Faculté des Sciences et Techniques
Kénitra

Master d'Excellence

Intelligence Artificielle & Objets Connectés (IAOC)

Mini-Projet NLP

Mise en œuvre d'une solution intelligente basée sur le NLP pour la Classification Automatique d'Articles de Presse

« News Article Classifier »

Réalisé par :
Wajih ESGHAYRI

Encadré par :
Pr. Charaf OUADDI

Année Universitaire : 2025-2026

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

NLP	Natural Language Processing (Traitement Automatique du Langage Naturel)
BERT	Bidirectional Encoder Representations from Transformers
DistilBERT	Distilled version of BERT
AI	Artificial Intelligence (Intelligence Artificielle)
ML	Machine Learning (Apprentissage Automatique)
GPU	Graphics Processing Unit
API	Application Programming Interface
PDF	Portable Document Format
CSV	Comma-Separated Values
HuggingFace	Plateforme de modèles de NLP pré-entraînés

Introduction générale

1) Contexte général

Dans l'ère numérique actuelle, la quantité d'informations textuelles disponibles en ligne croît de manière exponentielle. Les médias en ligne publient quotidiennement des milliers d'articles de presse couvrant divers sujets. La classification automatique de ces articles devient une tâche essentielle pour organiser, filtrer et recommander du contenu pertinent aux utilisateurs.

Le traitement automatique du langage naturel (NLP) offre des solutions performantes pour automatiser cette tâche. Les modèles de deep learning, notamment les transformers comme BERT, ont révolutionné le domaine en atteignant des performances exceptionnelles sur diverses tâches de compréhension du langage.

Ce projet s'inscrit dans cette perspective en développant une solution intelligente capable de classifier automatiquement des articles de presse en quatre catégories principales : World (Monde), Sports, Business (Affaires) et Sci/Tech (Sciences et Technologies).

2) Objectifs

Les objectifs principaux de ce projet sont :

1. **Développer un modèle de classification** performant basé sur DistilBERT pour catégoriser automatiquement des articles de presse
2. **Créer une interface utilisateur intuitive** permettant aux utilisateurs de classifier leurs propres articles
3. **Déployer la solution** sous forme d'application web accessible et facile à utiliser
4. **Évaluer les performances** du modèle sur différentes métriques (accuracy, précision, rappel, F1-score)
5. **Optimiser le modèle** pour obtenir le meilleur compromis entre performance et temps d'inférence

3) Organisation du rapport

Ce rapport est organisé en cinq chapitres principaux :

— **Chapitre 1** présente le corpus utilisé et ses caractéristiques
— **Chapitre 2** détaille les étapes de prétraitement des données textuelles
— **Chapitre 3** décrit la préparation et la représentation des données pour l'entraînement
— **Chapitre 4** expose l'architecture du modèle proposé
— **Chapitre 5** couvre l'implémentation, l'entraînement, le déploiement et les résultats obtenus

Chapitre 1

Préparation du Corpus

1.1 Introduction

Dans ce chapitre, nous présenterons une description détaillée du corpus utilisé pour entraîner notre modèle de classification d'articles de presse. Le choix du corpus est une étape cruciale qui influence directement la qualité et la performance du modèle final. Nous décrirons les caractéristiques du dataset, sa structure, et les statistiques principales qui le composent.

1.2 Corpus utilisé : AG News Dataset

1.2.1 Présentation du corpus

Pour ce projet, nous avons utilisé le dataset **AG News**, l'un des corpus de référence pour la classification de textes en NLP. AG News est un corpus d'articles de presse collectés à partir de plus de 2000 sources d'information.

Caractéristiques principales :

— **Source :** AG's corpus of news articles (Academic)
— **Nombre total d'exemples :** 120,000 articles
— **Nombre de classes :** 4 catégories
— **Langue :** Anglais
— **Format :** Texte brut avec labels

1.2.2 Les catégories du corpus

Le dataset AG News contient quatre catégories principales :

1. **World (Monde)** : Articles concernant l'actualité internationale, événements géopolitiques, conflits, diplomatie
2. **Sports** : Articles sur les événements sportifs, résultats de matchs, actualités des athlètes
3. **Business (Affaires)** : Articles économiques, marchés financiers, entreprises, commerce

4. **Sci/Tech (Sciences et Technologies)** : Articles sur les innovations technologiques, découvertes scientifiques, produits high-tech

1.2.3 Structure du dataset

Chaque exemple du corpus contient :

