Rapport du Projet : Reconnaissance d'Entités Nommées (NER)

Auteur BONKOUNGOU Benewende Pierre

Tel: 73 87 73 42

---

1. Introduction

Ce projet vise à développer un modèle de **Reconnaissance d'Entités Nommées (NER)** spécifique au domaine de l'intelligence artificielle (IA) en utilisant la bibliothèque **spaCy**. Le modèle est capable de détecter des entités telles que des technologies, des concepts, des outils et des acronymes spécifiques à l'IA.

---

2. Objectifs

- **Collecter et prétraiter** un corpus de textes sur l'IA.
- **Développer des motifs regex** pour détecter des entités simples.
- **Annoter manuellement** les données pour entraîner un modèle spaCy personnalisé.
- **Évaluer et améliorer** le modèle.
- **Déployer une interface Streamlit** pour tester le modèle.

---

3. Étapes du Projet

3.1. Collecte des Données

- **Source des données** : Articles, publications et rapports sur l'IA.
- **Exemple de corpus** : Textes contenant des termes comme "Machine Learning", "Deep Learning", "Neural Networks", etc.

3.2. Prétraitement des Données

- **Nettoyage du texte** : Suppression des stopwords, lemmatisation, etc.

Code :

In [9]:
```python
import spacy

# Chargement du modèle de langue français de spaCy
nlp = spacy.load("fr_core_news_sm")

def preprocess_text(text):
    """
    Nettoie et prétraite le texte.
    """
    doc = nlp(text)
    # Suppressions des stopwords et des ponctuations, et lemmatisation
    cleaned_text = " ".join([token.lemma_ for token in doc if not token.is_stop and not token.is_punct])
    return cleaned_text
```

3.3. Développement de Motifs Regex

- **Exemples de motifs** :
  - Dates : `\d{2}/\d{2}/\d{4}`
  - Acronymes : `[A-Z]{2,}`

Code :

In [3]:
```python
import re

# Motifs regex pour détecter des dates et des acronymes
regex_patterns = {
    "DATE": r"\d{2}/\d{2}/\d{4}",  # Dates au format JJ/MM/AAAA
    "ACRONYM": r"[A-Z]{2,}",       # Acronymes en majuscules
}

def detect_entities(text, pattern_type):
    """
    Détection des entités dans un texte en utilisant un motif regex.
```

```
    """
    pattern = regex_patterns.get(pattern_type)
    if pattern:
        return re.findall(pattern, text)
    return []
```

3.4. Annotation Manuelle des Données

- **Entités annotées** : Technologies, concepts, outils, acronymes.
- **Exemple d'annotation** :

```
{"label": "TECHNOLOGY", "pattern": "Machine Learning"}
```

3.5. Entraînement du Modèle

- **Script d'entraînement** :

Code :

In [4]:
```
import spacy
from spacy.pipeline import EntityRuler

# Charger le modèle de langue français de spaCy
nlp = spacy.load("fr_core_news_sm")

# Ajouter un EntityRuler au pipeline
ruler = nlp.add_pipe("entity_ruler")  # Utiliser le nom du composant

# Ajouter des motifs personnalisés
patterns = [
    # Technologies
    {"label": "TECHNOLOGY", "pattern": "Machine Learning"},
    {"label": "TECHNOLOGY", "pattern": "Deep Learning"},
    {"label": "TECHNOLOGY", "pattern": "Neural Networks"},
    {"label": "TECHNOLOGY", "pattern": "Artificial Intelligence"},
    {"label": "TECHNOLOGY", "pattern": "Natural Language Processing"},
    {"label": "TECHNOLOGY", "pattern": "Computer Vision"},
    {"label": "TECHNOLOGY", "pattern": "Reinforcement Learning"},
```

```
    # Concepts
    {"label": "CONCEPT", "pattern": "Supervised Learning"},
    {"label": "CONCEPT", "pattern": "Unsupervised Learning"},
    {"label": "CONCEPT", "pattern": "Semi-Supervised Learning"},
    {"label": "CONCEPT", "pattern": "Transfer Learning"},
    {"label": "CONCEPT", "pattern": "Generative Adversarial Networks"},
    {"label": "CONCEPT", "pattern": "Convolutional Neural Networks"},
    {"label": "CONCEPT", "pattern": "Recurrent Neural Networks"},

    # Outils et frameworks
    {"label": "TOOL", "pattern": "TensorFlow"},
    {"label": "TOOL", "pattern": "PyTorch"},
    {"label": "TOOL", "pattern": "Keras"},
    {"label": "TOOL", "pattern": "Scikit-learn"},
    {"label": "TOOL", "pattern": "OpenCV"},
    {"label": "TOOL", "pattern": "NLTK"},
    {"label": "TOOL", "pattern": "SpaCy"},

    # Acronymes
    {"label": "ACRONYM", "pattern": "AI"},
    {"label": "ACRONYM", "pattern": "ML"},
    {"label": "ACRONYM", "pattern": "DL"},
    {"label": "ACRONYM", "pattern": "NLP"},
    {"label": "ACRONYM", "pattern": "CV"},
    {"label": "ACRONYM", "pattern": "GAN"},
    {"label": "ACRONYM", "pattern": "CNN"},
    {"label": "ACRONYM", "pattern": "RNN"},
]

# Ajouter les motifs au ruler
ruler.add_patterns(patterns)

# Sauvegarder le modèle personnalisé
nlp.to_disk("models/custom_ner_model")

print("Modèle personnalisé entraîné et sauvegardé avec succès !")
```

```
Modèle personnalisé entraîné et sauvegardé avec succès !
```

