Mes premiers pas ...

Traitement Automatique des Langues avec des modèles génératifs

Eric Blaudez

A qui s’adresse ce cours ?

Aux professionnels de la fouille de données débutants dans l’analyse documentaire

Aux étudiants/débutants intéressés par l’analyse documentaire

Aux curieux avides de connaissances

Aux *datascientists* peu familiers avec le traitement des langues

Historique

Statistiques et apprentissage automatique

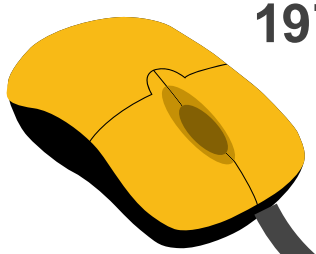
l'approche statistique a pris de l'importance avec l'utilisation de modèles de langage probabilistes et d'algorithmes d'apprentissage automatique; des modèles tels que les chaînes de Markov cachées (HMM) et les modèles de Markov conditionnels (CRF) ont été utilisés.

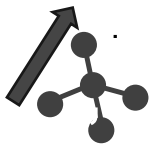
Essor des reseaux de neurones

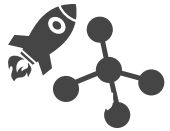
Les progrès dans le domaine de l'apprentissage profond ont révolutionné le TALN. Les réseaux de neurones profonds ont permis d'obtenir des performances remarquables dans de nombreuses tâches, y compris la classification de texte, la traduction automatique et bien d'autres.

Les transformers

Les Transformers, introduits par le modèle **"Attention is All You Need"** en 2017, ont révolutionné le TALN. Ils permettent de prendre en compte les dépendances à long terme dans les données et ont été appliqués avec succès à une grande variété de tâches du TALN.







1950-1960

- expérimentations en linguistique computationnelle et en traduction
- premiers systèmes étaient basés sur des règles codées manuellement.

1970-1990

1990-2000

Début des reseaux de neurones

Ils ont commencé à être appliqués au TALN.

2000-2010

2010-2016

Words embedings

Introduction des word embeddings : Word2Vec (Mikolov et al., 2013) permet d'apprendre des représentations vectorielles denses via CBOW ou Skip-gram ; GloVe (Pennington et al., 2014) les complète par une approche matricielle sur les co-occurrences globales.

2017

2018-

LLM

Les modèles de langage massivement pré-entraînés, comme GPT-2 et GPT-3 de OpenAI, ainsi que BERT de Google, ont établi de nouveaux sommets de performances dans de nombreuses tâches du TALN. Ces modèles sont entraînés sur de vastes corpus de texte et peuvent être adaptés à des tâches spécifiques avec peu de données d'entraînement supplémentaires.

Introduction

01 Le traitement automatique des langues

02 Quelques exemples

03 Les difficultés des langues

Traitement Automatique du Langage (TAL – NLP en englais)

Traitement **automatique** des contenus textuels sous plusieurs formes : livres, articles, blogs, forums, tweets, sms … ; afin d'extraire et de représenter l'information.

Nous cherchons à structurer des sources textuelles non structurées par nature.

Pourquoi ?

L'explosion de la quantité de données textuelles nécessite des traitements automatiques pour valoriser les contenus (l'or des données), pour les comprendre et pour les chercher.

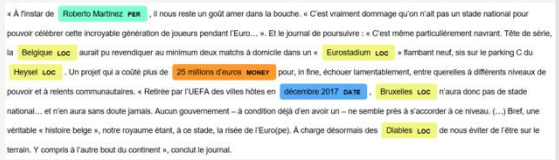
Le traitement automatique du langage offre la possibilité d'extraire de l'information des documents, de les structurer et d'y apporter du sens.

De nombreux exemples montrent la pertinence des outils de TAL.

Applications du TAL / NLP

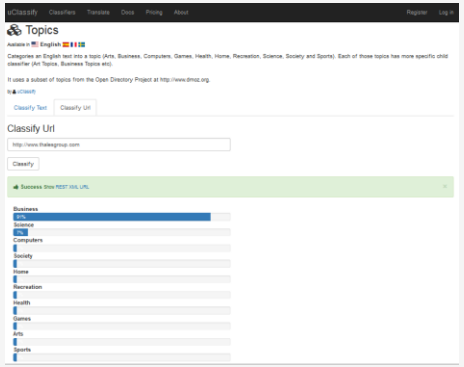
Extraction d'information

Extraire de l'information depuis un texte



Classification de document

Classifier des documents (supervisé ou non supervisé)



Recherche d'information

Trouver des documents à partir d'une requête

Question / Réponse

Trouver une réponse courte à une question

Résumé automatique

Extraire une partie ou générer un texte résumant le document

Traduction automatique

Traduire un texte dans toutes les langues

Analyse de sentiments

Déterminer si le sujet d'un document est négatif ou positif

La complexité du langage écrit

Alphabets

Les langues n'utilisent pas toutes les mêmes alphabets : par exemple le russe et le français.

Mots composés

- Par exemple « **pomme de terre** » est un mot composé : ce n'est pas une pomme qui pousse dans la terre !
- En Allemand le mot « ***Donaudampfschiffahrtsgesellschaft*** » signifie « **Société de navigation à vapeur du Danube** »

Grammaire / Syntaxe

Parfois les langues sont écrites de manières phonétiques

Les langues dans les langues

Avec la technologie, de nouveaux usages apparaissent et le langage écrit peut également se trouvé quelque peu déformé. Prenez l'exemple du langage SMS ou des *Tweets*

L’écosystème LLM moderne

01 Évolution 2022-2025

02 Familles de modèles

03 Les benchmarks

L’écosystème LLM : la révolution 2022-2025

- **2022 - ChatGPT et RLHF** : le tournant de l'interaction. OpenAI publie InstructGPT puis lance ChatGPT, propulsant les LLMs dans le grand public grace au Reinforcement Learning from Human Feedback (RLHF).

- **2023 - Democratisation** : GPT-4 (OpenAI), Claude 2 (Anthropic), Gemini (Google). Cote open-source : Llama 2, Falcon 40B, Mistral 7B atteignent des performances remarquables avec bien moins de parametres.

- **2024 - Efficacite et contexte long** : Gemini 1.5 Pro (1 million de tokens de contexte), Llama 3, Mistral Large, Command R+. L'accent passe de la puissance brute a l'efficacite, la securite et le controle.

- **2025 - Raisonnement et agents** : o3 (OpenAI), DeepSeek R1 - les modeles 'pensent' avant de repondre (Chain-of-Thought interne). Les agents autonomes multi-taches et les modeles multimodaux natifs deviennent mainstream.

Les grandes familles de modèles LLM

Deux grandes categories structurent l'ecosysteme : les modeles proprietaires (acces via API, opacite) et les modeles open-source (poids disponibles, customisables).

- **GPT-4o (OpenAI)** modele proprietaire multimodal (texte+image+audio). 128K tokens de contexte. Reference pour la plupart des benchmarks academiques et la generation de code.
- **Claude 3.5 / Claude 4 (Anthropic)** fort en raisonnement complexe, comprehension de longs documents (200K tokens), generation de code et taches de securite. Disponible via API.
- **Gemini 2.0 (Google DeepMind)** nativement multimodal (texte, image, audio, video). Gemini 1.5 Pro : 1 million de tokens de contexte, permet d'analyser des livres entiers ou des heures de video.
- **Llama 3.3 70B (Meta)** modele open-source de reference. Poids disponibles gratuitement. Peut etre deploye localement avec Ollama, vLLM ou HuggingFace. Performances proches de GPT-4.
- **Mistral Large (Mistral AI)** modele europeen open-weight tres efficient. Mistral 7B depasse GPT-3.5 avec 10x moins de parametres. Mixture-of-Experts avec Mixtral 8x7B.
- **DeepSeek R1 (DeepSeek)** modele de raisonnement open-source chinois (2025). Egale o1 d'OpenAI sur les benchmarks de maths et de code. A democratise les modeles de raisonnement.

Évaluer un LLM : les benchmarks de référence

- MMLU (Massive Multitask Language Understanding) : 57 disciplines academiques. Mesure la connaissance generale. GPT-4 depasse 86%, score humain de reference ~89%.
- HumanEval / MBPP : evaluation de la generation de code. Taux de programmes corrects au premier essai (pass@1). Code Llama 3, DeepSeek Coder atteignent plus de 80%.
- TruthfulQA : mesure la tendance a halluciner sur des questions piegees. Revele que les modeles les plus grands ne sont pas toujours les plus honnetes.
- HELM (Holistic Evaluation of Language Models) : framework multi-dimensionnel - precision, calibration, equite, biais, robustesse, efficacite...
- MT-Bench / Chatbot Arena (LMSYS) : evaluation par preference humaine. Les humains choisissent la meilleure reponse entre deux modeles en aveugle. Reference pour le classement en conditions reelles.
- Limite importante : un LLM peut etre entraine involontairement sur les donnees de test (contamination). Les benchmarks deviennent rapidement obsoletes face a la vitesse du domaine.

Les données & les corpus

01 La collecte des données

02 Le filtrage & la qualité

03 L’analyse et la transformation

04 La gestion & la gouvernance

Collecte, qualité, transformation et gouvernance des données pour le traitement automatique du langage. Ce module explore le cycle de vie complet de la donnée, de sa collecte brute jusqu'à son archivage conforme, en passant par les étapes critiques de filtrage, d'annotation et de versioning.

« L'or des données » ne vaut que raffiné : un modèle n'est jamais meilleur que le corpus qui l'a nourri. La valeur ne réside pas dans la quantité brute, mais dans la qualité maîtrisée de chaque étape du cycle de vie.

Pourquoi les données décident de tout

Le paradigme *data-centric*

La qualité d'un système de TAL dépend davantage de la qualité et de la pertinence des **données** que du choix de l'architecture. À modèle égal, c'est le corpus qui fait la différence — une réalité souvent sous-estimée dans les projets où l'attention se concentre sur les hyperparamètres plutôt que sur la matière première.

Garbage In, Garbage Out (GIGO)

Un biais, un doublon ou une erreur d'étiquetage dans le corpus se propage — et **s'amplifie** — dans toutes les prédictions du modèle. La corruption des données en entrée est systématiquement reproduite à l'échelle de l'ensemble des inférences réalisées.

Lois d'échelle (*scaling laws*)

La performance croît avec la **quantité** de données, le nombre de paramètres et le budget de calcul. Le travail *Chinchilla* (DeepMind, 2022) montre qu'à budget de calcul fixé, l'optimum se situe autour de **~20 tokens de données par paramètre** : sous-alimenter un grand modèle est un gâchis computationnel et économique considérable.

