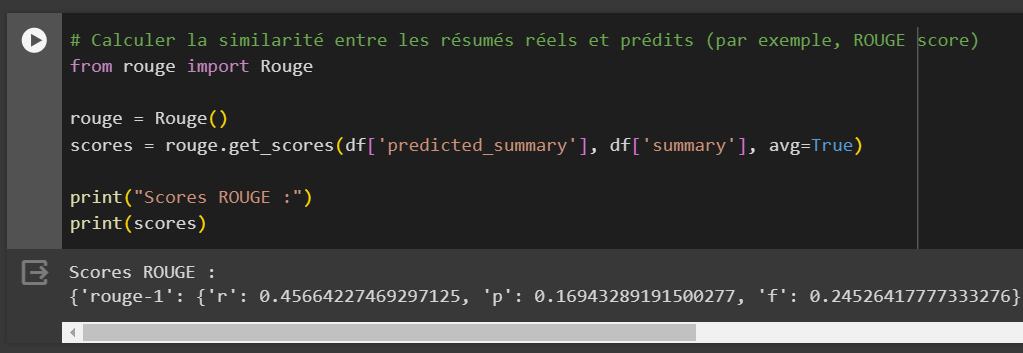
- Évaluation fidèle : Les modèles génératifs peuvent fournir une évaluation plus fidèle au résumé d'origine, ce qui est crucial pour maintenir la cohérence et la pertinence du résumé par rapport au texte source.



## **Evaluation du modèle BART :**

Les métriques ROUGE (Recall-Oriented Understudy for Gisting Evaluation) évaluent la qualité d'un résumé généré par rapport à un résumé de référence en mesurant la similarité des unigrammes, bigrammes et unités lexicales. Pour la tâche de résumé automatique de texte, les résultats indiquent que le modèle utilisé a atteint des scores de rappel (r) allant de 22,43% à 45,66% pour différentes n-grammes, démontrant ainsi sa capacité à rappeler des éléments importants du résumé de référence. Cependant, les scores de précision (p) sont relativement bas, variant de 7,09% à 16,94%, ce qui suggère que le modèle peut générer des mots ou expressions qui ne sont pas présents dans le résumé de référence. En outre, les mesures F1-score, combinant précision et rappel, oscillent entre 0,1070 et 0,2453, indiquant une performance modérée dans la génération de résumés précis. En choisissant le modèle BartForConditionalGeneration, ces résultats soulignent son équilibre entre rappel et précision pour la tâche spécifique de summarization de texte.

Ci-joint une capture de l’évaluation du Modèle BART



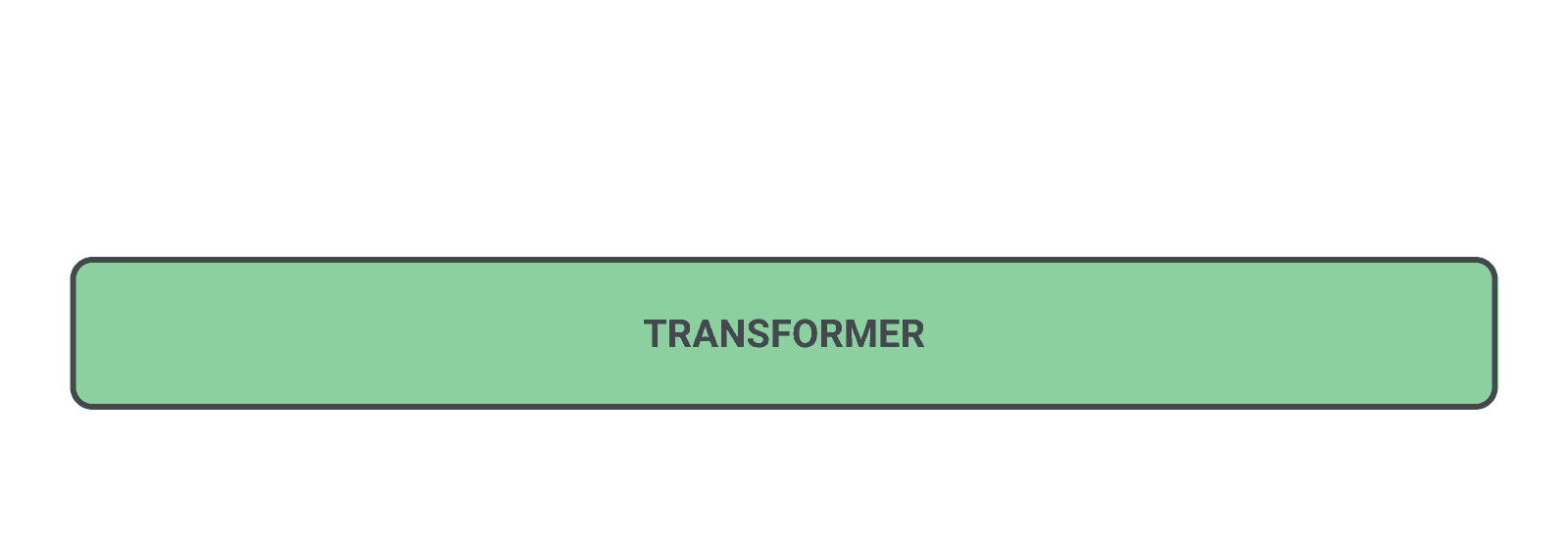
Les mesures **ROUGE-1, ROUGE-2 et ROUGE-L** indiquent que le résumé généré par le modèle BART a une performance modérée, avec des scores de rappel, de précision et de f1-score variables pour les unigrammes, les bigrammes et les unités lexicales par rapport au résumé de référence.