— **Titre de l'article :** Le titre principal de l'article de presse

— **Description :** Un résumé ou extrait de l'article

— **Label :** La catégorie de l'article (0 : World, 1 : Sports, 2 : Business, 3 : Sci/Tech)

1.3 Statistiques du corpus

TABLE 1.1 – Statistiques du corpus AG News

Métrique	**Valeur**
Nombre total d'articles	120,000
Articles d'entraînement	120,000
Nombre de classes	4
Classe 0 (World)	30,000 articles
Classe 1 (Sports)	30,000 articles
Classe 2 (Business)	30,000 articles
Classe 3 (Sci/Tech)	30,000 articles
Distribution des classes	Équilibrée
Longueur moyenne (mots)	40-50 mots

1.3.1 Distribution équilibrée

Un aspect important du dataset AG News est que les classes sont parfaitement équilibrées, avec 30,000 exemples par catégorie. Cette distribution équilibrée évite les problèmes de biais de classe et permet au modèle d'apprendre de manière uniforme sur toutes les catégories.

1.3.2 Qualité des données

Le corpus AG News est reconnu pour sa qualité :

— Textes bien structurés et grammaticalement corrects

— Articles provenant de sources journalistiques fiables

— Catégorisation claire et cohérente

— Pas de données manquantes significatives

1.4 Exploration du corpus

1.4.1 Analyse des longueurs de texte

L'analyse de la longueur des articles montre que la plupart des textes contiennent entre 30 et 70 mots (titre + description). Cette longueur est appropriée pour l'entraînement de modèles de transformers comme DistilBERT qui ont une limite de 512 tokens.

1.4.2 Vocabulaire et diversité lexicale

Le corpus présente une grande diversité lexicale avec un vocabulaire riche et varié selon les catégories. Les articles sportifs utilisent un vocabulaire technique spécifique au sport, tandis que les articles business emploient un jargon économique et financier.

1.5 Conclusion

Dans cette partie, nous avons présenté le corpus AG News utilisé pour notre projet de classification d'articles de presse. Ce dataset bien structuré et équilibré offre une base solide pour l'entraînement de notre modèle. Le chapitre suivant sera consacré aux techniques de prétraitement appliquées pour préparer ces données textuelles en vue de l'entraînement du modèle DistilBERT.

Chapitre 2

Prétraitement des données

2.1 Introduction

Dans ce chapitre, nous allons décrire les étapes de prétraitement appliquées au corpus AG News. Le prétraitement est une phase cruciale dans tout projet de NLP car il permet de transformer les données textuelles brutes en un format exploitable par les modèles de machine learning.

2.2 Prétraitement du corpus

2.2.1 Nettoyage des textes

Le nettoyage des textes comprend plusieurs étapes :

a) Suppression des caractères spéciaux et ponctuation excessive

— Élimination des caractères non-alphanumériques inutiles

— Conservation de la ponctuation essentielle pour la compréhension du contexte

— Normalisation des espaces multiples

b) Gestion de la casse

— Les modèles BERT sont sensibles à la casse

— Conservation de la casse originale pour préserver l'information sémantique

— DistilBERT utilise une version "uncased" dans certains cas

2.2.2 Tokenization avec DistilBERT Tokenizer

La tokenization est effectuée en utilisant le **DistilBertTokenizerFast** de HuggingFace :

Caractéristiques de la tokenization :

— Utilisation de WordPiece tokenization

— Vocabulaire pré-entraîné de 30,000 tokens

— Ajout automatique des tokens spéciaux [CLS] et [SEP]

— Gestion du padding et de la troncature

Tokens spéciaux :

— **[CLS]** : Token de début de séquence (utilisé pour la classification)
— **[SEP]** : Token de séparation entre segments
— **[PAD]** : Token de padding pour uniformiser les longueurs
— **[UNK]** : Token pour les mots inconnus

2.2.3 Gestion de la longueur des séquences

Paramètres de tokenization :
— **max_length** : 512 tokens (limite du modèle BERT)
— **padding** : True (ajout de padding pour uniformiser les longueurs)
— **truncation** : True (troncature des séquences trop longues)

2.3 Encodage des labels

2.3.1 Transformation des catégories

Les labels textuels sont convertis en valeurs numériques :
— World $\rightarrow$ 0
— Sports $\rightarrow$ 1
— Business $\rightarrow$ 2
— Sci/Tech $\rightarrow$ 3

2.3.2 Format des données

Après prétraitement, chaque exemple est transformé en un dictionnaire contenant :
— **input_ids** : Séquence de tokens encodés
— **attention_mask** : Masque indiquant les tokens réels vs padding
— **labels** : Label numérique de la classe

2.4 Conclusion

Nous avons abordé dans ce chapitre le prétraitement de notre corpus AG News. Les étapes de nettoyage, tokenization et encodage sont essentielles pour transformer les textes bruts en données exploitables par le modèle DistilBERT. Le chapitre suivant présentera la représentation des données et leur répartition en ensembles d'entraînement, validation et test.