Qualité > quantité au-delà d'un seuil

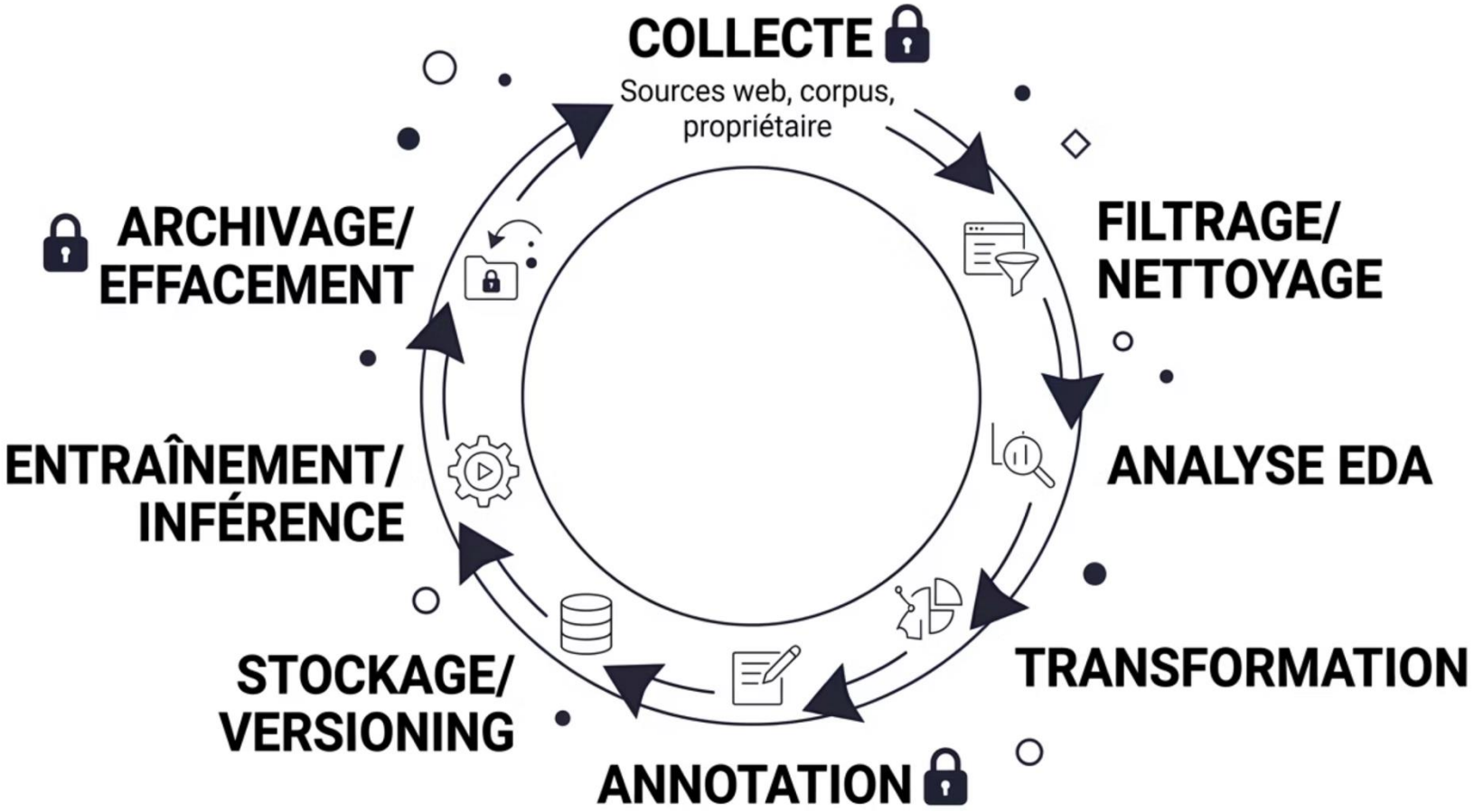
Dédupliquer et filtrer un corpus web améliore souvent plus les performances qu'ajouter des données brutes supplémentaires. Les corpus *RefinedWeb* et *FineWeb* en sont des illustrations récentes et probantes : moins de volume, mais bien plus de signal utile.

Le coût caché : 60 à 80 % de l'effort

Dans un projet TAL réel, la grande majorité du temps et des ressources est consacrée à la collecte, au nettoyage et à l'annotation — **pas à la modélisation**. Anticiper et budgéter cet effort est une compétence clé du praticien.

Le cycle de vie de la donnée

Chaque étape du cycle de vie est interdépendante. Une décision prise lors de la collecte conditionne toutes les étapes suivantes, jusqu'à l'archivage ou l'effacement légal.



Les étapes **Collecte**, **Annotation** et **Archivage/Effacement** sont particulièrement encadrées par le RGPD et l'AI Act

La collecte : sources & méthodes

Les grandes familles de sources

Web ouvert à grande échelle

Common Crawl (pétaoctets), corpus filtrés dérivés : *C4*, *The Pile*, *OSCAR*, *mC4*, *RefinedWeb*, *RedPajama*, *Dolma*, *FineWeb*.

Corpus curés / de référence

Wikipédia (dumps), corpus de presse, livres, articles scientifiques, données gouvernementales ouvertes, *WordNet*.

Données propriétaires / internes

Bases métier, tickets clients, e-mails, logs, documents contractuels — souvent la **vraie valeur différenciante** d'un projet d'entreprise.

Crowdsourcing & données synthétiques

Plateformes d'annotation (MTurk, Prolific) ; génération par LLM (distillation, augmentation) quand la donnée réelle est rare, sensible ou coûteuse.

Méthodes d'acquisition

→ Crawling / scraping

Respect du `robots.txt` et des CGU — une obligation légale et éthique non négociable.

→ APIs officielles & flux temps réel

Streaming *vs* batch selon le cas d'usage ; les APIs garantissent souvent un meilleur respect des droits.

→ ETL/ELT depuis bases existantes

Extraction de données structurées vers le pipeline NLP.

Formats courants

Texte brut, `CSV/TSV`, **JSON/JSONL** (standard NLP, un objet par ligne), **Parquet / Avro / ORC** (colonnaire compressé, pour le volume), XML, PDF/DOCX (extraction multimodale).

⚠ La **licence** de chaque source conditionne son usage en entraînement et sa redistribution. L'AI Act impose aux modèles GPAI un résumé public des données et une politique de droit d'auteur.

Constituer un corpus de qualité

Représentativité, couverture & documentation

Collecter ne suffit pas : il faut **construire** un corpus *représentatif* et *documenté*. Un corpus mal conçu dès l'origine compromet toutes les étapes ultérieures, quelle que soit la sophistication du modèle.

Représentativité & couverture

Distribution réelle cible

Le corpus doit refléter les domaines, registres, langues et populations que verra le modèle en production. Un déséquilibre à cette étape induit un modèle structurellement biaisé.

Biais d'échantillonnage

Sur-représentation de l'anglais, de certaines régions géographiques ou points de vue idéologiques. Ces biais doivent être **identifiés dès la collecte**, pas découverts après l'entraînement.

Langues peu dotées (*low-resource*)

Variantes dialectales, code-switching, langage SMS et réseaux sociaux nécessitent une attention particulière lors de l'échantillonnage.

Documentation du corpus (essentielle & trop souvent négligée)

Datasheets for Datasets *(Gebru et al.)*

Motivation, composition, collecte, prétraitement, usages recommandés et déconseillés. Un standard de facto pour toute publication sérieuse.

Data Statements for NLP *(Bender & Friedman)*

Caractéristiques des locuteurs et annotateurs, contexte de production des textes — indispensable pour évaluer les biais.

Croissant *(MLCommons)*

Format de **métadonnées standardisé** pour les jeux de données ML, adopté par les grands hubs (Hugging Face, Kaggle). Inclut date, source, licence, langue, auteur — piliers de la traçabilité et de la conformité.

Le filtrage & le nettoyage

De la donnée brute à la donnée exploitable

Passer de la donnée **brute** à la donnée **exploitable** : une chaîne d'opérations séquentielles, généralement appliquées dans cet ordre. À chaque étape, une fraction du volume initial est écartée — sur un corpus web typique, on passe de 100 % brut à environ **~30 % exploitable**.

Normalisation d'encodage

Tout ramener en `UTF-8`, normalisation Unicode (NFC/NFKC), correction des caractères mal encodés (*mojibake*).

Suppression du *boilerplate*

Menus, bandeaux cookies, pieds de page, balises HTML, publicités — ne conserver que le contenu éditorial utile.

Filtrage par langue

Identification automatique via *fastText langid* ou CLD3 pour ne conserver que la ou les langues cibles.

Filtrage qualité

Heuristiques (longueur, ratio ponctuation, lignes répétées), perplexité d'un LM (KenLM), classifieurs de qualité entraînés à distinguer « texte éditorial » du bruit.

Déduplication

Exacte (hachage SHA) et **quasi-doublons** via MinHash + LSH. Réduit la mémorisation et améliore la généralisation du modèle.

Suppression contenus toxiques & PII

Classifieurs de toxicité, listes de blocage, retrait de CSAM ; détection et **anonymisation/pseudonymisation** des données personnelles. Exigence RGPD impérative.

Décontamination

Retirer du corpus d'entraînement les données issues des **benchmarks de test** — sans quoi l'évaluation est faussée par du *data leakage*.

Maîtriser la qualité des données

Un processus mesurable, pas un état

La qualité n'est pas un état binaire mais un **processus continu et mesurable**, défini selon plusieurs dimensions complémentaires. L'approche **DataOps** consiste à intégrer ces tests dans le pipeline CI/CD pour bloquer automatiquement les régressions de qualité avant qu'elles n'atteignent l'entraînement.

Les six dimensions de la qualité

Complétude

Absence de champs ou valeurs manquants critiques pour la finalité du modèle.

Exactitude & Cohérence

Les valeurs correspondent au réel ; absence de contradictions entre sources ou champs.

Validité, Unicité & Fraîcheur

Conformité aux formats attendus, absence de doublons, données à jour pour la finalité visée.

Problèmes courants & traitements

Valeurs manquantes

Suppression, imputation statistique, ou indicateur explicite selon le contexte métier.

Valeurs aberrantes (*outliers*)

Détection statistique (IQR, z-score), mise en quarantaine pour revue humaine.

Déséquilibre des classes

Ré-échantillonnage (oversampling SMOTE, undersampling), pondération des pertes.

Incohérences de format

Dates, casse, unités, identifiants → standardisation par règles ou normalisation.

Outils de tests qualité à grande échelle

Great Expectations, *Pandera*, *Soda*, *dbt tests*, *Deequ* (Spark). Principe : intégrer ces tests **dans le pipeline** pour bloquer les régressions de qualité en CI/CD, avant tout chargement en production.