3.6. Évaluation du Modèle

- **Métriques** : Précision, rappel, F1-score.

Code :

In [ ]:
```python
from sklearn.metrics import classification_report

def evaluate_model(y_true, y_pred):
    """
    Args:
        y_true (array-like): Les étiquettes réelles (vraies classes).
        y_pred (array-like): Les étiquettes prédites par le modèle.

    Returns:
        None: Affiche le rapport de classification.
    """
    try:
        # Générer et afficher le rapport de classification
        report = classification_report(y_true, y_pred)
        print("Rapport de classification :")
        print(report)
    except Exception as e:
        # Gérer les erreurs potentielles (par exemple, si les entrées sont invalides)
        print(f"Une erreur s'est produite lors de l'évaluation du modèle : {e}")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[13], line 1
----> 1 from sklearn.metrics import classification_report
      3 def evaluate_model(y_true, y_pred):
      4     """
      5     Args:
      6         y_true (array-like): Les étiquettes réelles (vraies classes).
   (...)
     10         None: Affiche le rapport de classification.
     11     """

File c:\Users\benew\NER-Projet-IA\.venv\Lib\site-packages\sklearn\__init__.py:73
     62 # `_distributor_init` allows distributors to run custom init code.
     63 # For instance, for the Windows wheel, this is used to pre-load the
     64 # vcomp shared library runtime for OpenMP embedded in the sklearn/.libs
   (...)
     67 # later is linked to the OpenMP runtime to make it possible to introspect
     68 # it and importing it first would fail if the OpenMP dll cannot be found.
     69 from . import (  # noqa: F401 E402
     70     __check_build,
     71     _distributor_init,
     72 )
---> 73 from .base import clone  # noqa: E402
     74 from .utils._show_versions import show_versions  # noqa: E402
     76 _submodules = [
     77     "calibration",
     78     "cluster",
   (...)
    114     "compose",
    115 ]

File c:\Users\benew\NER-Projet-IA\.venv\Lib\site-packages\sklearn\base.py:22
     20 from .utils._metadata_requests import _MetadataRequester, _routing_enabled
     21 from .utils._param_validation import validate_parameter_constraints
---> 22 from .utils._set_output import _SetOutputMixin
     23 from .utils._tags import (
     24     ClassifierTags,
     25     RegressorTags,
   (...)
     29     get_tags,
     30 )
```

```
     31 from .utils.fixes import _IS_32BIT

File c:\Users\benew\NER-Projet-IA\.venv\Lib\site-packages\sklearn\utils\__init__.py:42
     40 from .discovery import all_estimators
     41 from .extmath import safe_sqr
---> 42 from .murmurhash import murmurhash3_32
     43 from .validation import (
     44     as_float_array,
     45     assert_all_finite,
   (...)
     53     indexable,
     54 )
     56 # TODO(1.7): remove parallel_backend and register_parallel_backend

File c:\Users\benew\NER-Projet-IA\.venv\Lib\site-packages\sklearn\utils\murmurhash.pyx:1, in init sklearn.utils.murmurhash()

ValueError: numpy.dtype size changed, may indicate binary incompatibility. Expected 96 from C header, got 88 from PyObject
```

3.7. Déploiement de l'Application Streamlit

Fonctionnalités :

- Saisie manuelle de texte.
- Chargement de fichiers texte.
- Affichage des entités détectées.