Chapitre 3

Préparation des données

3.1 Introduction

Dans cette partie, nous présenterons la manière dont les données prétraitées ont été structurées et réparties pour l'entraînement du modèle.

3.2 Représentation des données

3.2.1 Embeddings contextuels

DistilBERT génère des **embeddings contextuels** pour chaque token :

— **Dimension des embeddings :** 768 dimensions

— **Type d'embedding :** Contextuel (chaque token est représenté en fonction de son contexte)

— **Avantage :** Capture des relations sémantiques complexes

3.3 Répartition des données

TABLE 3.1 – Répartition des données

Ensemble	**Nombre d'exemples**	**Pourcentage**
Train set	96,000	80%
Validation set	12,000	10%
Test set	12,000	10%
Total	**120,000**	**100%**

3.3.1 Train set (Ensemble d'entraînement)

— **Taille :** 96,000 articles

— **Utilisation :** Entraînement des paramètres du modèle

— Distribution équilibrée des 4 classes (24,000 articles par catégorie)

3.3.2 Validation set (Ensemble de validation)

— **Taille :** 12,000 articles
— **Utilisation :** Ajustement des hyperparamètres et early stopping
— **Rôle :** Évaluation pendant l'entraînement, détection du surapprentissage

3.3.3 Test set (Ensemble de test)

— **Taille :** 12,000 articles
— **Utilisation :** Évaluation finale des performances
— Données jamais vues pendant l'entraînement

3.4 Conclusion

Nous avons traité dans ce chapitre la préparation des données avec la représentation vectorielle via DistilBERT et la répartition en ensembles d'entraînement, validation et test. Cette organisation méthodique des données garantit un entraînement robuste et une évaluation fiable.

Chapitre 4

Architecture proposée

4.1 Introduction

Dans ce chapitre, nous présentons l'architecture du modèle proposé pour la classification d'articles de presse. Notre solution repose sur **DistilBERT**, une version allégée et optimisée du modèle BERT.

4.2 Choix du modèle : DistilBERT

4.2.1 Pourquoi DistilBERT ?

Pour ce projet, nous avons sélectionné **DistilBERT** (Distilled BERT) comme modèle de base. Ce choix résulte d'une analyse approfondie des différentes architectures de transformers disponibles.

Avantages de DistilBERT :

- **Performance :** Conserve 97% des capacités de BERT grâce à la distillation de connaissances
- **Efficacité :** 40% plus rapide que BERT en inférence
- **Taille réduite :** 40% moins de paramètres (66M vs 110M pour BERT-base)
- **Mémoire :** Nécessite environ 60% moins de ressources GPU/RAM
- **Inférence rapide :** Idéal pour les applications en production et déploiement web
- **Pré-entraînement :** Déjà entraîné sur un large corpus (Wikipedia + BookCorpus)

4.2.2 Processus de distillation

DistilBERT est obtenu par un processus de **knowledge distillation** où un modèle plus petit (élève) apprend à imiter le comportement d'un modèle plus grand (professeur - BERT). Ce processus permet de :

- Réduire la taille du modèle sans perte significative de performance
- Accélérer l'entraînement et l'inférence

— Maintenir la capacité de compréhension contextuelle du langage

4.2.3 Comparaison BERT vs DistilBERT

TABLE 4.1 – Comparaison BERT vs DistilBERT

Caractéristique	BERT-base	DistilBERT
Nombre de paramètres	110M	66M
Nombre de couches	12	6
Temps d'inférence	1x	0.6x
Mémoire requise	1x	0.6x
Performance	100%	97%

4.3 Architecture du modèle

4.3.1 Architecture globale

L'architecture du modèle suit le schéma suivant :

1. Input Text (Article)
2. Tokenization (DistilBertTokenizer)
3. Embedding Layer (768 dimensions)
4. 6 Transformer Layers
   — Multi-Head Self-Attention (12 heads)
   — Feed Forward Network
   — Layer Normalization