L'analyse exploratoire (EDA)

Comprendre le corpus avant de le transformer

Avant toute transformation, **comprendre** le corpus par l'**Exploratory Data Analysis**. L'EDA conditionne tous les choix en aval : longueur de séquence maximale, stratégie d'équilibrage, tokenisation, découpage en chunks. Une EDA rigoureuse est le « carnet de bord » du corpus, rédigé avant toute décision de modélisation.

Statistiques descriptives

Volumétrie totale, nombre de documents, distribution des longueurs (caractères, tokens, phrases). Identifier les extrêmes (très courts, très longs).

Profilage du vocabulaire

Taille du lexique, fréquences (loi de Zipf), n-grammes les plus fréquents, ratio mots-outils *vs* mots de contenu. Révèle la richesse et le registre du corpus.

Analyse diversité / redondance

Taux de doublons résiduels, entropie du corpus, couverture thématique. Un corpus très redondant sur-entraîne des patterns superficiels.

Analyse des étiquettes (supervisé)

Distribution des labels, **déséquilibre de classes**, taux d'accord/désaccord inter-annotateurs. Un déséquilibre non détecté biaise les métriques d'évaluation.

Détection d'anomalies

Documents vides, tronqués, dupliqués, hors-langue, hors-domaine. Mettre en quarantaine pour revue manuelle avant inclusion.

Visualisation

Histogrammes de longueurs, nuages de points d'embeddings réduits (PCA/t-SNE/UMAP), matrices de co-occurrence. Rendent visibles les structures cachées du corpus.

La transformation des données

Mettre la donnée en forme exploitable par le modèle

Opérations de transformation

1 Normalisation textuelle

Casse, accents, ponctuation, espaces, expansions d'abréviations via regex — à doser avec précaution pour ne pas supprimer d'information pertinente.

2 Tokenisation en sous-mots

BPE, WordPiece, Unigram LM — le découpage en unités sub-lexicales est la brique fondamentale de tout pipeline transformer moderne.

3 Vectorisation / *embeddings*

TF-IDF (historique) puis embeddings contextuels issus des transformers : transformer le texte en vecteurs denses exploitables par le modèle.

4 Structuration (non structuré → structuré)

Extraire des champs, entités, relations à partir de texte libre. C'est le cœur du passage **« texte → connaissance »** — introduit le Module B.

Encodage, découpage & orchestration

Encodage des étiquettes

One-hot, indices entiers, format **BIO/IOB** pour l'étiquetage de séquences NER/NP-chunking.

Découpage & *sharding*

Segmentation en *chunks* compatibles avec la fenêtre de contexte du modèle ; partitionnement pour l'entraînement distribué multi-GPU/multi-nœuds.

Pipelines ETL / ELT

ETL (transformer avant de charger) *vs* **ELT** (charger puis transformer dans l'entrepôt). Orchestration : *Airflow*, *Dagster*, *Prefect*, *Spark*, *dbt*.

L'annotation & l'augmentation

Créer la vérité-terrain & enrichir le corpus

L'annotation : créer la vérité-terrain pour l'apprentissage supervisé

Annotation manuelle

Guidelines précises et exhaustives, formation des annotateurs, double annotation systématique pour détecter les désaccords. La qualité du guide d'annotation détermine directement celle des labels.

Apprentissage actif (*active learning*)

Le modèle sélectionne les exemples les plus **informatifs** (incertitude maximale) à faire annoter en priorité — réduit drastiquement le coût d'annotation.

Supervision faible (*Snorkel*)

Règles, heuristiques et fonctions d'étiquetage programmatiques pour étiqueter à grande échelle sans annotation manuelle exhaustive.

Pré-annotation par LLM

Le modèle propose des labels, l'humain corrige (*human-in-the-loop*). Accélère considérablement la production de données labelisées.

Qualité de l'annotation : accord inter-annotateurs mesuré par le **kappa de Cohen** (2 annotateurs) ou **de Fleiss** (≥3) ; résolution des désaccords par adjudication d'un expert. Outils : *Label Studio*, *Prodigy*, *doccano*, *INCEpTION*, *Argilla*.

L'augmentation de données

Élargir le corpus, surtout en régime de faibles ressources (*low-resource*) :

Rétro-traduction

FR→EN→FR pour paraphraser et diversifier les formulations sans créer de nouvelles annotations.

EDA (*Easy Data Augmentation*)

Remplacement de synonymes, insertion, suppression et permutation aléatoire de tokens.

Génération synthétique par LLM

Paraphrase, distillation d'un modèle plus puissant, génération de cas limites rares.

⚠ L'**AI Act art. 10** (systèmes à haut risque) exige des jeux d'entraînement, validation et test **pertinents, représentatifs et examinés sous l'angle des biais**.

La gestion & le versioning

Industrialiser : stocker, versionner, tracer, orchestrer

Architectures de stockage

Data Lake

Stockage brut, schéma à la lecture. Tolérant aux formats hétérogènes. Idéal pour l'exploration et les cas d'usage NLP non encore définis.

Data Warehouse

Structuré, schéma à l'écriture. Requêtes analytiques performantes. Adapté aux *features* définies et stables.

Lakehouse

Hybride : *Delta Lake*, *Iceberg*, *Hudi*. Transactions ACID sur un data lake. Le standard émergent pour le ML industriel.

Object storage (S3, GCS, Azure Blob) pour le volume ; **formats colonnaires** (Parquet) pour l'analytique ; **Feature store** pour centraliser les features prêtes à l'emploi, partagées entre entraînement et inférence.

Versioning, lineage & orchestration

Versionner les données

DVC, *Git-LFS*, *lakeFS*, versions de tables Delta/Iceberg. Répond à la question clé : **« quel corpus exact a produit ce modèle ? »** — condition sine qua non de la reproductibilité.

Traçabilité / *data lineage*

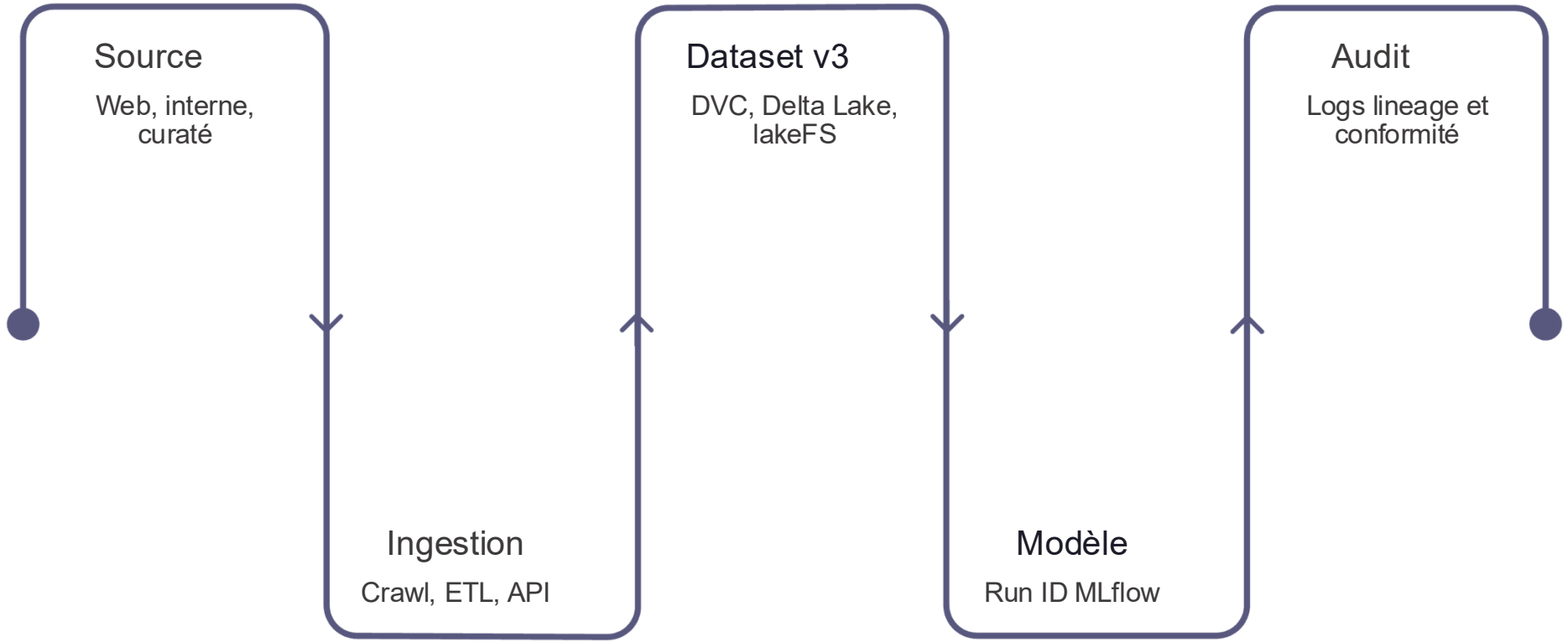
Suivre l'origine et toutes les transformations d'une donnée. Pilier de la **reproductibilité** et de la **conformité** (preuve à fournir à l'AI Act). Outils : *DataHub*, *Amundsen*, *OpenMetadata*.

Orchestration DataOps & MLOps

Automatiser collecte→tests→transformation→entraînement. Outils : *Airflow*, *Dagster*, *Prefect*, *Kubeflow*, *MLflow*.

