

Introduction sur le TAL/NLP

De la lexicométrie aux modèles pré-entraînés


Émilien Schultz

Natural Language Processing

Le texte : données non structurées

- Au début : une suite de caractères
 - Avec un encodage : binary symbole
- À la fin : extraire du sens
 - Proche de l'humain

Entre : comment faire ?

 **Une approche pratique plutôt que théorique**

De nombreux domaines concernés pour la théorie

NLP/TAL : une grande famille

Des techniques très diverses

- **Lexicométrie** (compter des mots)
 - un peu *old school*
- **Machine Learning** sur données textuelles vectorisées
- Évolution vers les **modèles de langage**
 - Approches potentiellement plus proche du langage naturel

Chacune a ses particularités/limites/coûts

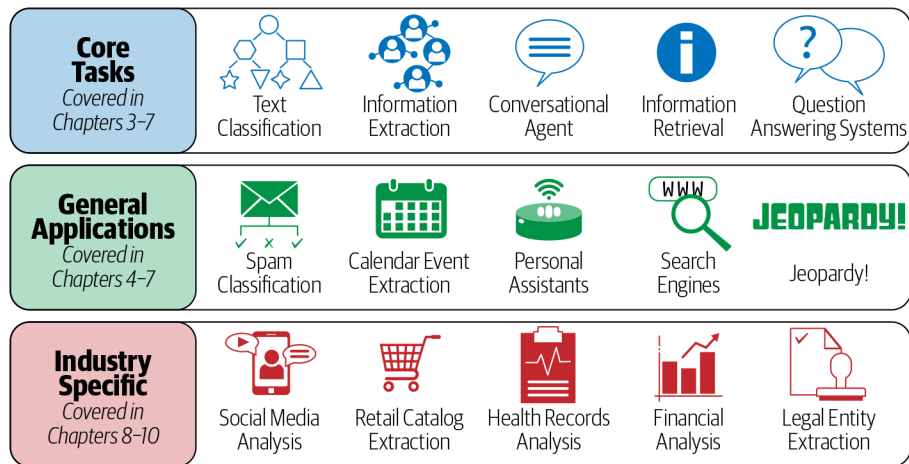
Des tâches différentes

- Structure du texte
 - Identifier le rôle d'un mot dans un texte (POS)
 - Classifier des textes
 - Faire des groupes
 - Retrouver des éléments
 - Identifier des éléments spécifiques
 - Name Entity Recognition (NER)
 - Générer des textes ...
-

Des tâches très interdisciplinaires



Liées à différentes applications



Point de départ : du texte

Texte Représentation numérique

- Déjà mis en forme (Articles, posts)
- Pas encore mis en forme (PDF, images)
 - Enjeux d’OCR, de spatialisation (frame)

Donc :

- Qualité différentes
 - Problèmes spécifiques
-

Données “non structurées”

- Différentes langues (mélangées)
- Des éléments supplémentaires (émoticones)
- Des erreurs (OCR)
- Quelle unité pertinente
 - Texte entier

- Paragraphe
 - Phrase
-

Première étape : représenter un texte

- Par la présence de certains mots
 - Approches par dictionnaires
 - Ou par motifs : expressions régulières
 - Par l'ensemble des mots
 - Approches par *sacs de mots*
 - Par encodage de la structure
 - Approches par plongement (embeddings)
 - Directement par un modèle génératif
-

Représenter : un pipeline fondamental

Plusieurs manières de faire (plus ou moins automatisée)

- fenêtrage
- tokenisation
- suppression des mots vides
- lemmatisation/stemmisation

Pour de nombreux besoins spécifiques, intéressant de maîtriser les manipulations de bas niveaux

La notion de tokenisation

- Passage en unités discrètes
 - **Tokenisation par mots** : “Je vais bien” → ["Je", "vais", "bien"]
 - **Sous-mots (Byte-Pair Encoding, WordPiece)** :
 - “inconnue” → ["in", "##con", "##nue"]
 - **Caractères** : chaque caractère est un token → ["J", "e", " ", "v", "..."]
 - Dépend du pipeline/conséquences importantes
-