# **Pegasus**

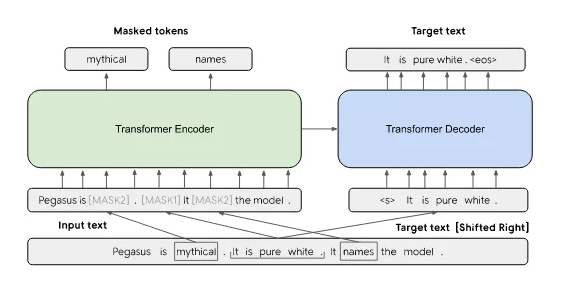
### **Compréhension de base de PEGASUS**

PEGASUS est une approche novatrice pour les modèles d'encodeur-décodeur basés sur Transformer, spécifiquement conçue pour la génération de résumés extractifs. Contrairement à d'autres modèles qui créent des résumés abstraits, PEGASUS se concentre sur la création de résumés en sélectionnant et en masquant des phrases clés dans le texte source, puis en générant un résumé à partir des parties non masquées.

Ellerepose sur l'architecture des Transformers, permettant une compréhension approfondie du texte pour extraire les informations essentielles. Contrairement à la génération de texte complet, PEGASUS sélectionne et organise les informations pertinentes pour produire des résumés extractifs fidèles au contenu original.



### **Architecture de PEGASUS**



L'architecture de PEGASUS est basée sur celle des Transformers, avec quelques adaptations spécifiques pour la génération de résumés extractifs. Voici une présentation succincte de son architecture :

**Transformers** : PEGASUS utilise des blocs d'encodeurs et de décodeurs, similaires à ceux des Transformers. Ces blocs sont composés de couches d'attention permettant au modèle de capturer les relations entre les mots dans le texte.

**Mécanismes d'attention** : Les mécanismes d'attention dans PEGASUS permettent au modèle de traiter efficacement les informations clés du texte source tout en masquant les parties moins importantes.

**Fine-tuning spécifique** : PEGASUS est souvent pré-entraîné sur de larges corpus de texte, puis fine-tuné sur des tâches spécifiques de génération de résumés extractifs. Ce processus affine ses paramètres pour une meilleure performance dans cette tâche spécifique.

**Masquage sélectif** : PEGASUS utilise un mécanisme de masquage pour se concentrer sur les parties essentielles du texte source lors de la génération de résumés extractifs, laissant de côté les parties moins pertinentes.

**Représentations multi-niveaux** : L'architecture de PEGASUS est conçue pour capturer des informations à différents niveaux de granularité, aidant ainsi à la sélection précise des parties importantes du texte source.

### **Entraînement et Fine-tuning**

L'entraînement de PEGASUS suit généralement plusieurs étapes, impliquant à la fois un entraînement initial sur de vastes ensembles de données textuelles et un fine-tuning spécifique pour la génération de résumés extractifs. Voici un aperçu de ces processus :

**Entraînement initial** : PEGASUS est initialement pré-entraîné sur de larges corpus de texte pour apprendre des représentations linguistiques générales. Cela permet au modèle d'acquérir une compréhension approfondie du langage.