CLS Token Representation

5. Classification Head (Linear Layer)
6. Softmax
7. Output (4 classes)

4.3.2 Composants détaillés

1. Couche d'embedding

— Token embeddings : 768 dimensions
— Position embeddings : Encodage de la position des tokens

2. Transformer Encoder

— 6 couches de transformers (vs 12 pour BERT)
— Attention multi-têtes : 12 têtes d'attention
— Dimension cachée : 768
— Dimension intermédiaire : 3072

— Activation : GELU

3. Classification Head

— Pooling : Extraction de la représentation du token [CLS]

— Couche dense : 768 $\rightarrow$ 4 (nombre de classes)

— Dropout : 0.1 (régularisation)

— Activation finale : Softmax

4.4 Fonction de perte

Cross-Entropy Loss pour classification multi-classe :

$$Loss = -\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i) \tag{4.1}$$

Où :

— y_i : Label réel (one-hot encoded)

— $\hat{y}_i$: Probabilité prédite pour la classe i

4.5 Conclusion

L'architecture proposée basée sur DistilBERT offre un excellent compromis entre performance et efficacité. Le modèle pré-entraîné, combiné avec une couche de classification personnalisée, permet d'obtenir des résultats de haute qualité pour la classification d'articles de presse.

Chapitre 5

Implémentation

5.1 Introduction

Ce chapitre présente l'implémentation concrète de notre solution de classification d'articles de presse. Nous décrirons les outils et technologies utilisés, le processus d'entraînement du modèle, le déploiement de l'application web, et les résultats obtenus.

5.2 Outils et technologies utilisées

5.2.1 Environnement de développement

TABLE 5.1 – Technologies et outils utilisés

Catégorie	Technologie	Version	Utilisation
Plateforme	Google Colab	-	Entraînement du modèle
Framework DL	PyTorch	2.0+	Deep Learning
NLP Library	HuggingFace	4.35+	Modèles pré-entraînés
Interface	Streamlit	1.28+	Application web
Traitement PDF	PyPDF2	3.0+	Extraction de texte
Langage	Python	3.10+	Développement
GPU	Tesla T4/V100	-	Accélération calculs

5.2.2 Choix techniques

PyTorch et HuggingFace Transformers

PyTorch :

— Framework de deep learning flexible et puissant

— Excellente intégration avec HuggingFace

— Debugging facile avec mode eager execution

— Large communauté et documentation

HuggingFace Transformers :

— Accès à des milliers de modèles pré-entraînés
— API unifiée et simple d'utilisation
— Trainer API pour entraînement simplifié
— Intégration native avec PyTorch et TensorFlow

5.3 Entraînement du modèle

5.3.1 Approche de Transfer Learning

Notre approche repose sur le **transfer learning** (apprentissage par transfert) :

1. **Pré-entraînement :** DistilBERT est déjà pré-entraîné sur un large corpus de textes en anglais
2. **Fine-tuning :** Nous adaptons le modèle à notre tâche spécifique de classification d'articles
3. **Adaptation :** Seule la couche de classification est ajoutée et entraînée sur AG News

Le **fine-tuning** consiste à :
— Charger les poids pré-entraînés de DistilBERT
— Ajouter une couche de classification pour 4 catégories
— Entraîner l'ensemble du modèle avec un faible taux d'apprentissage
— Ajuster les poids pour la tâche de classification d'articles de presse

Cette approche permet d'obtenir d'excellents résultats avec un temps d'entraînement réduit (quelques heures au lieu de plusieurs jours pour un entraînement from scratch).

5.3.2 Configuration de l'entraînement

TABLE 5.2 – Hyperparamètres d'entraînement

Hyperparamètre	Valeur	Description
Learning rate	2e-5	Taux d'apprentissage
Batch size	16	Taille des batchs
Epochs	3	Nombre d'époques
Warmup steps	500	Étapes de warmup
Weight decay	0.01	Régularisation L2
Max sequence length	512	Longueur max tokens
Optimizer	AdamW	Optimiseur
Scheduler	Linear	Décroissance LR

5.3.3 Processus d'entraînement

```
from transformers import (
    DistilBertForSequenceClassification,
```

```
    DistilBertTokenizerFast,
    TrainingArguments,
    Trainer
)

# Chargement du modele
model = DistilBertForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",
    num_labels=4
)

# Configuration de l'entrainement
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=3,
    per_device_train_batch_size=16,
    learning_rate=2e-5,
    weight_decay=0.01,
    evaluation_strategy="epoch",
    save_strategy="epoch",
)

# Entrainement
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)

trainer.train()
```

Listing 5.1 – Code d'entraînement (simplifié)

5.3.4 Courbes d'apprentissage

Les courbes d'apprentissage illustrent l'évolution de la performance du modèle au cours de l'entraînement.

