Projet :
Prédictions structurées pour le traitement automatique du langage naturel

Pre-Train Transformer
Versus
Home-Made Transformer

Présenté par Victor Tancrez, Paul Peyssard & Thomas Oxisoglou
Supervisé par Carlos Ramisch & Alexis Nasr

M2 IAAA
15/01/2024

Sommaire

Introduction

- Problématique : Les modèles pré-entraîné surperforment t-ils systématiquement les modèles spécifique à une tâche ?

- But : Créer des transformers et le comparer à des transformers pré-entraînés

- Comparaison effectuée sur la tâche de POS Tagging

Introduction

- Les données utilisées : Le dataset UD French Sequoia :
  - 70% Train, 15% Validation, 15% Test

- Le format : CoNLL-U

- Environnement utilisé : Entraînement sur GPU (colab et local)

Une Baseline LSTM

- Une simple baseline
- L'entraînement
- Les paramètres :

Model	Learning Rate	Batch Size	Epochs	Embedding_dim	Hidden_dim	Optimizer
baseline_LSTM	0.01	16	50	64	128	SGD

Table 1: Paramètres de la baseline

Transformer Pre-Train

- BERT :
  - Blocs d’encodeur des transformers empilés
  - Self-Attention bidirectionnelle
  - Attention multi-têtes
  - Tokenisation BPE (WordPiece)

- Deux modèles différent de cette architecture :
  - CamemBERT-base
  - Bert-base-multilingual-cased

CamemBERT-base

- Architecture du modèle :
  - Basé sur le modèle pré-train RoBERTa
  - Spécialisé sur le français
  - 111 millions de paramètres
  - Architecture BERT

- Entrainement du modèle :
  - Entraîné sur le OSCAR dataset

- Encodage des entrées :
  - Algorithme SentencePiece (Tokenisation en divisant les séquences de caractères en sous mots)

Source : https://arxiv.org/pdf/1911.03894.pdf

Bert-base-multilingual-cased

- Architecture du modèle :
  - 179 millions de paramètres
  - Architecture Bert

- Entrainement du modèle :
  - Entraîné sur les 104 langues les plus utilisées de wikipedia
  - Variante “cased” permettant de prendre en compte la casse des lettres (Majuscule/Minuscule)

- Encodage des entrées :
  - Algorithme WordPiece (Tokenisation en divisant les mots en sous ensemble de sous mots)

Source : https://arxiv.org/pdf/1810.04805.pdf

Transformers Pre-Train

- Les différents paramètres fine-tunés :
  - Learning Rate
  - Batch Size
  - Nombre d'époques
  - Optimizer
- Les paramètres finaux obtenus :

Model	Learning Rate	Batch Size	Epochs	Optimizer
Camembert-base	0.0001	32	6	Adam
Bert-base-multilingual-cased	0.0001	64	7	Adam

Table 2: Paramètres finaux des modèles pré-entrainés

Home-Made Transformer

-Embedding sur les mots ou convolution sur les caractères

-Positional encoding

-TransformerEncoder

-Couche dense : sortie en dimension

(nombre de mot,nombre de POS)

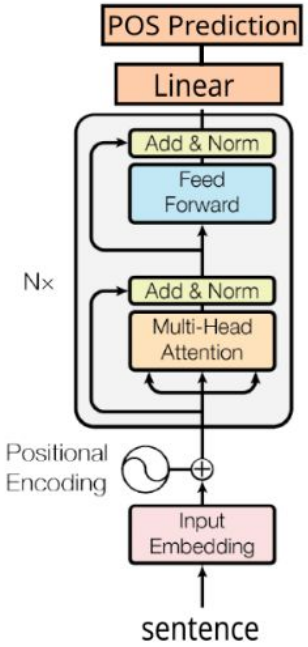


Figure 3: Principe général de notre architecture (Source : Documentation PyTorch, modifié pour notre implémentation).