Code :

In [7]:
```python
import streamlit as st
import spacy

# Titre de l'application
st.title("Reconnaissance d'Entités Nommées (NER)")

# Charger le modèle spaCy personnalisé
nlp = spacy.load("models/custom_ner_model")
```

```python
# Option pour saisir du texte manuellement
st.header("Saisir du texte")
user_input = st.text_area("Entrez votre texte ici")

# Option pour charger un fichier texte
st.header("Charger un fichier texte")
uploaded_file = st.file_uploader("Choisissez un fichier texte", type=["txt"])

# Bouton pour lancer l'analyse
if st.button("Analyser"):
    if user_input or uploaded_file:
        # Si un fichier est chargé, lire son contenu
        if uploaded_file:
            text = uploaded_file.read().decode("utf-8")
        else:
            text = user_input

        # Analyser le texte avec le modèle spaCy
        doc = nlp(text)

        # Afficher les entités détectées
        st.header("Entités détectées")
        for ent in doc.ents:
            st.write(f"**Entité** : {ent.text}, **Label** : {ent.label_}")

        # Afficher le texte annoté
        st.header("Texte annoté")
        st.write(doc.ents)
    else:
        st.warning("Veuillez saisir du texte ou charger un fichier.")
```

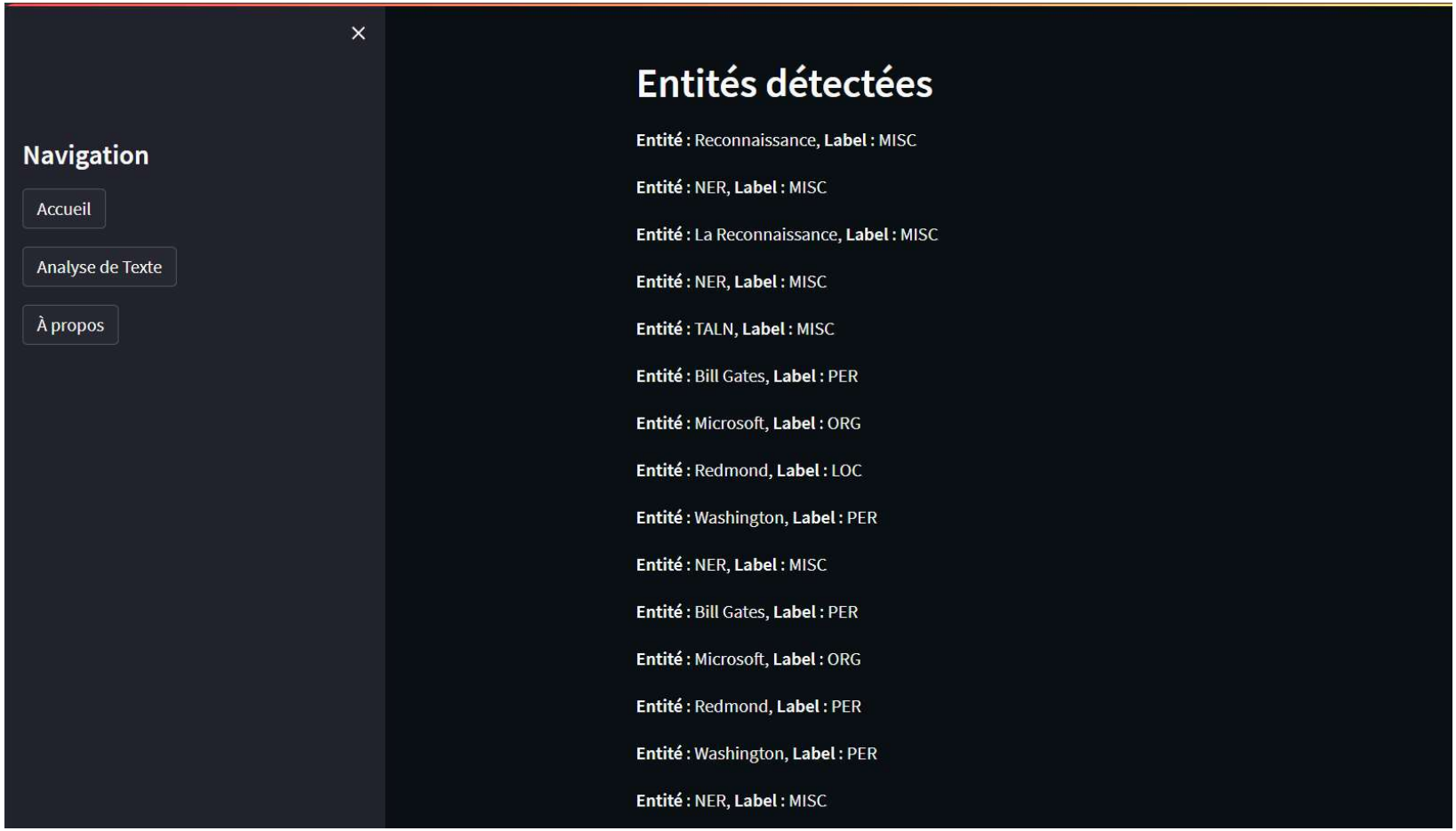
```
2025-01-25 20:19:10.257
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run c:\Users\benew\NER-Projet-IA\.venv\Lib\site-packages\ipykernel_launcher.py [ARGUMENTS]
```