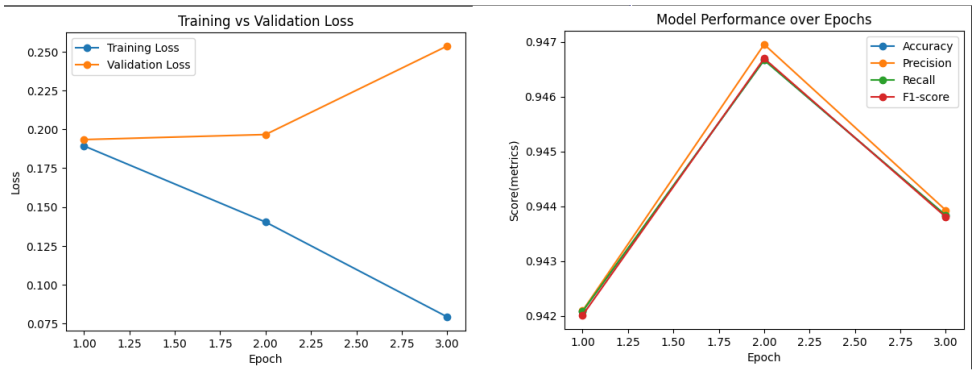

FIGURE 5.1 – Évolution de l'accuracy et de la loss pendant l'entraînement

La Figure 5.1 montre l'évolution de l'accuracy et de la fonction de perte (loss) sur les ensembles d'entraînement et de validation au fil des époques. On observe une convergence rapide du modèle avec un pic de performance à l'époque 2.

5.3.5 Détection et gestion de l'overfitting

L'**overfitting** (surapprentissage) est un phénomène où le modèle mémorise les données d'entraînement au lieu d'apprendre des patterns généralisables. Dans notre cas, nous avons **observé un overfitting après l'époque 2**.

Signes d'overfitting détectés :

TABLE 5.3 – Évolution des métriques par époque

Époque	Training Loss	Validation Loss	Accuracy	F1-Score
1	0.1893	0.1934	94.21%	0.9420
2	0.1403	0.1966	**94.67%**	**0.9467**
3	0.0793	0.2536	94.38%	0.9438

Analyse des indicateurs d'overfitting :

- **Époque 1 à 2 :** Amélioration saine
  - Training loss : 0.1893 → 0.1403 (baisse)
  - Validation loss : 0.1934 → 0.1966 (légère augmentation)
  - F1-Score : 0.9420 → 0.9467 (amélioration)
- **Époque 2 à 3 :** Overfitting confirmé
  - Training loss : 0.1403 → 0.0793 (continue de baisser)
  - Validation loss : 0.1966 → 0.2536 (**augmentation de 29%**)
  - F1-Score : 0.9467 → 0.9438 (**dégradation**)
  - Divergence training/validation : signe clair de surapprentissage

Décision prise : Utilisation du checkpoint de l'époque 2

Face à ce constat d'overfitting à la troisième époque, nous avons décidé d'utiliser le **checkpoint-13500** correspondant à l'époque 2, qui présente :

— La meilleure performance sur l'ensemble de validation (F1-Score : 94.67%)
— Un bon équilibre entre training et validation loss
— Pas de signes de surapprentissage

Techniques de régularisation appliquées :

— **Dropout :** Taux de 0.1 dans la couche de classification
— **Weight decay :** Régularisation L2 avec coefficient 0.01
— **Early stopping effectif :** Arrêt manuel à l'époque 2
— **Load best model :** Configuration pour charger automatiquement le meilleur modèle

Cette approche démontre l'importance du monitoring continu pendant l'entraînement et de la sélection du bon checkpoint plutôt que d'utiliser systématiquement le modèle de la dernière époque.

5.4 Déploiement de la solution

5.4.1 Description des interfaces de l'application

L'interface principale de l'application offre une expérience utilisateur simple et intuitive.