Embedding sur les mots

-Mots représentés par des indices

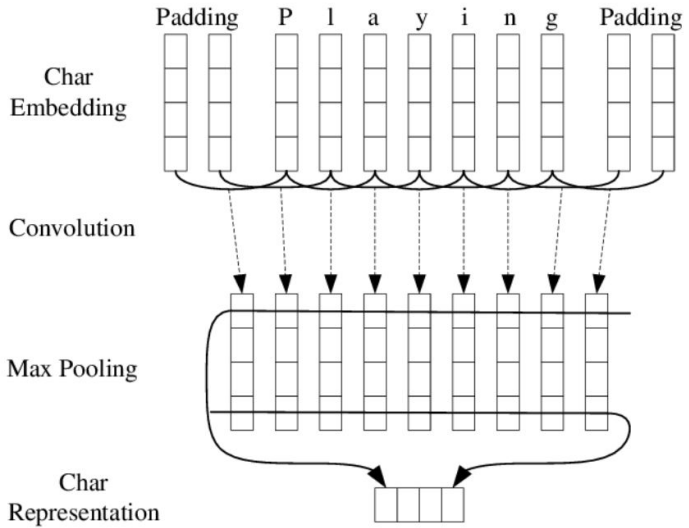
-Mots inconnus

-Calcul des embeddings lors de l’entraînement

Convolution sur les caractères

-Lettres représentées par des indices

-Lettres inconnues



Source: Ma & Hovy, 2016

Hyperparamètres du modèle

-Taille de l’embedding pour les caractères

-Nombre de filtres

-Nombre de têtes d’attention

-Taille du noyau de convolution

-Taille de la représentation des couches du transformer encoder

-Nombre de couche de transformer encoder

Etude des hyperparamètres du modèle

-Problème combinatoire

-Si 5 valeurs testées pour chaque hyperparamètre ⇒ 6^5 = 7776 modèles à entraîner

-Repérage des hyperparamètres importants

-Choix de faire varier 4 hyperparamètres en fixant les autres : taille du noyau de convolution, le nombre de têtes d'attention, le nombre de filtres et le nombre de couche de transformer encoder.

- Adapter les hyperparamètres fixés au fur et à mesure.

Etude des hyperparamètres du modèle

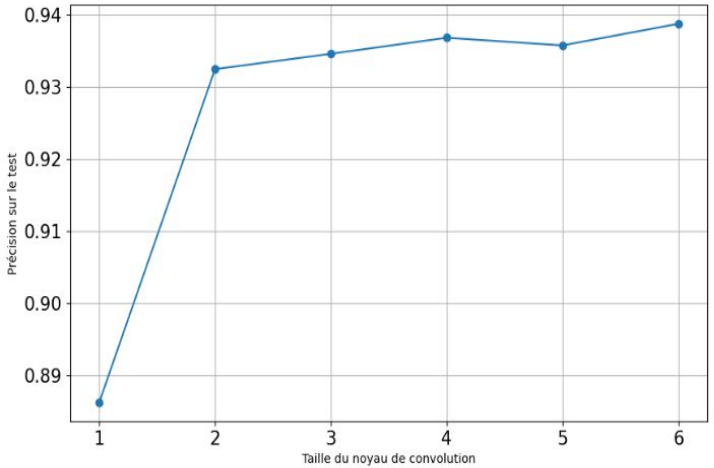


Figure 4: Précision sur le test en fonction de la taille du noyau de convolution

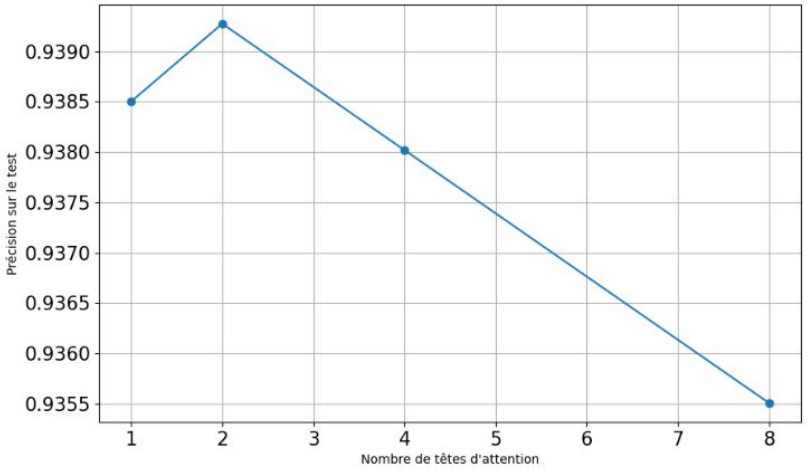


Figure 5: Précision sur le test en fonction du nombre de têtes d'attention avec une taille de noyau de convolution fixé à 6