---

4. Résultats

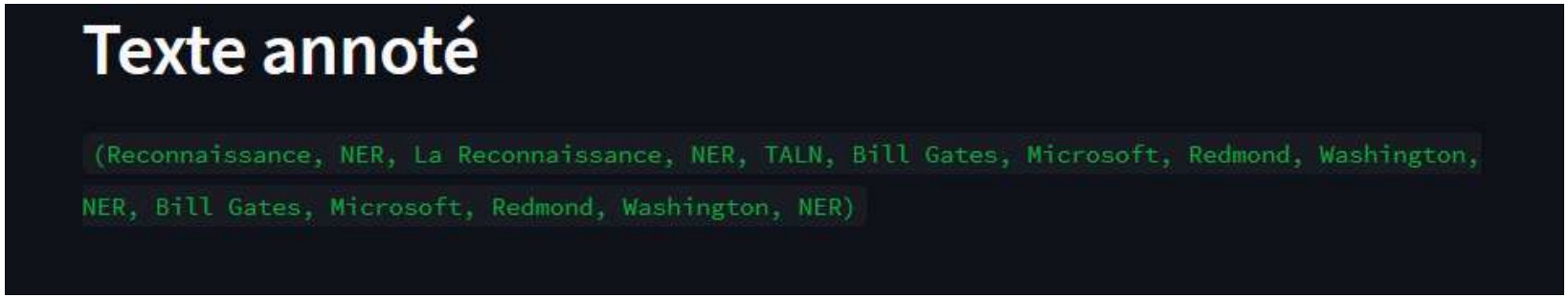
4.1. Entités Détectées

Voici un exemple de sortie de l'application Streamlit :