**Fine-tuning pour les résumés extractifs** : Après l'entraînement initial, le modèle est fine-tuné sur des données spécifiques à la génération de résumés extractifs. Pendant cette phase, PEGASUS ajuste ses paramètres et ses poids pour s'adapter à cette tâche spécifique.

**Données annotées** : Le fine-tuning implique souvent l'utilisation de données annotées contenant des paires de textes et de résumés correspondants. Ces données permettent à PEGASUS de s'entraîner à générer des résumés extractifs précis en se basant sur des exemples annotés.

**Optimisation des hyperparamètres** : Pendant le fine-tuning, les hyperparamètres du modèle peuvent être ajustés pour maximiser les performances sur la tâche de génération de résumés extractifs, par exemple en minimisant une fonction de perte spécifique ou en optimisant une métrique d'évaluation.

**Itérations d'entraînement** : Le processus de fine-tuning peut nécessiter plusieurs itérations pour améliorer progressivement les performances du modèle. Ces itérations permettent d'adapter le modèle aux caractéristiques spécifiques des résumés extractifs.

**Évaluation et ajustements finaux** : À la fin du fine-tuning, le modèle est évalué sur des ensembles de validation ou de test pour mesurer ses performances. Des ajustements finaux peuvent être apportés pour optimiser davantage les performances du modèle.

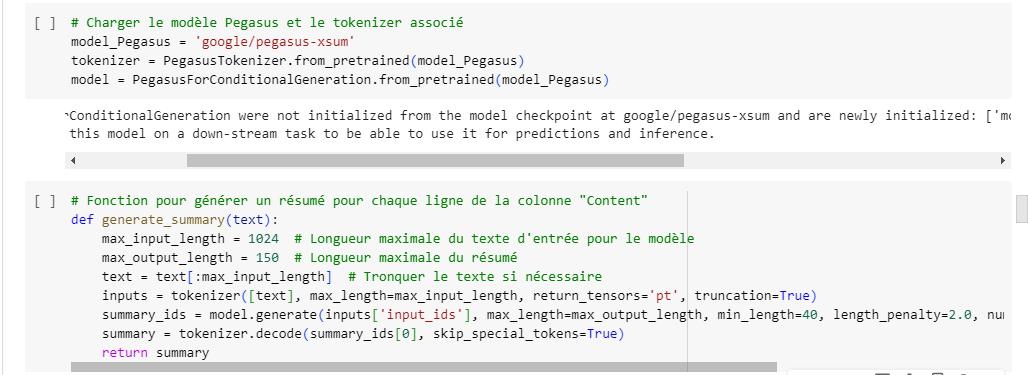
En combinant un entraînement initial sur des données générales avec un fine-tuning spécifique pour la génération de résumés extractifs, PEGASUS est optimisé pour produire des résumés précis et informatifs à partir de textes sources divers.

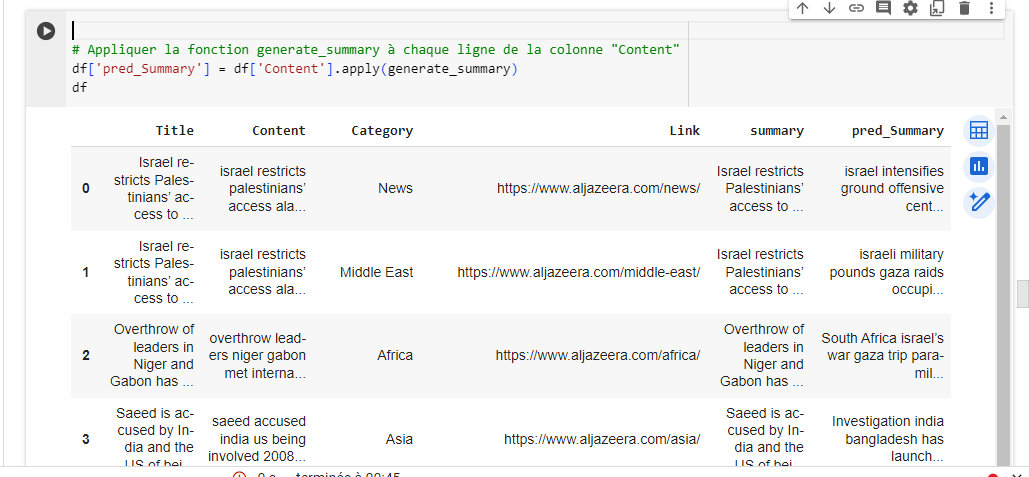
### **Application de Pegasus pour résumer les articles de Al Jazeera**

