M2SOL034 Corpus, ressources et linguistique outillée

TD1 : Introduction aux Corpus et à l'IA

Wiem TAKROUNI

Sorbonne Université

Master « Langue et Informatique »

UFR Sociologie et Informatique pour les Sciences Humaines

Semestre 2, 2024-2025, le 21 mars 2025

Objectifs du TD

- Comprendre la notion de corpus et son rôle en IA
- Explorer différents types de corpus
- Effectuer des premières manipulations sur un corpus textuel
- Analyser l'impact des données sur les modèles d'IA

Exercice 1 : Identifier et Classifier des Corpus

Consigne : Pour chaque corpus donné ci-dessous, indiquez :

1. **Son type** (textuel, oral, multilingue, annoté…)
2. **Son utilisation potentielle en IA**
3. **Un modèle d'IA qui pourrait l'exploiter**

- **Corpus 1 : Common Crawl**
- **Corpus 2 : LibriSpeech**
- **Corpus 3 : CoNLL-2003**

Exemple:

Corpus : Wikipedia en français

☑ **Type** : Corpus textuel, non annoté

☑ **Utilisation** : Pré-entraîner un modèle de langage

☑ **Modèle exploitable** : GPT, BERT

Exercice 2 : Extraire les représentations de texte avec BERT

Consigne :

- Charger un modèle **BERT pré-entraîné**
- Appliquer BERT sur un corpus pour extraire les **vecteurs de mots (embeddings)**
- Observer les représentations générées

◆ Installation des bibliothèques

```
pip install transformers torch
```

- Utilisation de BERT pour générer des embeddings

```
import torch
from transformers import BertTokenizer, BertModel

# Charger le modèle BERT et son tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
model = BertModel.from_pretrained("bert-base-uncased")

# Exemple de texte (on peut remplacer par un corpus)
texte = "Les corpus sont essentiels pour entraîner des modèles d'intelligence artificielle."

# Tokenisation du texte
tokens = tokenizer(texte, return_tensors="pt", padding=True, truncation=True)

# Passage du texte dans BERT
with torch.no_grad():  # Désactive le calcul du gradient pour économiser les ressources
    output = model(**tokens)

# Extraction des embeddings du dernier niveau de BERT
embeddings = output.last_hidden_state

print(f" 🔹 Nombre de tokens : {tokens.input_ids.shape[1]}")
print(f" 🔹 Taille des embeddings : {embeddings.shape}")  # (1, nombre de tokens, 768)
```

- Interprétation des résultats
  - tokens.input_ids.shape[1] : Nombre de mots (tokens) après tokenisation.
  - embeddings.shape : Taille des vecteurs générés

```
(1, nombre_de_tokens, 768)
```

- 1 → Une phrase traitée
- nombre_de_tokens → Nombre de mots (tokens) dans la phrase
- 768 → Dimension du vecteur de chaque mot