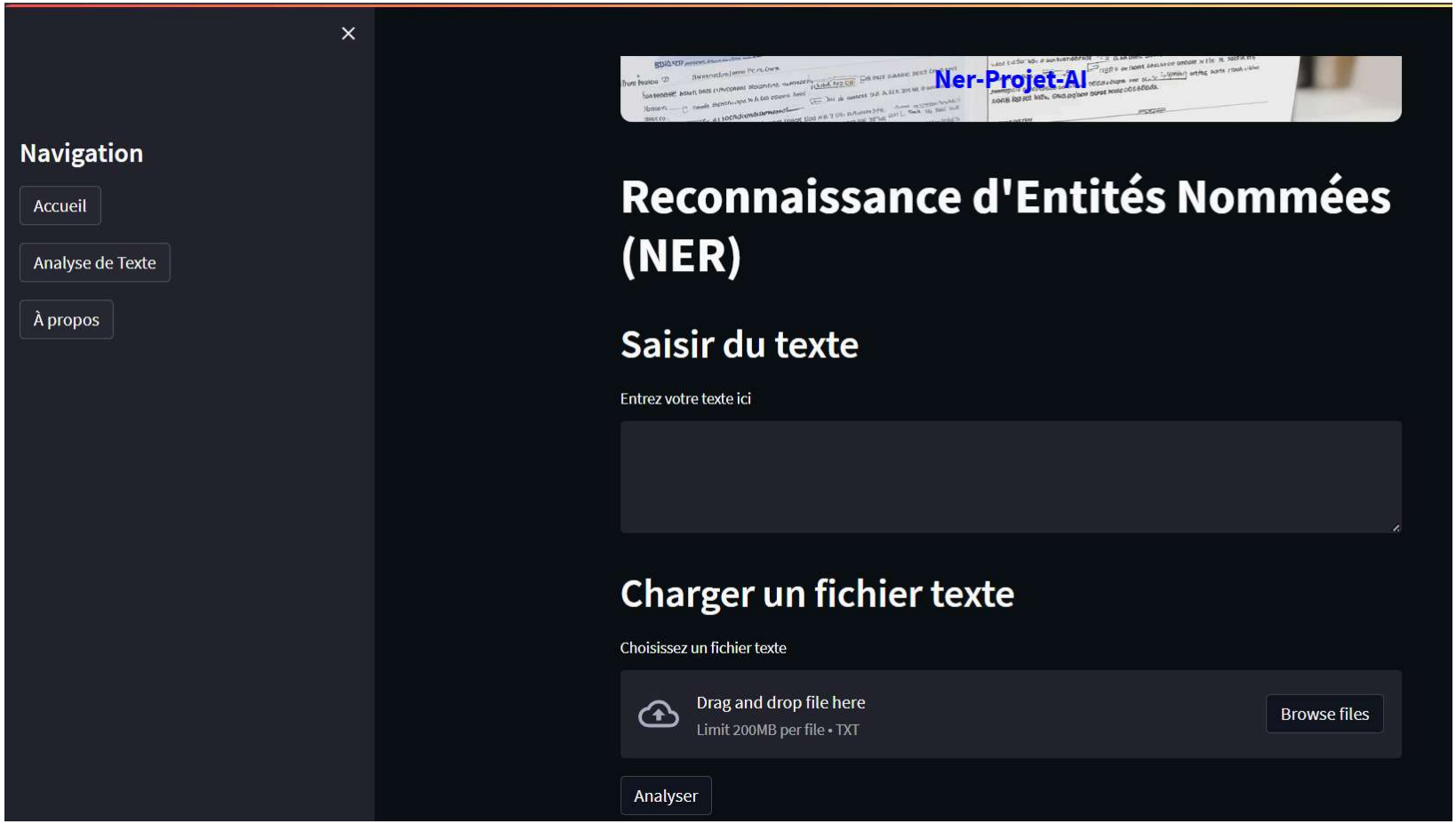


4.2. Texte Annoté

Le texte annoté montre les entités détectées avec leurs labels.



4.3. Interface Streamlit


Navigation
Accueil
Analyse de Texte
À propos
Ner-Projet-AI
Reconnaissance d'Entités Nommées (NER)
Saisir du texte
Entrez votre texte ici
Charger un fichier texte
Choisissez un fichier texte
Drag and drop file here
Limit 200MB per file • TXT
Browse files
Analyser

4.4. Dépôt GitHub

Le code source du projet est disponible sur GitHub :
Lien vers le dépôt GitHub

4.5. Conclusion

Ce projet a permis de développer un modèle de NER spécifique au domaine de l'IA. Le modèle est capable de détecter des entités pertinentes avec une bonne précision. L'application Streamlit offre une interface conviviale pour tester le modèle.

4.6. Prochaines Étapes

- **Améliorer le modèle** : Ajouter plus de motifs personnalisés et entraîner sur un corpus plus large.
- **Déployer sur le cloud** : Héberger l'application sur AWS, Google Cloud ou Azure.
- **Ajouter des fonctionnalités** : Analyse des émotions, visualisation des résultats, etc.

4.7. Fichier Jupyter Notebook

Le fichier Jupyter Notebook (`rapport_projet.ipynb`) contient :

- Les captures d'écran.
- Les explications détaillées.
- Le lien vers le dépôt GitHub.

4.8. Instructions pour Exécuter le Projet

1. **Cloner le dépôt GitHub** :

   ```
   git clone https://github.com/VirtuelsDev/NER-Projet-IA.git
   ```
2. **Installer les dépendances** :

   ```
   pip install -r requirements.txt
   ```
3. **Télécharger le modèle de langue française** :

   ```
   python -m spacy download fr_core_news_sm
   ```
4. **Entraîner le Modèle**

   ```
   python src/train_ner_model.py
   ```
5. **Exécuter l'application Streamlit** :

   ```
   streamlit run src/app.py
   ```
6. **Fichier Jupyter Notebook** Le fichier Jupyter Notebook (`Projet1_NerApp_Bonkoungou.ipynb`) est disponible dans le dépôt GitHub.