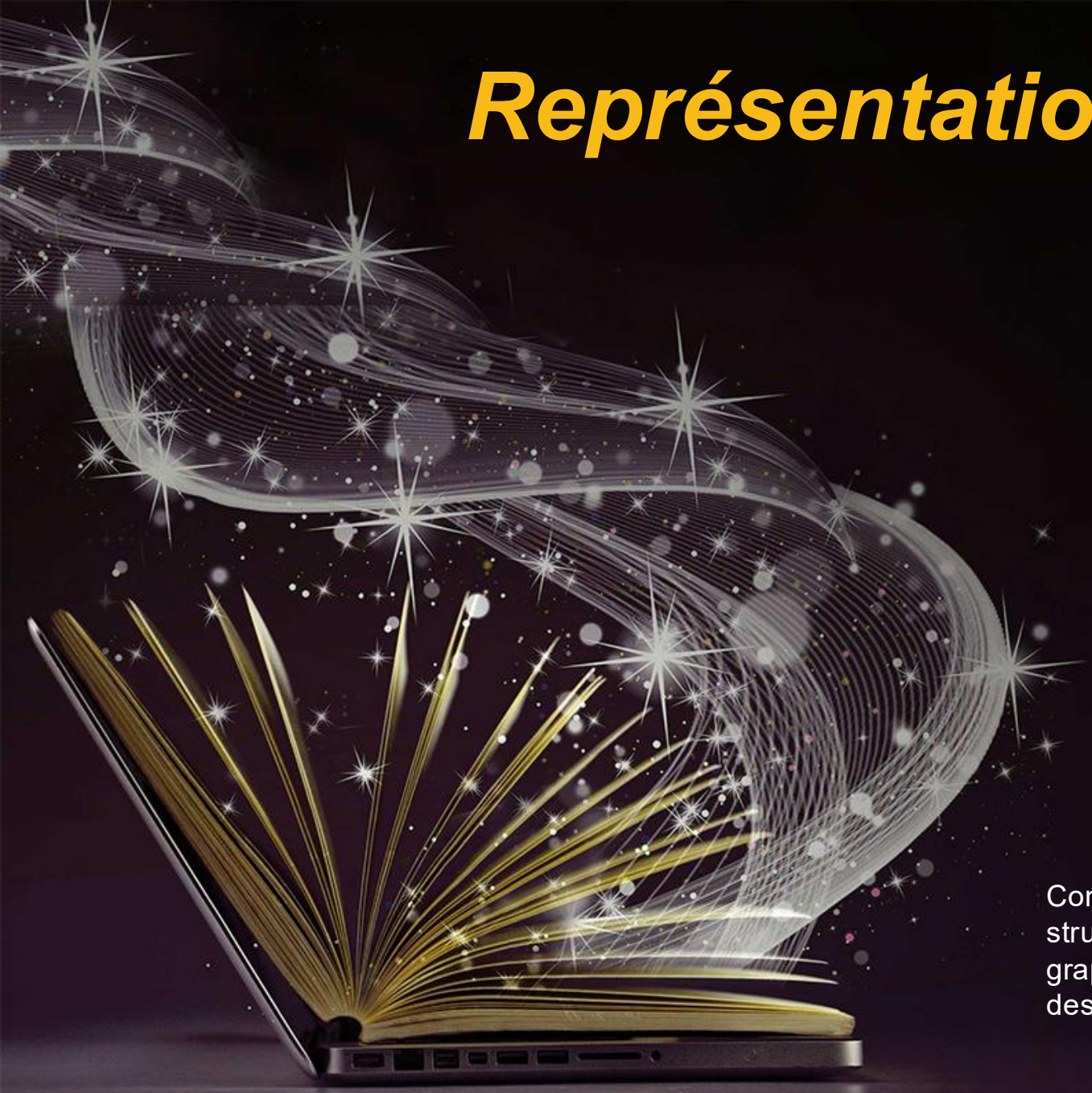
Du corpus au modèle : le lineage de données

La traçabilité (*data lineage*) est la capacité à reconstituer, à tout moment, le chemin complet d'une donnée depuis sa source jusqu'au modèle en production — et inversement. C'est à la fois un outil de débogage, un gage de reproductibilité et une obligation réglementaire sous l'AI Act.



Bonne pratique : chaque artefact (dataset, modèle, évaluation) doit être identifiable par un **identifiant unique versionné**, liant le modèle au corpus exact qui l'a produit et à l'expérience MLflow correspondante.

Catalogues de données : *DataHub*, *Amundsen*, *OpenMetadata* permettent de découvrir, documenter et gouverner les jeux disponibles au sein d'une organisation.



Représentation des connaissances

01 Des données à la connaissance

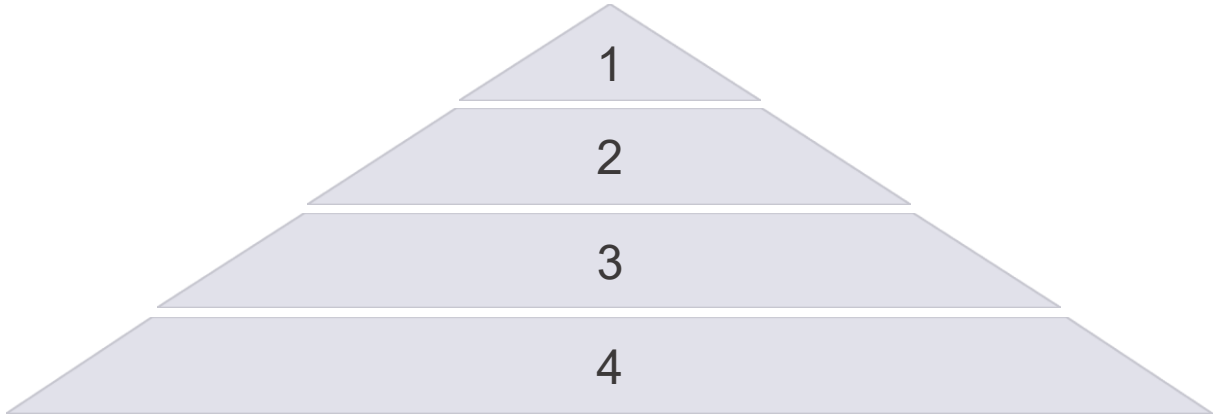
02 Taxonomie et structure de classification

03 Ontologies et logiques sous-jacentes

04 Graphes de connaissances

Comment passer de la donnée brute à la connaissance formalisée, structurée et exploitable par les machines. De la pyramide DIKW aux graphes sémantiques, en passant par les ontologies et les logiques de description.

La Pyramide DIKW



1. Wisdom
2. Knowledge
3. Information
4. Data

Donnée (Data)

Faits bruts, sans contexte. Ex : "28", "Gaston", "Paris".

Information

Donnée contextualisée. Ex : *"Gaston se produit à Paris le 28 février."*

Connaissance

Information raisonnée. Permet la déduction logique et répond au *Comment ?*

Sagesse (Wisdom)

Jugement critique, éthique et décision. Répond au *Pourquoi ?*

Connaissance Tacite vs Connaissance Explicite

L'ingénierie des connaissances segmente le savoir selon son niveau de formalisation, suivant le modèle **SECI de Nonaka & Takeuchi**.

Socialisation — Tacite → Tacite

Le partage de connaissances tacites s'opère directement entre individus par le biais d'**expériences partagées, de l'observation, de l'imitation et du mentorat**, sans recours au langage formel ni à la documentation écrite. C'est le mécanisme le plus naturel et le plus ancien de transmission du savoir.
Exemple : Un apprenti cuisinier qui apprend le coup de main d'une recette secrète en observant attentivement un chef et en pratiquant à ses côtés, absorbant progressivement des gestes et des intuitions impossibles à écrire.

Externalisation — Tacite → Explicite

Étape cruciale consistant à **articuler la connaissance tacite pour la transformer en concepts explicites** compréhensibles par tous. On traduit l'intuition et le savoir-faire grâce à des métaphores, des analogies, des modèles et des hypothèses, rendant ainsi le savoir individuel transmissible à grande échelle.
Exemple : Un ingénieur senior qui rédige un manuel de dépannage ou une procédure opérationnelle standard (SOP) sur la base de ses années d'expérience personnelle, cristallisant ainsi un savoir autrement perdu à son départ.

Combinaison — Explicite → Explicite

Ce processus consiste à **organiser, fusionner et synthétiser différents blocs de connaissances explicites** pour créer de nouvelles connaissances plus systématiques et complexes. On trie, recatégorise et recompose des documents, des bases de données et des rapports pour produire une vision intégrée.
Exemple : Un analyste financier qui rassemble divers rapports de marché, des feuilles de calcul de ventes internes et des données de tendances pour construire la stratégie financière globale de l'entreprise, générant une connaissance nouvelle à partir de sources existantes.

Internalisation — Explicite → Tacite

Ce processus se produit lorsque les individus **absorbent les connaissances explicites et les intègrent dans leurs propres modèles mentaux**, les transformant en savoir tacite personnel. C'est le principe du "learning by doing" : la formation et l'expérience répétée jusqu'à ce que la connaissance devienne un réflexe naturel.
Exemple : Un nouvel employé qui lit le guide du service client de l'entreprise et s'exerce régulièrement jusqu'à gérer naturellement les appels difficiles sans consulter les règles — la procédure est devenue instinct.

Pourquoi représenter la connaissance à l'ère des LLM ?

Ancrage (Grounding)

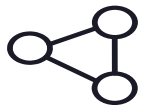
Relier les vecteurs probabilistes des LLM à des faits certifiés réduit drastiquement les **hallucinations**.

Raisonnement Explicable (XAI)

Contrairement aux boîtes noires neuronales, une ontologie offre une **trace d'inférence logique auditable** étape par étape.

Édition de Connaissances

Modifier un fait erroné dans un graphe prend une milliseconde avec une **cohérence absolue garantie** — impossible dans les poids d'un LLM.



Interopérabilité

Partager un **vocabulaire commun** et des contraintes métiers immuables entre systèmes hétérogènes.

Le Paradigme de l'IA Neuro-Symbolique

IA Symbolique — Logique

Forces : Connaissance explicite, logique formelle stricte, zéro hallucination, raisonnement exact et déterministe.
Faiblesses : Fragile face au bruit, incapable de gérer naturellement l'ambiguïté du langage naturel.

IA Connexionniste — Neuronale

Forces : Apprise par l'exemple, perception fluide, robustesse face aux variations textuelles.
Faiblesses : Opaque (boîte noire), probabiliste, sujette aux hallucinations logiques, sans garantie d'axiomes métiers.

L'**IA Neuro-Symbolique** réalise la **convergence** des deux courants : la flexibilité connexionniste au service de la rigueur symbolique. C'est l'architecture d'avenir des systèmes RAG et KG-augmented LLMs.

Fondements de la KR & Le Compromis de Levesque

Qu'est-ce que la KR ?

La **Représentation des Connaissances (KR)** est la discipline de l'IA qui étudie comment encoder formellement ce qu'un système « sait » afin qu'un agent computationnel puisse simuler un raisonnement logique sur le monde réel.

Le Compromis de Levesque (1984)

Hector Levesque a formalisé la tension fondatrice de toute la KR :

> **Expressivité du formalisme ↔ Calculabilité (Tractabilité)**

- **Logique du premier ordre (FOL) : très expressive, mais semi-décidable ;** les preuves peuvent boucler indéfiniment.
- **Logiques de Description (DL) :** Sous-ensembles *décidables* de la FOL. Point d'équilibre choisi pour les langages du Web sémantique.

La Pile du Web Sémantique / Layer Cake W3C

L'architecture standardisée du Web sémantique s'articule en couches de protocoles où chaque niveau s'appuie strictement sur le précédent.

IRI / URI

Identifiants uniques globaux pour chaque ressource ou concept.

RDF

Modèle de données en triplets (Sujet – Prédicat – Objet).

RDFS & OWL

Schémas de classes et axiomes logiques complexes (restrictions, disjonctions).

SPARQL & SHACL

Requête sémantique déclarative et validation de conformité structurelle.

Les Structures Taxonomiques Formelles

Une **taxonomie** est un système de classification hiérarchique organisant les concepts d'un domaine selon des relations d'inclusion strictes, fondées sur la relation sémantique **Is-A** (Est-Un).