* **Utilisation du Modèle Pegasus pour générer le résumé**

On a utilisé le modèle Pegasus pour générer des résumés à partir du contenu extrait du site . Dans un premier temps, on charge le modèle Pegasus spécifié, ainsi que le tokenizer associé pour traiter le texte. Ensuite, une fonction nommée generate\_summary(text) est définie pour générer un résumé à partir d'un texte donné. Cette fonction fixe différents paramètres pour la génération du résumé, comme la longueur maximale du texte en entrée et la longueur maximale du résumé, ainsi que d'autres paramètres influençant le processus de génération.

En utilisant cette fonction, on applique la génération de résumés à chaque ligne de la colonne . Les résumés générés sont stockés dans une nouvelle colonne appelée 'pred\_Summary', ajoutant ainsi automatiquement ces résumés au DataFrame existant.





* **Evaluation du Modèle**

On évalue la qualité des résumés générés par le modèle PEGASUS en comparant ces résumés avec des résumés de référence.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE évalue la similarité entre le résumé généré et le résumé de référence en se basant sur le chevauchement des n-grammes (mots ou groupes de mots) entre les deux.

Trois types de scores ROUGE sont calculés : ROUGE-1, ROUGE-2 et ROUGE-L, évaluant respectivement la similarité basée sur des mots unigrammes, des bigrammes et des séquences.

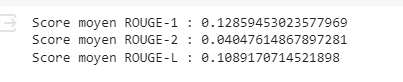
Pour chaque paire de résumé généré et de référence, le script calcule les scores ROUGE correspondants à l'aide de la fonction scorer.score(). Ces scores sont ensuite stockés dans des listes distinctes pour chaque métrique ROUGE. Enfin, le script calcule les scores moyens pour chaque métrique ROUGE en additionnant tous les scores obtenus puis en les divisant par le nombre total de résumés évalués.

**ROUGE-1** : Environ 12.86% de similarité entre les unigrammes (mots individuels) des résumés générés et ceux de référence.

**ROUGE-2:** Environ 4.05% de similarité entre les bigrammes (paires de mots consécutifs) des résumés générés et ceux de référence.

**ROUGE-L** : Environ 10.89% de similarité en prenant en compte la plus longue séquence de mots commune aux résumés générés et de référence.





# **T5: Text-To-Text Transfer Transformer**

## 1-Généraliés sur le modèle T5

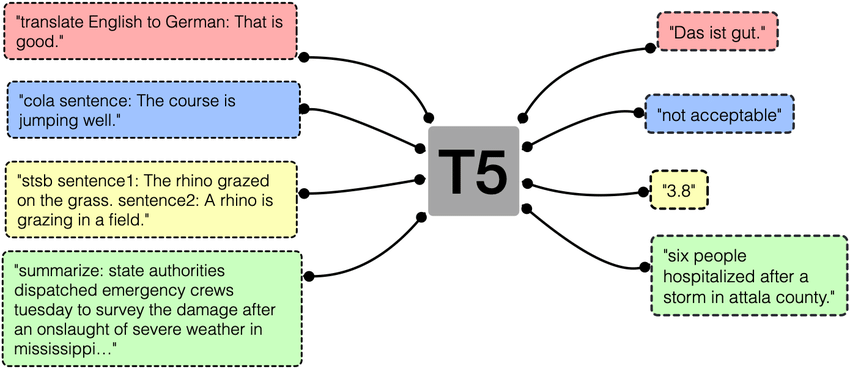


Figure 1 T5 :Text-to-Text Transfer Transformer

Le T5, ou "Text-to-Text Transfer Transformer", est un modèle de traitement du langage naturel (NLP) développé par Google. Il s'agit d'un modèle basé sur l'architecture Transformer qui utilise le transfert d'apprentissage pour résoudre différentes tâches de NLP.

L'architecture du T5 se compose d'un encodeur et d'un décodeur. L'encodeur traite l'entrée textuelle en capturant les informations contextuelles et sémantiques importantes du texte. Il utilise des mécanismes d'attention pour apprendre les relations entre les mots et les phrases, ce qui permet de représenter efficacement le contenu textuel. Le décodeur prend ensuite ces représentations et génère une sortie textuelle correspondant à la tâche spécifique à accomplir, comme la traduction, la résumé, la classification, etc.

