Transformers & Hugging Face

Jour 4 — Après-midi

Julien Rolland

Formation M2 Développement Fullstack

Jour 4

Plan du module

La polysémie en pratique

- « La **souris** mange le fromage. »
- « Ma **souris** d'ordinateur est cassée. »

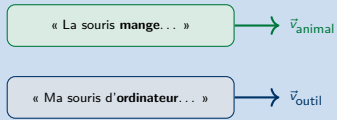
Dans Word2Vec, « souris » n'a **qu'un seul vecteur** — une moyenne des deux sens.

Les autres limites (rappel J4-AM)

- Embeddings **figés** après entraînement
- Ignorent l'**ordre** des mots dans la phrase
- Biais hérités du corpus d'entraînement

Le besoin : embeddings contextuels

Le vecteur d'un mot doit être calculé **à la volée** en fonction de ses voisins.



Solution

Le **mécanisme d'attention** — cœur du Transformer.

Principe & exemple

Chaque mot « vote » pour ceux qui l'aident à se définir.

« L'animal ... car **il** était fatigué. »

Distribution d'attention de « il » :

- **animal** : **90 %**
- fatigué : 6 %
- autres mots : 4 %

Avantage : parallélisme

Contrairement aux RNN, l'attention traite **tous les mots simultanément** — GPU.

Distribution d'attention de « **il** »

90 % **animal** — co-référence directe

6 % **fatigué** — trait partagé

4 % autres mots (négligeable)

Interprétation

Le modèle « sait » que « il » réfère à « animal » **grâce au contexte** global — sans règle grammaticale explicite.

La formule de l'attention

Mécanisme Query / Key / Value

- Q (*Query*) : « Ce que je **cherche**. »
- K (*Key*) : « Ce que j'**offre**. »
- V (*Value*) : « L'information que je **contiens**. »

Équation

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

- $QK^\top$: score de pertinence (mot, mot)
- $\sqrt{d_k}$: normalisation (gradients stables)
- Softmax $\rightarrow$ poids $\in [0, 1]$, somme $= 1$

Multi-Head Attention

On lance h attentions en **parallèle** sur des sous-espaces distincts :

- Tête 1 : relations **grammaticales**
- Tête 2 : relations **sémantiques**
- Tête 3 : relations de **coréférence**
- ...

Les sorties sont **concaténées** puis projetées.

Note « Geek »

C'est une **recherche floue** dans une base de données vectorielle : Q est la requête, K l'index, V le résultat.

Encoder, Decoder, Seq2Seq

Encoder (ex: BERT)

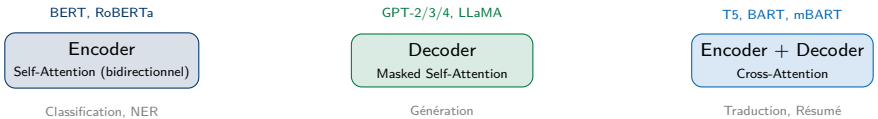
- \+ Lit la phrase **dans les deux sens**
- \+ Embeddings contextuels riches
- Idéal : **classification**, NER, question-answering

Decoder (ex: GPT)

- \+ Génère mot après mot (*autoregressive*)
- Attention **masquée** : ne voit que le passé
- Idéal : **génération** de texte, chatbots

Encoder-Decoder (ex: T5)

- \+ Encode la source, décode la cible
- Idéal : **traduction**, résumé, reformulation



Pre-training — la fondation

- Entraînement massif sur **tout Internet** (Wikipédia, livres, code...)
- Objectif : **prédire le token masqué** (BERT) ou le token suivant (GPT)
- Coût : millions d'euros de calcul GPU
- Résultat : le modèle apprend la **« langue »** — grammaire, faits, raisonnement

Fine-tuning — l'adaptation

On ajuste le modèle sur une **tâche métier** :

- Classification de tickets support
- Résumé automatique de rapports
- Détection de sentiment client

Quelques dizaines de lignes de code, quelques **minutes** de calcul → niveau état de l'art.

Pourquoi ça marche ?

Les couches inférieures encodent des connaissances **générales** réutilisables ; seule la tête de classification change.

L'écosystème Hugging Face

The Hub

Le « GitHub » des modèles d'IA :

- $>$ 100 000 modèles prêts à l'emploi
- Hébergement gratuit des poids
- Versionning, fiches modèles, benchmarks intégrés

`transformers`

Interface **unifiée** pour tout modèle :

- BERT, RoBERTa, CamemBERT
- GPT-2, LLaMA, Mistral
- T5, BART, Whisper, CLIP...

Même API quel que soit l'architecture.

`datasets`

- Accès instantané à des milliers de bases de données (IMDb, Wikipedia, GLUE, SQuAD...)
- Chargement en une ligne :

```
load_dataset("imdb")
```

Écosystème complet

- `tokenizers` : tokeniseurs rapides (Rust)
- `evaluate` : métriques standardisées
- `accelerate` : multi-GPU/TPU
- `PEFT` : fine-tuning efficace (LoRA)

La `pipeline` API — le mode « Product »

Concept

Abstraction totale : téléchargement du modèle, tokenisation, inférence, décodage — tout en **une seule ligne**.

```python
from transformers import pipeline

clf = pipeline("sentiment-analysis")
print(clf("Ce cours est dense !"))
# [{'label':'POSITIVE', 'score':0.98}]

trad = pipeline("translation",
    model="Helsinki-NLP/opus-mt-fr-en")
print(trad("Bonjour le monde"))
# [{'translation_text':'Hello World'}]
```

Tâches disponibles

- `"sentiment-analysis"`
- `"text-generation"`
- `"summarization"`
- `"translation"`
- `"question-answering"`
- `"ner"` (entités nommées)

Intérêt terrain

C'est ce qu'un dev Fullstack utilisera **90 % du temps** en entreprise. Pas besoin de connaître l'architecture interne.

Anatomie du Fine-Tuning — le mode « Expert »

Les 3 composants obligatoires

1. **Tokenizer** : identique au modèle pré-entraîné
2. **Model Head** : couche de sortie remplacée (ex: 30 000 $\rightarrow$ 2 classes)
3. **Trainer** : boucle d'entraînement optimisée

```python
from transformers import (
    AutoTokenizer,
    AutoModelForSequenceClassification,
)
model = AutoModelForSequenceClassification\
  .from_pretrained("camembert-base",
                   num_labels=2)
tokenizer = AutoTokenizer\
  .from_pretrained("camembert-base")
```

Pourquoi le Tokenizer doit coïncider ?

Le modèle a appris des embeddings pour des IDs précis. Si l'on utilise un tokenizer différent, les IDs ne correspondent plus aux poids — le modèle produit du bruit.

Alternatives légères

- **LoRA / QLoRA** : $< 1\,\%$ des params
- **Prompt-tuning** : tokens « soft »
- **Zero-shot** : sans fine-tuning

Pourquoi l'accuracy ment

99 % de mails normaux, 1 % de spams.

Prédire « normal » partout → **99 %** — mais le modèle est **inutile**.

Précision & Rappel

$$\text{Précision} = \frac{TP}{TP+FP} \qquad \text{Rappel} = \frac{TP}{TP+FN}$$

F1-Score — moyenne harmonique

$$F_1 = 2 \cdot \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Matrice de confusion

Visualiser **où** le modèle se trompe :

	Prédit +	Prédit –
Réel +	TP vrais +	FP faux +
Réel –	FN faux –	TN vrais –

Exemple

Confondre « colère » et « tristesse » est moins grave que « positif » et « négatif » — la matrice le révèle.