Sémantique de la Subsomption

La relation Is-A (⊑)

La subsomption encode l'inclusion de classes :

$$A \sqsubseteq B \Rightarrow \forall x\,(A(x) \rightarrow B(x))$$

Exemple :

$$\text{Humoriste} \sqsubseteq \text{Artiste} \sqsubseteq \text{Personne}$$

Propriétés fondamentales

Asymétrie

Si A est sous-concept de B, B ne peut être sous-concept de A (sauf égalité stricte).

Transitivité

Si Humoriste ⊑ Artiste et Artiste ⊑ Personne, le système infère automatiquement Humoriste ⊑ Personne (**héritage**).

Héritage Multiple

Un concept peut hériter de plusieurs branches : Humoriste ⊑ Artiste *et* Humoriste ⊑ Auteur.

Anomalies et Bonnes Pratiques en Ingénierie Taxonomique

La conception de taxonomies solides exige de respecter des règles logiques fondamentales pour éviter les contresens de modélisation.

⚠ Confusion Subsomption / Instanciation

Erreur : Gaston_LaGaffe ⊑ Humoriste
Règle : Gaston_LaGaffe est une *instance* (individu concret), pas une classe. La relation correcte est l'appartenance : Gaston_LaGaffe ∈ Humoriste.

⚠ Confusion Subsomption / Méréologie

Erreur : Roue ⊑ Voiture — une roue *n'est pas une* voiture.
Règle : Utiliser une propriété dédiée partieDe (méréologie) distincte de la relation Is-A.

✓ Disjonction Inter-branches

S'assurer que les sous-concepts d'une même classe supérieure s'excluent mutuellement :

$$\text{Concept\_Abstrait} \cap \text{Objet\_Physique} = \emptyset$$

Normalisation des Thésaurus : Le Standard W3C SKOS

Lorsque les contraintes logiques des taxonomies formelles sont trop rigides pour les besoins métier, l'ingénierie recourt au standard **SKOS** (*Simple Knowledge Organization System*).

skos:Concept

L'unité de pensée atomique — l'**idée**, indépendamment du mot qui la désigne.

Labels Multilingues

skos:prefLabel (terme préférentiel), skos:altLabel (synonyme), skos:hiddenLabel (fautes de frappe).

Relations Hiérarchiques

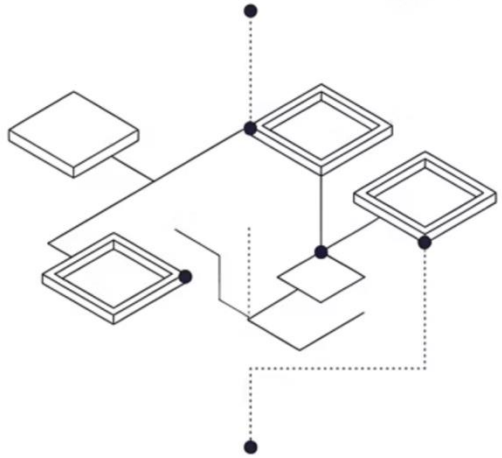
skos:broader : concept plus générique.
skos:narrower : concept plus spécifique (descendant).

Relation Associative

skos:related : **Proximité sémantique sans inclusion**. Ex : *Humour* et *Rire*.

Taxonomie Formelle vs SKOS : Quelle Différence ?

Taxonomie Formelle (DL/OWL)



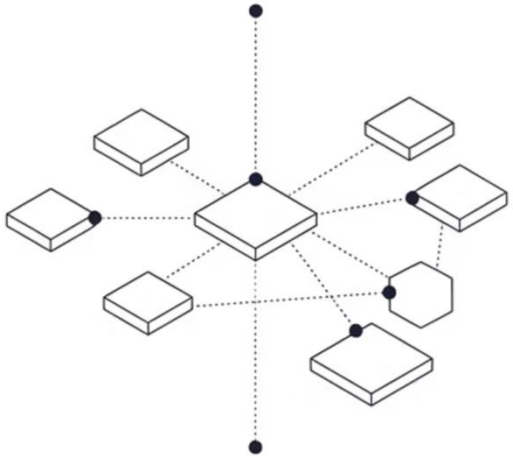
Taxonomie Formelle (DL/OWL)

Logique stricte, inférence décidable.

Relation Is-A (⊑), héritage transitif automatique.

Convient aux ontologies OWL.

SKOS



SKOS

Vocabulaire contrôlé souple, relations relâchées (broader/narrower/related).

Multilingue.

Convient aux thésaurus métier, pas d'inférence automatique.

Le choix entre une taxonomie formelle et SKOS dépend du niveau de rigueur logique requis. Pour des systèmes nécessitant un raisonnement automatique, OWL est indispensable. Pour l'indexation documentaire et la gestion terminologique, SKOS offre la souplesse nécessaire.

Les Ontologies et les Logiques de Description

Les **Logiques de Description (DL)** constituent le fondement formel des ontologies OWL 2. Elles définissent le découpage **TBox / ABox / RBox** et permettent le raisonnement automatique via l'algorithme du Tableau.

Découpage TBox / ABox / RBox

TBox — Terminological Box

Définit les **classes et leurs relations** générales (axiomes de subsomption, restrictions). C'est le *schéma* de la base de connaissances.

ABox — Assertional Box

Contient les **assertions sur les individus** (instances concrètes). C'est l'équivalent des données dans la base.

RBox — Role Box

Décrit les **propriétés des relations** (transitivité, symétrie, inverse, chaînes de rôles). Introduite par OWL 2.

L'**algorithme du Tableau** est la procédure de décision fondamentale des DL : il vérifie la cohérence d'une ontologie en construisant un modèle et en détectant les contradictions par réfutation.

Qu'est-ce qu'une ontologie ? (Définitions canoniques)

Le concept philosophique d'ontologie (étude de la nature de l'existence) a été transposé en informatique pour désigner un artefact logiciel.

> Tom Gruber (1993) : « Une ontologie est une spécification explicite d'une conceptualisation. »

> Studer, Benjamins & Fensel (1998) : « Une ontologie est une spécification formelle, explicite, d'une conceptualisation partagée. »

Conceptualisation

Un modèle abstrait d'un domaine ou phénomène du monde réel, identifiant les concepts et relations pertinents.

Explicite

Les concepts et les contraintes logiques sont déclarés sans aucune ambiguïté textuelle.

Formelle

Le langage possède une sémantique mathématique rigoureuse, rendant le modèle calculable et interprétable par une machine.

Partagée

Elle est issue d'un consensus, garantissant l'interopérabilité sémantique entre systèmes.

Les composants d'une ontologie

Une ontologie enrichit la taxonomie pure en introduisant des types complexes de données, des contraintes d'intégrité et une logique formelle.

Classes / Concepts (owl:Class)

Catégories d'objets du domaine (ex : Spectacle, Lieu, Artiste).

Individus / Instances (owl:NamedIndividual)

Les éléments réels peuplant les classes (ex : Gaston_LaGaffe).

Propriétés d'Objet (owl:ObjectProperty)

Relations binaires reliant deux instances entre elles (ex : seProduitDans(Spectacle, Lieu)).

Propriétés de Données (owl:DatatypeProperty)

Attributs reliant une instance à un littéral brut typé (ex : aPourDateDeDebut(Spectacle, xsd:dateTime)).

Axiomes

Énoncés logiques contraignant les assertions du modèle pour définir les limites du sens.

Axiomatique algébrique des propriétés en OWL 2

La sémantique des relations en OWL 2 permet d'inférer automatiquement de nouveaux liens logiques sans intervention humaine.

Domaine (rdfs:domain) & Image (rdfs:range)

Fixent le type logique de départ et d'arrivée. Si ⟨x, seProduitDans, y⟩, alors le système déduit automatiquement que x ∈ Spectacle et y ∈ Lieu.

Propriété Fonctionnelle

Un individu possède au maximum un seul objet pour cette propriété.

$$\forall x, y, z(P(x, y) \land P(x, z) \rightarrow y = z)$$

Propriété Fonctionnelle Inverse

L'objet de la relation détermine le sujet de manière unique (ex : un numéro de sécurité sociale).

Propriété Transitive

$$\forall x, y, z(P(x, y) \land P(y, z) \rightarrow P(x, z))$$

Ex : partieDe.

Propriété Symétrique

$$\forall x, y(P(x, y) \rightarrow P(y, x))$$

Ex : aPourCollaborateur.

Axiome de Disjonction de Classes (owl:disjointWith) : Déclarer formellement que deux ensembles n'ont aucune intersection (Lieu ⊓ Personne = ∅).

Logiques de Description (DL) & Architecture TBox / ABox / RBox

Les bases de connaissances s'organisent selon la séparation mathématique stricte issue des Logiques de Description (DL).

TBox (Terminological Box) — Le Schéma Général

Contient les définitions de concepts, les structures hiérarchiques et les axiomes logiques universels. Elle formalise la structure immuable du domaine.

Exemple : Humoriste ⊑ Artiste

ABox (Assertional Box) — Les Faits Contingents

Contient les assertions individuelles sur les entités réelles, leurs types déclarés et leurs relations mutuelles.

Exemple : Humoriste(Gaston_LaGaffe), seProduitA(Gaston_LaGaffe, Olympia)

RBox (Role Box)

Contient les interdictions, hiérarchies et caractéristiques s'appliquant spécifiquement aux rôles (propriétés).

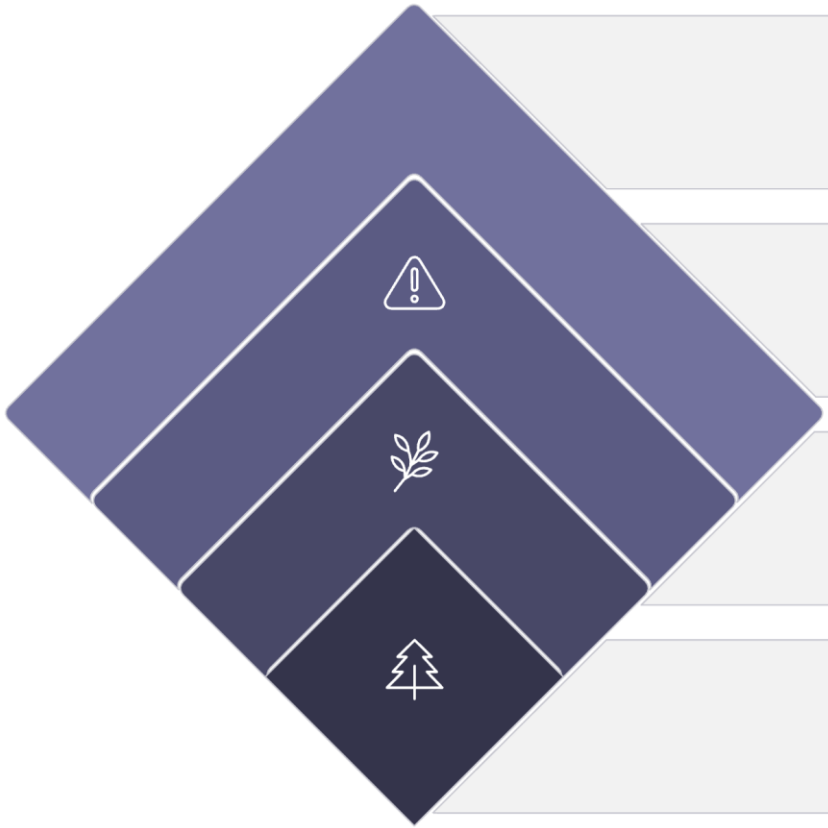
Exemple : aPourSousPropriété(aPourInvitéSpécial, aPourCollaborateur)

Les Algorithmes de Raisonnement : Le Tableau Sémantique

Les Raisonneurs sémantiques (HermiT, Pellet, FaCT++) appliquent une méthode mathématique appelée l'Algorithme du Tableau pour prouver la cohérence d'un modèle par réfutation.