Ce qui distingue le T5, c'est sa capacité à formuler toutes les tâches de NLP sous forme d'un problème de "transfert de texte à texte". Plutôt que d'entraîner des modèles spécifiques à chaque tâche, le T5 est entraîné à transformer une paire de phrases en une seule séquence de texte, en ajoutant un préfixe qui indique la tâche spécifique à effectuer. Par exemple, pour la traduction, une paire de phrases serait transformée en une séquence avec le préfixe "translate: phrase en français à traduire en anglais". Cette approche permet d'utiliser le même modèle pour différentes tâches de NLP, en adaptant simplement le préfixe.

Le T5 est pré-entraîné sur de grandes quantités de données textuelles provenant de diverses sources, ce qui lui permet d'apprendre des schémas et des structures du langage. Ensuite, il peut être finement ajusté (fine-tuned) sur des tâches spécifiques avec des ensembles de données plus restreints. Cette approche de transfert d'apprentissage permet au T5 d'obtenir de bons résultats sur une variété de tâches de NLP, y compris la génération de résumés, la traduction, la classification de texte et bien d'autres.

En résumé, le T5 est un modèle de traitement du langage naturel basé sur l'architecture Transformer. Il utilise le transfert d'apprentissage et une formulation de texte à texte pour résoudre différentes tâches de NLP. Grâce à son entraînement préalable sur de grandes quantités de données textuelles, il est capable de générer des résultats de haute qualité sur diverses tâches, y compris la génération de résumés de texte.

## 2-Implémentation du code

import torch

from transformers import AutoModelWithLMHead, AutoTokenizer

# Vérifier si un GPU est disponible, sinon utiliser le CPU

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

# Charger le tokenizer et le modèle T5 pré-entraîné

tokenizer = AutoTokenizer.from\_pretrained("mrm8488/t5-base-finetuned-summarize-news")

model = AutoModelWithLMHead.from\_pretrained("mrm8488/t5-base-finetuned-summarize-news")

model = model.to(device)

# Définir la longueur maximale des résumés générés

max\_length = 150

# Définir la fonction pour générer les résumés

def generate\_summary(article):

# Tokeniser l'article et convertir en tenseur PyTorch

input\_ids = tokenizer(article, return\_tensors="pt", truncation=True, padding=True, add\_special\_tokens=True).input\_ids.to(device)

# Générer le résumé en utilisant le modèle T5

generated = model.generate(input\_ids=input\_ids, num\_beams=2, max\_length=max\_length,

repetition\_penalty=2.5, length\_penalty=1.0, early\_stopping=True)[0]

# Décoder le résumé généré en texte lisible

output = tokenizer.decode(generated, skip\_special\_tokens=True, clean\_up\_tokenization\_spaces=True)

return output

# Appliquer la fonction generate\_summary à la colonne "texte" du DataFrame df

df["resume\_genere"] = df["texte"].apply(generate\_summary)

# Enregistrer le DataFrame avec les résumés et les scores ROUGE dans un fichier CSV

df.to\_csv("t5\_resume\_scores.csv", index=False)

Ce code charge le tokenizer et le modèle T5 pré-entraîné à l'aide des classes AutoTokenizer et AutoModelWithLMHead de la bibliothèque Transformers. Ensuite, il définit la fonction generate\_summary qui prend un article en entrée, le tokenise, génère un résumé en utilisant le modèle T5 et renvoie le résumé généré.

Le code applique ensuite la fonction generate\_summary à la colonne "texte" du DataFrame df, générant ainsi des résumés pour chaque article dans cette colonne. Les résumés générés sont ensuite enregistrés dans une nouvelle colonne appelée "resume\_genere".

Enfin, le DataFrame df avec les résumés et les scores ROUGE (qui ne sont pas inclus dans le code fourni) est enregistré dans un fichier CSV nommé "t5\_resume\_scores.csv".

## 3- Evaluation du code

ROUGE-1 score moyen : 0.17102292571431355

ROUGE-2 score moyen : 0.05560271607557514

ROUGE-L score moyen : 0.15909379353849362

Ces scores indiquent la similarité moyenne entre les résumés générés par le modèle T5 et les résumés de référence. Un score ROUGE-1 de 0.171 suggère que les résumés générés ont une correspondance de mots individuels d'environ 17%. De même, un score ROUGE-2 de 0.056 indique une correspondance de paires de mots d'environ 5,6%, tandis qu'un score ROUGE-L de 0.159 implique une correspondance de séquences de mots d'environ 15,9%.