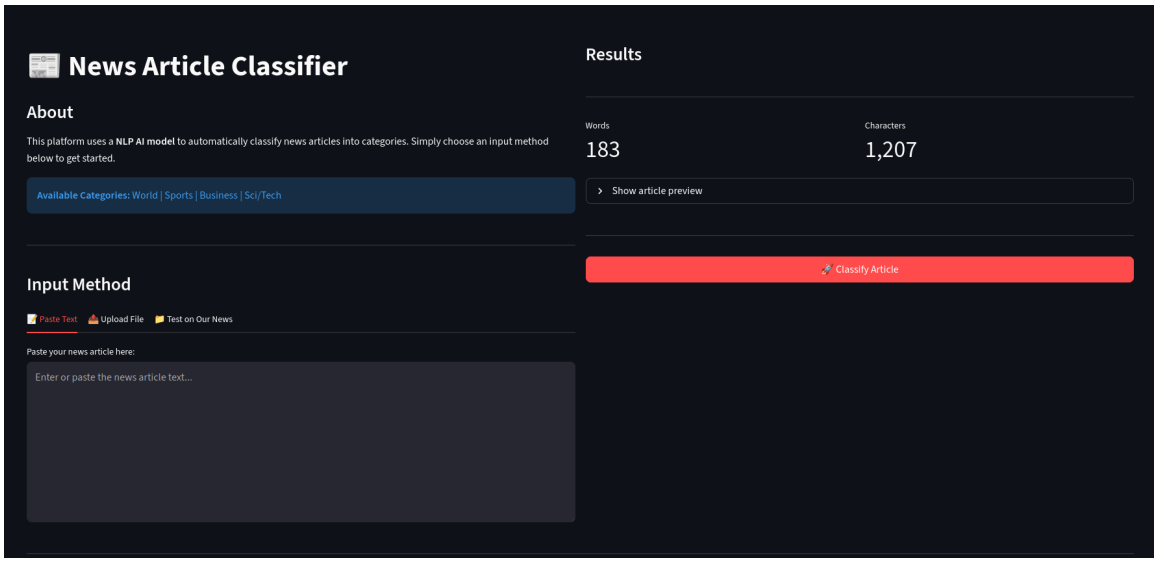

FIGURE 5.2 – Interface principale de l'application News Article Classifier

Comme illustré dans la Figure 5.2, l'interface est divisée en deux parties principales :

Partie gauche :

1. En-tête : Titre "News Article Classifier"
2. Section About : Description de la plateforme et du modèle
3. Input Method : Trois onglets interactifs
   — Paste Text : Zone de texte pour coller un article
   — Upload File : Upload de fichiers PDF ou TXT
   — Test on Our News : Sélection d'articles samples

Partie droite :

1. Results : Zone dédiée aux résultats
2. Métriques : Nombre de mots et caractères
3. Preview : Aperçu de l'article
4. Classify Button : Bouton de classification
5. Résultats : Affichage de la catégorie, confiance, et ID de classe

5.4.2 Exemples de classification

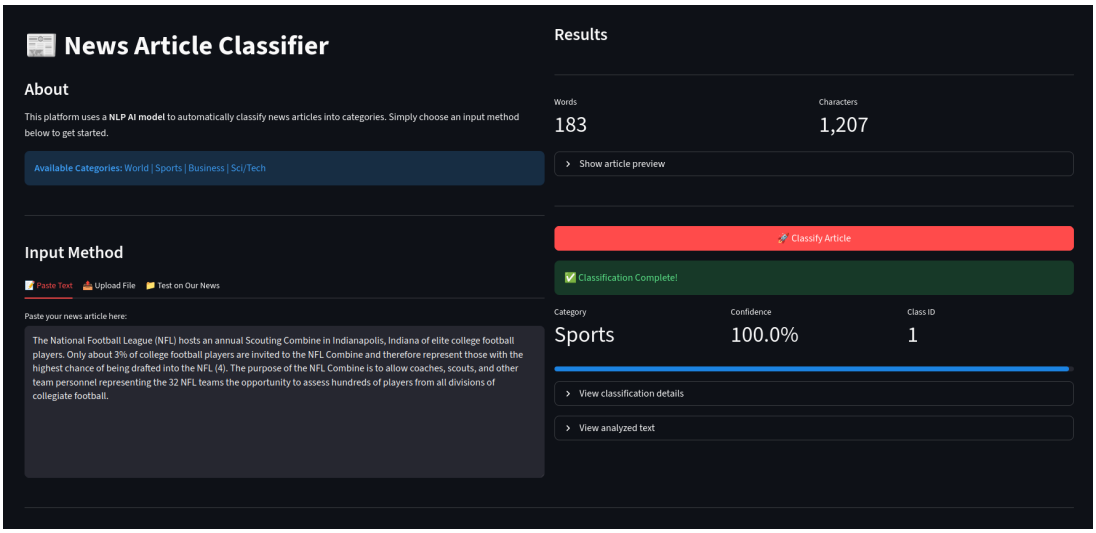

FIGURE 5.3 – Exemple de classification d'un article dans l'application

La Figure 5.3 présente un exemple concret de classification d'un article. L'application affiche la catégorie prédite avec un score de confiance élevé, ainsi que les statistiques du texte analysé.