Etude des hyperparamètres du modèle

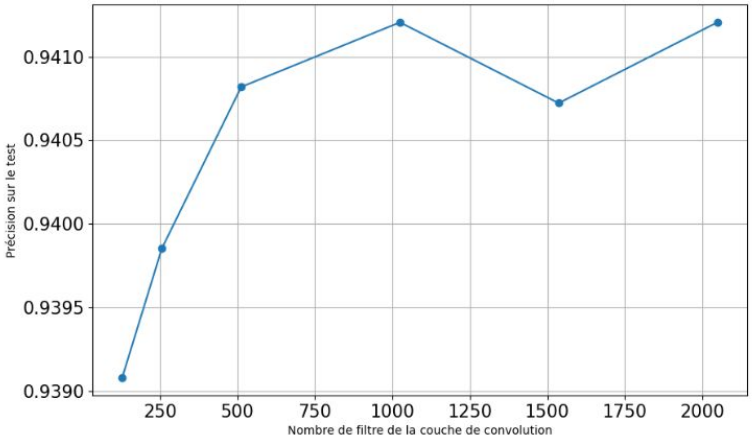


Figure 6: Précision sur le test en fonction du nombre de filtre de convolution

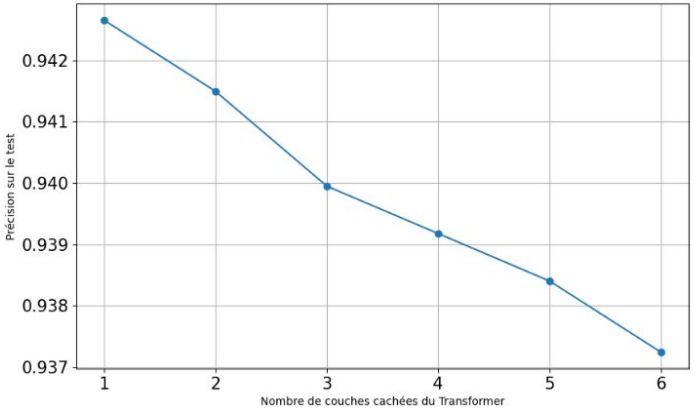


Figure 7: Précision sur le test en fonction du nombre de couche de Transformer avec un nombre de filtre à 1024

Résultats Principaux

- Choix de métrique : Accuracy et F1_Score
- Interprétation

Modèle	Train Accuracy	Test Accuracy	Test f1
base_line_LSTM	0.8869%	0.8664%	0.7892%
bert-base-multilingual-cased	0.9806%	0.9507%	0.8628%
camembert-base	0.9758%	0.9324%	0.8518%
home-made transformer encodage convolutif	0.9708%	0.9434%	0.8531%
home-made transformer embedding sur les mots	0.9678%	0.9110%	0.8217%

Table 3: Comparaison des performances des différents modèles

Bonus : dimension multi-tâches des modèles

- Une adaptation de nos modèles pour la tâche de lemmatisation
- Un choix d'hyperparamètres identiques à la tâche de POS-Tagging
- Axe d'amélioration possible

Modèle	Train Accuracy	Test Accuracy
base_line_LSTM	0.8379%	0.8580%
home-made transformer encodage convolutif	0.9947%	0.9722%

Table 4: Comparaison des performances des différents modèles pour prédiction de lemme

Conclusion

- Difficultés :
  - Temps d’entraînement long des transformers pré entraînés
  - Conception de transformers fait maison
- Ce que nous a apporté le projet :
  - Manipulation de texte
  - Familiarisation avec l'entraînement sur GPU
  - Travailler avec des modèles pré-entraîné (hugging face…)

Conclusion

- Travaux futurs et potentiel d’amélioration
  - Plus de données pour les modèles fait maison
  - Tester les performances sur différents datasets
  - Plus de fine tuning

Bibliographie et travaux liés

- BERT : https://arxiv.org/abs/1810.04805

- CamemBERT : https://www.aclweb.org/anthology/2020.acl-main.645

- Dataset Sequoia : https://universaldependencies.org/treebanks/fr_sequoia/index.html

- Autres liens utiles sur le rapport du projet

Pre-Train Transformer
Versus
Home-Made Transformer

Merci de votre attention !
(It’s all we needed)

Présenté par Victor Tancrez, Paul Peyssard & Thomas Oxisoglou
Supervisé par Carlos Ramisch & Alexis Nasr

M2 IAAA
15/01/2024