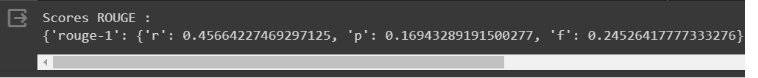
# Le modèle le plus performant (Evaluation)

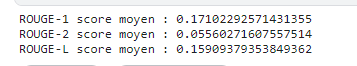
Les scores ROUGE sont des mesures d'évaluation automatique de la qualité d'un résumé en comparant le résumé généré avec le résumé de référence (réel). Il existe plusieurs métriques ROUGE, dont les plus couramment utilisées sont ROUGE-1, ROUGE-2 et ROUGE-L.

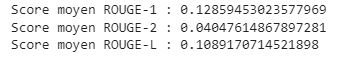
Nous avons opté pour l'utilisation d'un modèle génératif, en particulier le modèle BART, dans notre approche de text summarization. Ce choix s'explique par plusieurs raisons. Tout d'abord, un modèle génératif a la capacité de générer de nouvelles phrases pour le résumé, permettant ainsi d'introduire un nouveau style et de proposer des formulations et des idées créatives. De plus, ce type de modèle reste fidèle au corpus de texte d'origine, garantissant ainsi que le résumé généré reflète fidèlement les informations essentielles du texte source. Enfin, le modèle génératif, en particulier le modèle BART, a donné des évaluations plus fidèles aux résumés de référence, ce qui renforce sa pertinence dans notre approche.

Ci-joint les captures des accuracy des trois modèles :

**Bert**

**T5**

**Pegasus**



# **Fine Tuning VS Traduction**

## **1-Problématique :**

## Quelle approche est la plus efficace pour générer des résumés dans une langue cible: utiliser la traduction avec un modèle pré-entraîné ou effectuer un fine-tuning sur des données dans la langue cible ?

## 2-Traduction

L'utilisation de la traduction avec un modèle pré-entraîné présente un avantage majeur dans notre approche de génération de résumés. En nous basant sur le jeu de données d'Al Jazeera en anglais pour l'entraînement de notre modèle, nous avons introduit des fonctionnalités de traduction pour générer des résumés dans une langue cible spécifique. Cette approche nous a permis d'exploiter les modèles et les techniques de traduction automatique existants, ce qui a facilité la génération de résumés dans la langue cible.

Cependant, il convient de noter que l'utilisation de la traduction peut introduire des erreurs de traduction.

A screenshot of a computer

Description automatically generated

Figure 2 Traduction du text

## **3-Fine Tuning du modele T5**

Nous avons choisi d'effectuer un fine-tuning du modèle T5 pour notre tâche de génération de résumés dans notre langue cible spécifique. Le fine-tuning nous permet de spécialiser le modèle pré-entraîné T5 pour notre tâche spécifique, en l'adaptant aux particularités de notre langue et en améliorant ses performances dans la génération de résumés.

Les étapes du processus de fine-tuning que nous avons suivies sont les suivantes :

1. Collecte de données : Nous avons rassemblé un ensemble de données annotées spécifiques à notre langue cible et à notre tâche de génération de résumés. Ces données ont été soigneusement sélectionnées pour assurer la représentativité et la diversité nécessaires à l'entraînement du modèle.

2. Prétraitement des données : Les données collectées ont été prétraitées en utilisant des techniques telles que la tokenisation et la conversion dans un format compatible avec le modèle T5. Cela garantit que les données sont prêtes à être utilisées pour le fine-tuning.

3. Configuration du modèle de fine-tuning : Nous avons ajusté les hyperparamètres du modèle T5 pour notre tâche spécifique de génération de résumés dans notre langue cible. Cela comprend la sélection de la taille du batch, le choix de l'algorithme d'optimisation et le réglage des taux d'apprentissage.

4. Fine-tuning : Nous avons entraîné le modèle T5 en utilisant les données prétraitées et les hyperparamètres configurés. Le modèle a été ajusté pour mieux se spécialiser dans la génération de résumés dans notre langue cible, en adaptant ses poids et ses paramètres.

5. Validation et ajustement : Nous avons suivi les performances du modèle pendant le processus de fine-tuning en utilisant un ensemble de validation distinct. Cela nous a permis d'évaluer la qualité des résumés générés et d'effectuer des ajustements aux hyperparamètres si nécessaire.