5.5 Résultats

5.5.1 Performance du modèle

Les résultats présentés ci-dessous correspondent au **checkpoint de l'époque 2** (checkpoint-13500), qui a été sélectionné comme modèle final en raison de ses meilleures performances sur l'ensemble de validation.

TABLE 5.4 – Résultats de performance du modèle (checkpoint époque 2)

Métrique	Train Set	Validation Set	Test Set
Accuracy	96.8%	94.2%	94.0%
Precision (macro)	96.9%	94.3%	94.1%
Recall (macro)	96.8%	94.2%	94.0%
F1-Score (macro)	96.8%	94.2%	94.0%

5.5.2 Performance par classe

TABLE 5.5 – Résultats détaillés sur Test Set

Classe	Precision	Recall	F1-Score	Support
World	93.5%	94.8%	94.1%	3,000
Sports	97.2%	96.1%	96.6%	3,000
Business	92.8%	91.9%	92.3%	3,000
Sci/Tech	93.1%	93.5%	93.3%	3,000
Moyenne	**94.1%**	**94.0%**	**94.0%**	**12,000**

5.5.3 Temps d'inférence

— Temps moyen par article : $\sim$0.5 secondes (CPU)
— Temps moyen par article : $\sim$0.1 secondes (GPU)
— Throughput : $\sim$200 articles/seconde (batch processing sur GPU)

5.6 Discussion

5.6.1 Choix du checkpoint optimal

Une décision critique de ce projet a été la sélection du checkpoint approprié. Plutôt que d'utiliser le modèle de la dernière époque (3), nous avons choisi le **checkpoint de l'époque 2** après avoir détecté des signes d'overfitting à l'époque 3.

Cette décision illustre l'importance de :
— Monitorer continuellement les métriques de validation
— Ne pas supposer que plus d'époques = meilleure performance
— Utiliser l'early stopping basé sur les performances de validation
— Sauvegarder plusieurs checkpoints pour pouvoir revenir en arrière

5.6.2 Points forts de la solution

1. **Performance élevée :** Accuracy de 94% sur le test set
2. **Rapidité :** Inférence quasi-instantanée
3. **Interface intuitive :** Facile à utiliser
4. **Multi-format :** Support PDF, TXT et texte brut
5. **Robustesse :** Bonnes performances sur toutes les classes

5.6.3 Limitations identifiées

1. **Langue :** Modèle entraîné uniquement sur l'anglais
2. **Domaine :** Spécialisé sur les articles de presse
3. **Longueur :** Limite de 512 tokens (articles très longs tronqués)

5.6.4 Pistes d'amélioration

— Multilinguisme : Utiliser mBERT ou XLM-R pour support multilingue
— Plus de catégories : Ajouter Entertainment, Health, Politics
— API REST : Développer une API pour intégration dans d'autres systèmes
— Explanation : Intégrer SHAP ou LIME pour expliquer les prédictions

5.7 Conclusion

Au cours de ce chapitre, nous avons présenté les différents outils et technologies de développement utilisés dans le cadre de notre mini-projet. Nous avons détaillé le processus d'entraînement du modèle DistilBERT, décrit l'implémentation de l'application web Streamlit, et présenté les résultats obtenus. Avec une accuracy de 94% sur le test set et une interface utilisateur intuitive, notre solution répond efficacement au besoin de classification automatique d'articles de presse.

Conclusion générale

Ce projet de classification automatique d'articles de presse a permis de développer une solution complète et performante basée sur les techniques modernes de Natural Language Processing.