Mécanisme de la preuve par réfutation

Pour valider une assertion logique ou vérifier la cohérence d'un nouveau concept, le raisonneur introduit la négation de la proposition et tente de générer une contradiction explicite (un clash).



Résultat
Clashs tous ou ouvert

Détection de clash
Vérifier contradictions

Règles d'expansion
Conjonction vs disjonction

Créer arbre
Initialiser contraintes

Hypothèse de Monde Ouvert vs Monde Fermé

Une divergence théorique fondamentale sépare la KR sémantique des bases de données informatiques traditionnelles.

Hypothèse de Monde Ouvert (Open World Assumption - OWA) → OWL / RDF

- Principe : Si un fait n'est pas explicitement présent dans la base de connaissances, il est considéré comme inconnu, et non comme faux.
- Philosophie : Le Web est incomplet. Ne pas trouver l'information que Gaston a un permis de conduire ne signifie pas qu'il ne sait pas conduire.

Hypothèse de Monde Fermé (Closed World Assumption - CWA) → SQL / SHACL

- Principe : Si une donnée n'est pas encodée explicitement dans le système, elle est considérée comme mathématiquement fausse.
- Application : Idéal pour les contraintes de validation d'intégrité logicielle (ex : "Si l'attribut email n'existe pas, l'enregistrement est invalide").

Les Graphes de Connaissances : Définition et Modèle d'Équation

Un Graphe de Connaissances (Knowledge Graph - KG) matérialise et met à l'échelle industrielle les théories sémantiques.

Graphe de connaissance = Données d'intance (Abox) + Schéma Sémantique (Tbox)

Le graphe gère le stockage physique des nœuds (entités du monde réel) et de leurs arêtes topologiques typées (relations). Un graphe de connaissance peut-être être vu comme une instanciation particulière d'un ontologie

L'ontologie agit comme le système de types strict, le filtre de validation et le moteur d'inférence logique du graphe. L'ontonolgie peut-être vue comme le modèle d'un graphe de connaissance

Historique : Concept popularisé par Google en 2012 (« things, not strings »). Exemples emblématiques ouverts : Wikidata, DBpedia, YAGO.

Graphes Sémantiques RDF vs Graphes de Propriétés (LPG)

L'architecture technique sous-jacente d'un KG se divise entre deux grands modèles concurrents sur le marché de l'ingénierie.

Critère d'évaluation	Modèle RDF (Graphes Sémantiques / Triple Stores)	Modèle Labeled Property Graph (LPG)
Unité Fondamentale	Le triplet strict ⟨Sujet, Prédicat, Objet⟩	Le Nœud et l'Arête vus comme des objets complexes.
Métadonnées sur les liens	Impossible nativement (nécessite la réification ou le standard moderne RDF-star).	Paires Clé-Valeur stockées directement au sein des nœuds et des arêtes.
Identifiants	IRI globale unique au monde, favorisant l'interconnexion globale.	ID technique local à l'instance de la base de données.
Sémantique & Logique	Forte et standardisée (RDFS/OWL), raisonneurs natifs.	Faible ou purement applicative (pas de logique formelle native).
Langage de Requête	SPARQL (Standard déclaratif W3C).	Cypher (Neo4j) ou Gremlin.
Technologies clés	GraphDB, Apache Jena, Blazegraph, Stardog.	Neo4j, TigerGraph, Amazon Neptune.

Le Méta-Niveau dans RDF : Réification et standard RDF-star (RDF*)

Pour annoter un fait avec des métadonnées comme la provenance, une période de validité ou un indice de confiance, l'architecture RDF a dû évoluer.

La Réification Classique (Standard RDF 1.1)

Pour exprimer que le triplet ⟨Gaston, seProduitA, Olympia⟩ a un indice de confiance de 0.95, la réification classique crée un nœud virtuel de type rdf:Statement.

⚠ Inconvénient : Multiplie par 4 le volume de triplets dans la base, dégradant fortement les performances des requêtes SPARQL.

La solution moderne : RDF-star (RDF*)

Permet d'enchâsser un triplet complet en position de sujet ou d'objet d'un autre triplet, résolvant le problème de performance.

```
<< :Gaston_LaGaffe :seProduitA :Olympia_Paris >>
:indiceConfiance 0.95^^xsd:float ; :aPourSource
:Billetterie .
```

Pipeline de Construction Industrielle de Graphes

La transformation de documents textuels bruts non structurés en un graphe sémantique propre exploite les composants TAL du Module 4.

Extraction d'Entités (NER)

Repérage et classification des entités dans le texte (cf. slides 56-59).

Extraction de Relations (RE)

Détection des liens syntaxiques et sémantiques reliant les entités extraites.

Liage d'Entités (Entity Linking / Désambiguïsation)

Résolution des homonymies en rattachant l'entité textuelle à une IRI unique de référence du graphe.

Résolution d'Entités / Déduplication

Fusion des nœuds synonymes désignant le même objet réel à l'aide de l'axiome owl:sameAs.

Mapping Ontologique

Alignement final des instances et relations extraites sur le schéma strict de la TBox cible.

Complétion de Graphes et Plongements de Graphes (KGE)

Pour pallier l'incomplétude structurelle des graphes réels, l'ingénierie utilise les Knowledge Graph Embeddings (KGE) pour prédire les liens manquants (Link Prediction).

Principe

Projeter les entités (s, o) et les relations (r) dans un espace vectoriel continu $\mathbb{R}^d$, de manière à optimiser une fonction de score f_r(s,o) pour les triplets valides.

Modèles algorithmiques de référence

TransE (Bordes et al.)

Modélise la relation comme une pure translation géométrique : s + r ≈ o. La fonction de perte pénalise la distance euclidienne ||s + r - o|| pour les triplets invalides.

DistMult

Utilise une décomposition bilinéaire via des matrices diagonales (limité mathématiquement aux relations symétriques).

ComplEx

Projette les représentations dans l'espace des nombres complexes ($\mathbb{C}^d$) afin de pouvoir capturer les propriétés asymétriques et dirigées (ex : estLePereDe).

Algorithmique des Structures de Données pour le TAL

L'implémentation bas niveau des architectures de traitement des connaissances repose sur des structures informatiques fondamentales qu'il convie

Trie (Arbre Préfixe)

Structure optimale pour le stockage des lexiques géants de tokenization (ex : Byte-Pair Encoding). Garantit une recherche de motifs en complexité linéaire O(m) dépendante uniquement de la longueur du mot, indépendamment de la taille du dictionnaire

Arbres Syntaxiques & de Dépendances

Représentations hiérarchiques de la structure grammaticale d'une phrase

Index Inversé

Table associant un terme à la liste triée des identifiants des documents qui le contiennent, optimisée pour la recherche plein texte exacte (

Index Vectoriel Topologique (HNSW)

Graphe hiérarchique de type Small World structurant les embeddings continus pour résoudre la recherche des plus proches voisins (ANN) en temps logarithmique

Graphes de Connaissances & Structures de Données

Les **graphes de connaissances** (Knowledge Graphs) sont la concrétisation applicative des formalismes symboliques. Ils combinent richesse sémantique, scalabilité et compatibilité avec les pipelines d'apprentissage automatique.

Modèles RDF vs LPG & Extensions

RDF — Resource Description Framework

Modèle de triplets standardisés par le W3C (sujet, predicat, objet):.

- Interopérable sur le Web via IRI/URI
- Requêtes SPARQL déclaratives
- Extension **RDF-star** : permet d'annoter des triplets eux-mêmes (méta-triplets), résolvant l'incapacité native à représenter des faits sur des faits.

LPG — Labeled Property Graph

Modèle natif des bases de graphes comme **Neo4j** : nœuds et arêtes portent directement des propriétés clé-valeur.

- Plus expressif localement, performant pour les traversées
- Requêtes Cypher, moins interopérable que RDF
- Adapté aux applications métier nécessitant des attributs riches sur les relations

Plongements de graphes (KGE) : Les techniques comme TransE, RotatE ou ComplEx projettent les entités et relations dans des espaces vectoriels continus, permettant la complétion de graphes et l'intégration avec des pipelines de deep learning.

Points Clés à Retenir

1 DIKW & SECI

La connaissance n'est pas la donnée. La montée en abstraction — de la donnée brute à la sagesse — est le fil directeur de toute ingénierie des connaissances.

2 KR vs LLM

La représentation symbolique apporte ce que les LLM ne peuvent garantir seuls : ancrage factuel, explicabilité, édition déterministe et interopérabilité.

3 Taxonomies & SKOS

La relation Is-A (⊑) et ses propriétés (asymétrie, transitivité, héritage) sont le socle. SKOS offre la souplesse terminologique pour les cas métier moins rigides.

4 OWL 2 & Logiques de Description

Le découpage TBox/ABox/RBox et les profils OWL 2 (EL, QL, RL) permettent d'équilibrer expressivité et calculabilité dans les ontologies formelles.

5 RDF, LPG & KGE

Les graphes de connaissances concrétisent les formalismes symboliques. RDF-star et les plongements de graphes ouvrent la voie à l'intégration neuro-symbolique.