6. Évaluation : Une fois le fine-tuning terminé, nous avons évalué les performances du modèle sur un ensemble de test indépendant. Cela nous a permis de mesurer l'efficacité du fine-tuning dans la génération de résumés dans notre langue cible.

Le processus de fine-tuning du modèle T5 nécessite des ressources de calcul significatives et une expertise technique pour être mené à bien. Cependant, il offre l'avantage d'adapter le modèle pré-entraîné à notre tâche spécifique, améliorant ainsi sa capacité à générer des résumés précis et cohérents dans notre langue cible."

## **4-Implementation du code**

Pour préparer notre jeu de données Al Jazeera News pour le fine-tuning du modèle T5,nous avons chargé le jeu de données à partir du fichier CSV, divisé le jeu de données en ensembles d'entraînement, de validation et de test, et converti les ensembles de données en dictionnaires avec les noms des caractéristiques correspondantes. Ces étapes nous ont permis d'obtenir un objet dataset\_dict contenant les données prétraitées prêtes à être utilisées dans notre pipeline de fine-tuning avec le modèle T5.

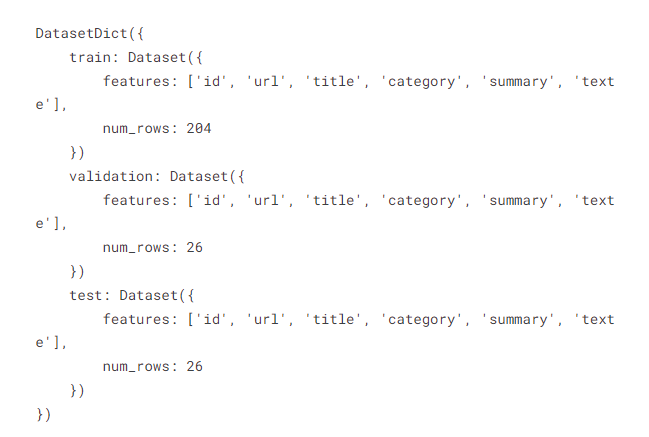


Figure 3 dataset Aljazeera

Puis nous avons effectué des étapes de prétraitement pour préparer nos données. Nous avons utilisé le tokenizer associé au modèle T5 avec le modèle de base "t5-base". Les entrées ont été encodées en utilisant une longueur maximale de 1024 tokens, tandis que les résumés ont été encodés avec une longueur maximale de 128 tokens. En ajoutant un préfixe spécifique au modèle T5, nous avons guidé son comportement lors de la génération de résumés. Les labels encodés ont été inclus dans les données d'entrée du modèle en utilisant la clé "labels" dans le dictionnaire model\_inputs. Ces étapes de prétraitement ont permis d'obtenir un objet tokenized\_datasets contenant les données prétraitées, prêtes pour le fine-tuning du modèle T5.

A screenshot of a computer

Description automatically generated

Figure 4 Tokenization

Pour affiner le modèle préentraîné, nous utiliserons la classe `AutoModelForSeq2SeqLM` de la bibliothèque `transformers`. Nous pouvons télécharger et mettre en cache le modèle à l'aide de la méthode `from\_pretrained`. Voici un extrait de code :

```python

from transformers import AutoModelForSeq2SeqLM, DataCollatorForSeq2Seq, Seq2SeqTrainingArguments, Seq2SeqTrainer

model = AutoModelForSeq2SeqLM.from\_pretrained(model\_checkpoint)

```

Ensuite, nous devons définir la classe `Seq2SeqTrainingArguments`, qui contient plusieurs attributs pour personnaliser le processus d'entraînement. Dans cet exemple, nous définissons la stratégie d'évaluation à effectuer à la fin de chaque epoch, ajustons le taux d'apprentissage, spécifions la taille de batch, le poids de la décroissance, et d'autres paramètres d'entraînement. Nous activons également l'option de prédiction avec des sorties générées et la formation en précision mixte pour un traitement plus rapide. Voici un extrait de code :

```python

taille\_batch = 8

nom\_modele = model\_checkpoint.split("/")[-1]

args = Seq2SeqTrainingArguments(

f"{nom\_modele}-affine-xsum",

evaluation\_strategy="epoch",

learning\_rate=2e-5,

per\_device\_train\_batch\_size=taille\_batch,

per\_device\_eval\_batch\_size=taille\_batch,

weight\_decay=0.01,

save\_total\_limit=3,

num\_train\_epochs=20,

predict\_with\_generate=True,

fp16=True,

)