Ensuite aller vers la tâche

1. Représentation du texte
2. Opération spécifique sur le texte
 - Avec des domaines spécifiques

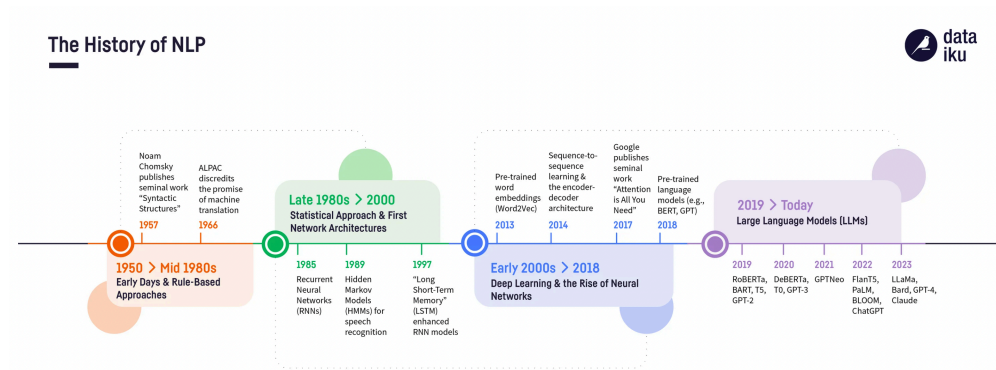
i Les modèles préentraînés

De + en + d'opérations mobilisant des modèles de machine learning pré-entraînés

Importance des modèles

- Outils pré-entraînés permettant la prédiction
 - Importance du corpus d'entraînement
 - Spécifiques à la langue / type de textes
 - Dépendent de plusieurs niveaux
 - Tokenisation
 - Corpus d'entraînement
 - Méthodes (RLHF)...
-

Une multitude de modèles



<https://blog.dataiku.com/nlp-metamorphosis>

En ce qui nous concerne

- Transformers
 - BERT (encoder only, 2018+)
 - * pour le français [CamemBert](#) ou [FlauBERT](#)
 - * Récemment, [ModernBERT](#)
- Depuis 2022, explosion des LLM
 - Dépasse le NLP (par exemple, Whisper)
 - [HuggingFace](#)

Modèles BERT

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

Abstract

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5 (7.7 point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

Anthology ID: N19-1423

Volume: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)

Month: June

Year: 2019

Address: Minneapolis, Minnesota

Editors: Jill Burstein, Christy Doran, Tamar Solorio

Venue: NAACL

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#)

Des modèles de toutes taille

Tout n'est pas un LLM

- Indicateur : nombre de paramètres
- Des modèles de grande taille
 - Certains nécessitent des GPU
 - Penser à utiliser des services dédiés si nécessaires

Quel intérêt des SLM ?

Faire du NLP !

Petit panorama de différentes tâches

Suivant les besoins, trouver la bonne tâche :

- rapidité
- robustesse
- efficacité
- ...

Et importance d'évaluer la qualité du traitement.

Faisons un petit tour des lieux

Les bibliothèques Python

- Avant, un peu périmée NLTK
 - Le plus pratique : **SpaCy**
 - Faire du ML avec **Scikit-learn**
 - Utiliser directement des modèles de HuggingFace avec **Transformers**
 - Ou des bibliothèques construites dessus ...
-

Cas 0 : Cibler le texte / les expressions régulières

- bibliothèque **re** ou **regex**
- un ensemble d'opérateurs pour construire des masques
 - `\w` pour un mot
 - `*` pour n'importe quel caractère...
- différentes fonctions pour chercher :
 - le premier
 - tous
 - remplacer

[Aller voir une cheatsheet](#)

Application

i Découper en mots

```
import re
mots = re.split(r"\W+", texte)
```

Cas 1 : Distribution de mots

- Mais qu'est-ce qu'un mot ?
 - Les enjeux de la tokenisation
- Compter les mots (utiliser `collections.Counter`)
- Comment gérer les variations des mots ?
 - La lemmatisation

Cas 2 : Document-Term Matrice (DTM)

Passer un texte en vecteur

- Prendre chaque document
- Réduire à un sac de mots
- Construire une matrice textes - mots
- Scikit-learn avec `CountVectorizer`