L’avènement des transformers

01 Des stratégies classiques au réseau de neurones « *transformers* »

02 Les *transformers* en détails

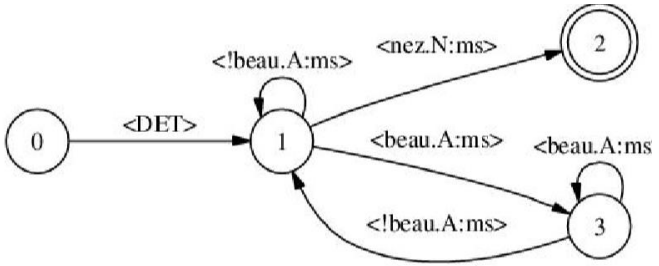
03 Les premiers pas …

Approches à base de grammaires et de règles

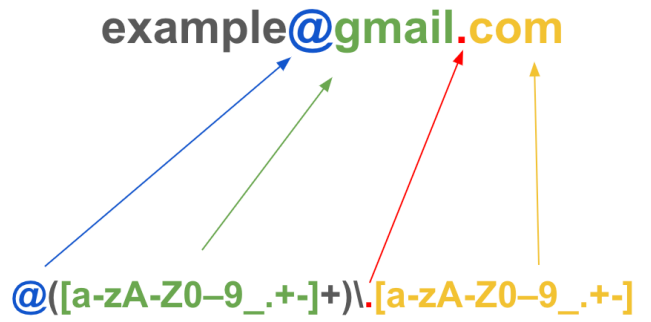
- **Grammaires** fournissent un ensemble de règles formelles pour décrire la structure syntaxique et sémantique d'une langue donnée.

$$
\begin{aligned}
E &\rightarrow E + E \\
&\mid E * E \\
&\mid (E) \\
&\mid \texttt{int}
\end{aligned}
$$

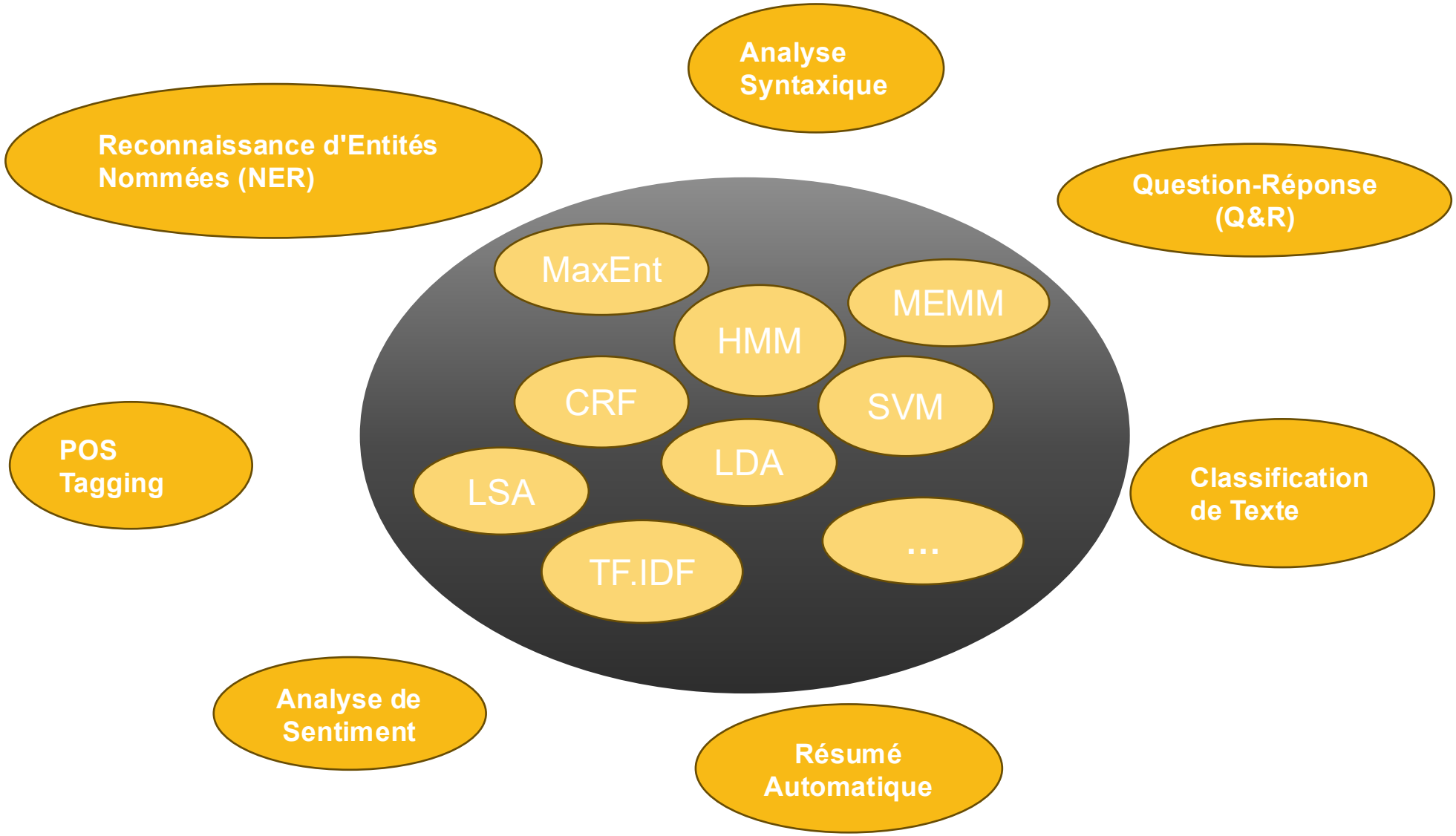
- **Systèmes à Base de Règles** fonctionnent sur la base d'un ensemble de règles prédéfinies qui spécifient des correspondances entre des motifs linguistiques et des actions à entreprendre



- **Expressions Régulières**, abrégées en regex, sont des séquences de caractères qui définissent des motifs de recherche.



Approches statistiques



Approches connexionnistes

Pour l'ensemble des tâches d'étiquetages les approches connexionnistes sont pertinentes, voire révolutionnent les approches précédentes avec les réseaux de type *transformers*:

- **Capacité à modéliser les dépendances à long terme** : ils sont capables de capturer des dépendances à long terme dans les données. Contrairement aux modèles classiques qui ont du mal à saisir les relations complexes entre les mots distants dans une phrase, les transformers peuvent intégrer ces informations de manière efficace.
- **Attention multi-têtes** : ils utilisent une architecture d'attention multi-têtes qui permet au modèle de se concentrer sur différentes parties de la séquence d'entrée simultanément. Cela permet de capturer des informations importantes à partir de différentes parties de la phrase.
- **Adaptabilité à différentes tâches** : ils sont très adaptables et peuvent être appliqués à une grande variété de tâches en TALN. Ils ont prouvé leur efficacité dans des tâches allant du POS tagging à la traduction automatique en passant par la génération de texte.
- **Apprentissage auto-supervisé** : ils peuvent être pré-entraînés sur de grands corpus de texte dans le cadre d'une tâche d'apprentissage auto-supervisé (comme la prédiction du mot suivant dans une phrase). Cette pré-entraînement permet au modèle d'acquérir des connaissances linguistiques générales.
- **Fine-Tuning sur des tâches spécifiques,** une fois pré-entraînés, ils peuvent être fine-tunés sur des ensembles de données spécifiques à une tâche particulière. Cela permet d'adapter le modèle pour des performances optimales dans un contexte donné.
- **Réduction du besoin d'ingénierie des caractéristiques** :Contrairement aux approches classiques qui nécessitent souvent une ingénierie minutieuse des caractéristiques, ils sont capables d'apprendre des représentations de haute qualité directement à partir des données brutes.
- **État de l'art sur de nombreuses tâches :** ils ont établi de nouveaux états de l'art dans de nombreuses tâches de TALN, notamment la traduction automatique, le résumé automatique, la génération de texte et bien d'autres.
- **Adaptabilité à des langues et domaines divers** : ils peuvent être entrainés sur des corpus dans différentes langues et domaines. Ils sont capables de généraliser leurs connaissances pour traiter des données dans des contextes variés.

Les « transformers » pour le traitement automatique de la langue

- Une percée dans le domaine du TALN, initiée par Google Brain. En 2017, Ashish Vaswani et son équipe ont publié l'article pionnier intitulé "Attention is All You Need", introduisant ainsi le concept d'architectures de réseaux neuronaux basées sur l'attention, communément appelées "Transformers".

- Les architectures de réseaux de neurones traditionnelles étaient confrontées à des problèmes de dépendances à longue distance et de traitement séquentiel limitaient leur capacité à saisir les nuances et les contextes complexes du langage humain. Google a conçu une architecture fondée sur un mécanisme d'attention. Cette innovation a permis aux Transformers de surpasser les modèles précédents en offrant une flexibilité dans la manipulation de données textuelles, ainsi qu'une capacité à saisir les relations complexes entre les éléments d'une séquence.

- Leur impact est ressenti dans divers domaines et a ouvert la voie à de nouvelles avancées dans l'interaction entre les machines et le langage naturel.

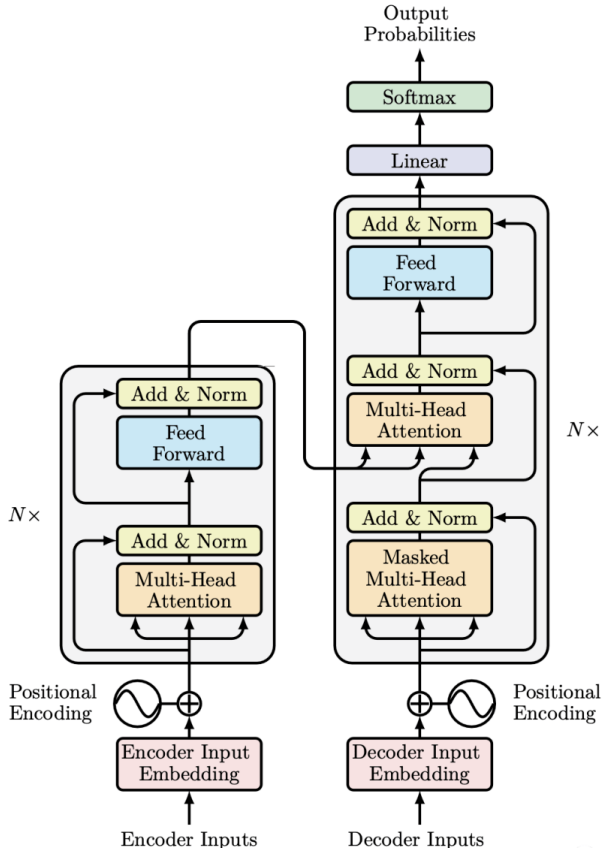
Architecture globale

- **L’encodage de positions** est une étape qui permet au modèle de comprendre l'ordre des éléments dans la séquence. Cela se fait en ajoutant des vecteurs de position aux représentations de mots ou de tokens.