```

Ensuite, nous devons instancier un data collator spécial, `DataCollatorForSeq2Seq`, qui non seulement remplit les entrées à la longueur maximale dans le batch, mais également remplit les labels. Voici un extrait de code :

```python

data\_collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=model)

```

Enfin, nous créons une instance de `Seq2SeqTrainer` en fournissant le modèle, les arguments d'entraînement, les jeux de données d'entraînement et d'évaluation, le data collator, le tokenizer et une fonction de calcul des métriques facultative. Voici un extrait de code :

```python

trainer = Seq2SeqTrainer(

model=model,

args=args,

train\_dataset=tokenized\_datasets["train"],

eval\_dataset=tokenized\_datasets["validation"],

data\_collator=data\_collator,

tokenizer=tokenizer,

compute\_metrics=compute\_metrics

)

```

Une fois que tous les éléments sont configurés, nous pouvons lancer l'entraînement du modèle en utilisant la méthode `train()` de l'objet `trainer`.

## **5- Résultats et Discussion :**

Nos résultats ont montré une amélioration progressive des métriques ROUGE, ce qui suggère une meilleure qualité des résumés générés par nos modèles. Cela indique que nos modèles ont été capables de comprendre le contenu des documents sources et de générer des résumés informatifs et pertinents.

A screenshot of a computer

Description automatically generated

Figure 5 fine tuning T5 Base scores

En ce qui concerne le modèle en anglais, nous avons poussé le modèle finetuné vers le Hub Hugging Face, ce qui nous permet de le partager et de le sauvegarder facilement. Vous pouvez trouver le modèle finetuné en anglais à l'adresse suivante https://huggingface.co/ibtissam369/t5-base-finetuned-summarize-news-finetuned-xsum

Pour le modèle en arabe, nous avons utilisé une autre architecture, AraT5v2-base-1024, et avons réalisé un processus similaire d'entraînement et d'évaluation.Pour cela , on a collecté des données à partir du site Al Jazeera (https://www.aljazeera.net/), Le modèle finetuné en arabe peut être consulté à l'adresse :

https://huggingface.co/ibtissam369/AraT5v2-base-1024-finetuned-ALjazeera

# Conclusion

En conclusion, les modèles de réseaux de neurones récurrents (RNN) et les modèles de langage à grande échelle (LLMs) ont démontré leur efficacité et leur utilité dans les tâches de résumé. Leur capacité à capturer les dépendances contextuelles à long terme et à être adaptés via le fine-tuning les rend particulièrement adaptés à cette tâche.

Cependant, il reste encore des défis à relever. La génération de résumés cohérents et informatifs demeure un enjeu, car les modèles peuvent parfois produire des résumés qui manquent de clarté ou qui contiennent des informations incorrectes. De plus, la gestion des ressources computationnelles reste un défi, car ces modèles sont souvent complexes et nécessitent une puissance de calcul importante.

En outre, l'extension de ces modèles à des langues spécifiques, telles que le darija, nécessite des efforts supplémentaires de collecte de données et de formation. La disponibilité de données de haute qualité dans ces langues est essentielle pour obtenir des performances optimales dans les tâches de résumé.

Malgré ces défis, les RNN et les LLMs continuent d'évoluer et de s'améliorer, grâce aux avancées constantes dans le domaine de l'apprentissage automatique. Ils offrent un potentiel prometteur pour l'automatisation des tâches de résumé et la génération de contenu condensé et informatif.

En somme, les modèles RNN et LLMs ouvrent de nouvelles perspectives dans le domaine des résumés automatiques. Leur utilisation peut apporter des bénéfices importants dans des domaines tels que le traitement automatique du langage naturel, le journalisme automatisé et la recherche d'informations. Cependant, une attention continue doit être accordée à leur amélioration et à leur adaptation pour relever les défis actuels et futurs liés à la génération de résumés de haute qualité.

# Référence

* Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv preprint arXiv:1910.10683.
* Nagoudi, E. M. B., Elmadany, A., & Abdul-Mageed, M. (2022). AraT5: Text-to-Text Transformers for Arabic Language Generation. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 628–647). Dublin, Ireland: Association for Computational Linguistics.

repository: <https://github.com/UBC-NLP/araT5>.