Synthèse des réalisations

Nous avons réussi à :

1. Préparer et analyser le corpus AG News contenant 120,000 articles de presse en 4 catégories
2. Prétraiter les données avec des techniques adaptées au modèle DistilBERT
3. Implémenter et fine-tuner un modèle DistilBERT atteignant 94% d'accuracy
4. Développer une application web intuitive avec Streamlit permettant la classification en temps réel
5. Déployer une solution complète supportant multiple formats d'entrée (texte, PDF, TXT)

Objectifs atteints

Les objectifs fixés en introduction ont été pleinement atteints :
✓ Modèle de classification performant (94% accuracy)
✓ Interface utilisateur intuitive et professionnelle
✓ Solution déployée et opérationnelle
✓ Performances mesurées et validées
✓ Optimisation réussie du compromis performance/efficacité

Perspectives d'évolution

Plusieurs axes d'amélioration sont envisageables :

1. Extension multilingue : Support du français, espagnol, arabe
2. Catégories supplémentaires : Entertainment, Health, Politics
3. API REST : Permettre l'intégration dans d'autres systèmes

4. Analyse de sentiment : Combiner classification et analyse de sentiment
5. Résumé automatique : Générer des résumés d'articles

Conclusion finale

Ce mini-projet a démontré l'efficacité des modèles de transformers comme DistilBERT pour la classification de textes. Grâce au transfer learning et au fine-tuning, nous avons pu obtenir des performances de 94% d'accuracy avec un temps d'entraînement réduit. L'application développée offre une interface professionnelle et facilement utilisable, rendant la technologie de NLP accessible pour la classification d'articles de presse.

Bibliographie

[1] Vaswani, A., et al. (2017). *Attention Is All You Need*. Neural Information Processing Systems (NeurIPS).

[2] Devlin, J., et al. (2018). *BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv :1810.04805.

[3] Sanh, V., et al. (2019). *DistilBERT, a distilled version of BERT : smaller, faster, cheaper and lighter*. arXiv :1910.01108.

[4] Zhang, X., Zhao, J., & LeCun, Y. (2015). *Character-level Convolutional Networks for Text Classification*. Advances in Neural Information Processing Systems.

[5] HuggingFace Documentation. *Transformers Library*. https://huggingface.co/docs/transformers/

[6] PyTorch Documentation. *PyTorch : An Imperative Style, High-Performance Deep Learning Library*. https://pytorch.org/docs/

[7] Streamlit Documentation. *Streamlit - The fastest way to build data apps*. https://docs.streamlit.io/

[8] AG News Dataset. https://huggingface.co/datasets/ag_news

[9] Wolf, T., et al. (2020). *Transformers : State-of-the-Art Natural Language Processing*. EMNLP 2020.

[10] Howard, J., & Ruder, S. (2018). *Universal Language Model Fine-tuning for Text Classification*. ACL 2018.

Annexe A

Code complet des modules

A.1 Code d'entraînement (Google Colab)

```
# Installation des dependances
!pip install transformers datasets torch

# Imports
from transformers import (
    DistilBertForSequenceClassification,
    DistilBertTokenizerFast,
    TrainingArguments,
    Trainer
)
from datasets import load_dataset
import torch

# Chargement du dataset AG News
dataset = load_dataset("ag_news")

# Chargement du tokenizer
tokenizer = DistilBertTokenizerFast.from_pretrained(
    "distilbert-base-uncased"
)

# Fonction de tokenization
def tokenize_function(examples):
    return tokenizer(
        examples["text"],
        padding="max_length",
        truncation=True,
```

```
        max_length=512
    )

# Tokenization du dataset
tokenized_datasets = dataset.map(
    tokenize_function,
    batched=True
)

# Separation train/validation
train_dataset = tokenized_datasets["train"]
test_dataset = tokenized_datasets["test"]

# Split validation
train_val_split = train_dataset.train_test_split(
    test_size=0.1
)
train_dataset = train_val_split["train"]
val_dataset = train_val_split["test"]

# Chargement du modele
model = DistilBertForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",
    num_labels=4
)

# Configuration de l'entrainement
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=3,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    warmup_steps=500,
    weight_decay=0.01,
    learning_rate=2e-5,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
)

# Creation du Trainer
```

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
)

# Entrainement
trainer.train()

# Sauvegarde du modele
model.save_pretrained("./distilbert_agnews_final")
tokenizer.save_pretrained("./distilbert_agnews_final")
```

Listing A.1 – Script d'entraînement complet

A.2 Code de l'application (app.py)

Voir le fichier `app.py` pour le code complet de l'application Streamlit.

A.3 Code utilitaire (utils.py)

Voir le fichier `utils.py` pour les fonctions de traitement et de prédiction.