- **L'Encodeur (Encoder)** est responsable de traiter l'entrée et de générer une représentation intermédiaire appelée représentation cachée. Il est composé de plusieurs blocs de traitements contenant:
  - Un mécanisme d’attention multi-têtes permettant au modèle de se focaliser sur des aspects différents de la séquence d'entrée.
  - De couches *feedforward*: après l'opération d'attention, les résultats sont passés à une couche feedforward.
  - Des connexions résiduelles et normalisation de Type "LayerNorm" suit chaque sous-module

- **Décodeur (Decoder)** prend la représentation cachée générée par l'encodeur et génère la sortie finale, souvent sous forme de séquence de mots ou de tokens. A l'instar de l'encodeur il est composé de blocs de traitement contenant:
  - Mécanisme d'Attention Multi-têtes (Auto-attention), il s'agit d'une forme particulière appelée auto-attention. Cela signifie que le décodeur donne plus de poids à certaines parties de la séquence d'entrée en fonction du contexte de sortie.
  - Connexions Résiduelles et Normalisation de Type "LayerNorm", comme pour l'encodeur, chaque sous-module (auto-attention et couche feedforward) est suivi d'une connexion résiduelle et d'une normalisation de type "LayerNorm".

Architecture	Exemples	Tâches typiques
Encodeur seul	BERT, CamemBERT, RoBERTa	Classification, NER, similarité sémantique
Décodeur seul	GPT-2, GPT-3, LLaMA	Génération de texte, complétion
Encodeur–Décodeur	T5, BART, mBART	Traduction, résumé, Q&A génératif



Source : Attention is All You Need

Mécanisme d'attention

Le mécanisme établi des dépendances globales entre l'entrée et la sortie. Il est fondé sur le principe de calcul de poids d'attention pour chaque élément de l'entrée, puis de combiner ces éléments en fonction de ces poids.

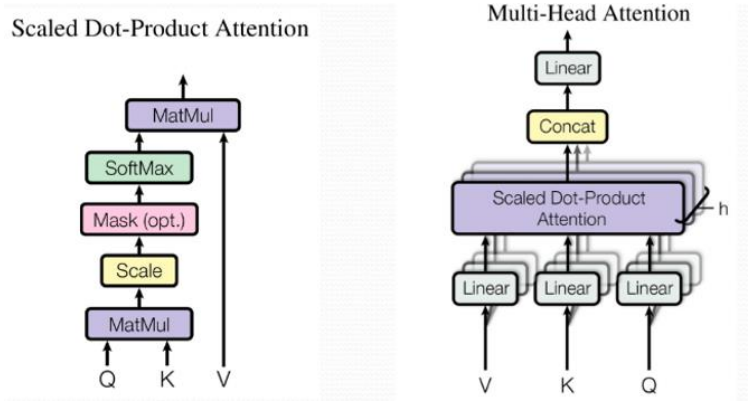
- **Entrée, requête (Q), clé (K), et valeur (V) : l**'entrée, est une séquence de mots ou de tokens. Le mécanisme d'attention repose sur trois éléments: requête (Q), clé (K), et valeur (V). Pour chaque token d'entrée de représentation ***x***, on calcule **Q**, **K**, **V** par projection linéaire apprise :**Q = xW^Q**, **K = xW^K**, **V = xW^V**. Ces trois matrices de poids sont les paramètres entraînés du mécanisme d'attention.

- **Calcul des scores d'attention : l**es scores d'attention sont calculés en mesurant la similarité entre les requêtes (Q) et les clés (K) de chaque élément de l'entrée. Le résultat est une distribution de poids pour chaque élément de l'entrée, indiquant l'importance relative de cet élément par rapport à la requête.

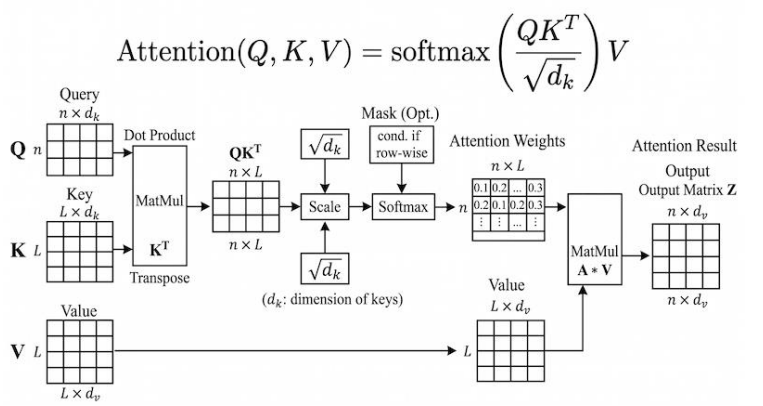
- **Pondération des valeurs: l**es poids d'attention obtenus sont ensuite utilisés pour pondérer les valeurs (V) correspondantes. Cela signifie que les éléments qui ont une forte similarité avec la requête reçoivent plus d'attention, tandis que les éléments moins pertinents en reçoivent moins.

- **Combinaison des valeurs pondérées : l**es valeurs pondérées (V) sont combinées pour former le résultat de l'attention.

- **Multi-têtes (optionnel) :** le mécanisme d'attention peut être répété plusieurs fois. Cela permet de capturer des types d'attention différents à partir de la même entrée, puis de les concaténer ou de les fusionner pour obtenir une représentation finale.



Source : Attention is All You Need



Ce mécanisme d'attention permet au modèle Transformer de capturer des dépendances globales entre l'entrée et la sortie, car il peut attribuer des poids d'attention en fonction du contexte global de la séquence.

LayerNorm & FeedForward

La combinaison de la couche feedforward et de la normalisation de type "LayerNorm" dans l'encodeur et le décodeur des Transformers contribue à

- stabiliser l'apprentissage,
- atténuer les effets de l'initialisation des poids
- introduire de la régularisation

Ce qui en fait des éléments essentiels de l'architecture qui permettent aux Transformers d'atteindre des performances remarquables dans le domaine du TALN.

Le tokenizer

Fonctionnement

- **Découpage en tokens** : le texte brut est divisé en unités discrètes.
- **Création d'un vocabulaire** :le tokenizer crée un vocabulaire, qui est une liste de tous les tokens possibles qui peuvent être rencontrés dans les données d'entraînement.
- **Encodage des tokens** :chaque token est ensuite encodé sous forme numérique en utilisant l'indice correspondant dans le vocabulaire.
- **Gestion des cas spéciaux** :le tokenizer peut gérer des cas spéciaux tels que la ponctuation, les majuscules, les caractères spéciaux, etc. Certains tokenizers prennent également en compte des éléments comme les emojis et les mentions.
- **Gestion des mots inconnus** (OOV - Out Of Vocabulary) :en cas de rencontre d'un token qui n'est pas dans le vocabulaire, le tokenizer peut utiliser une méthode spécifique pour traiter ces mots inconnus.
- **Tokenisation de sous-mots** (Subword Tokenization) : certains tokenizers utilisent des techniques de tokenisation de sous-mots.

Tokenizer pour les Transformers

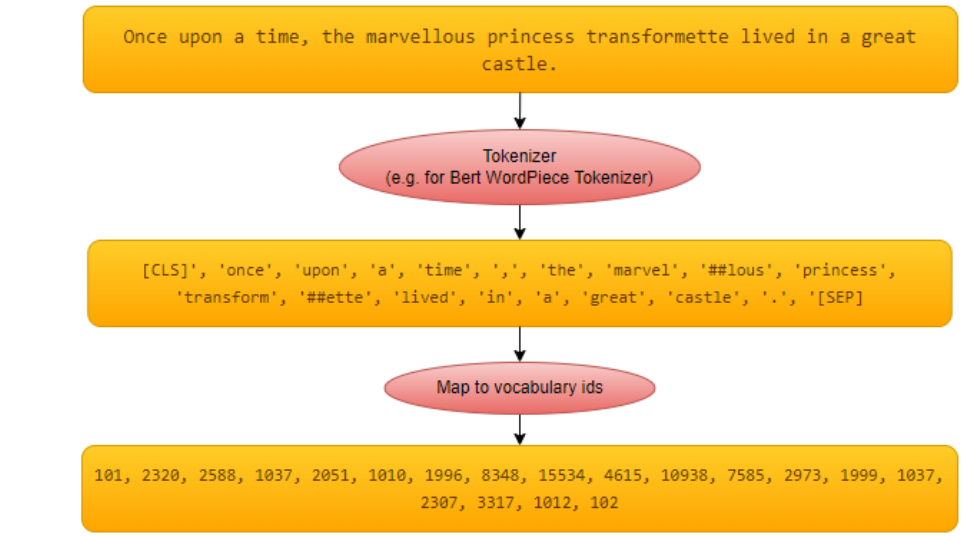
Il existe deux types principaux de tokenizers appliqués aux transformers : les tokenizers de type word et les tokenizers de type subword.

- **Les tokenizers de type word** divisent le texte en mots individuels.
- **Les tokenizers de type subword** divisent le texte en tokens plus petits que des mots

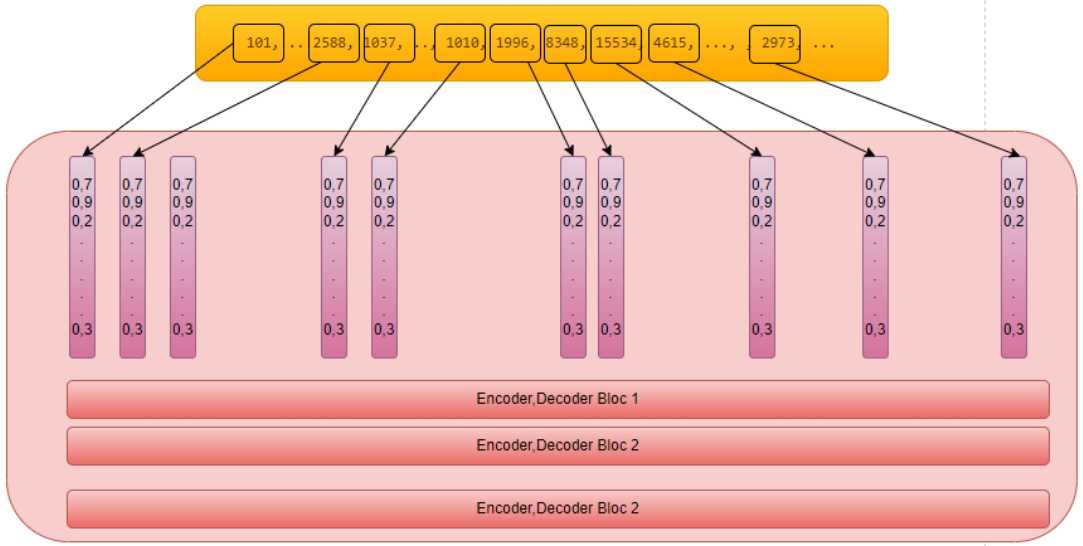
Les transformers utilisent généralement des tokenizers de type subword, car ils sont plus précis et peuvent traiter une plus grande variété de textes.

Algorithme	Utilisé par	Principe
BPE (Byte-Pair Encoding)	GPT-2, GPT-3, GPT-4, LLaMA, Mistral	Fusionne itérativement les paires de caractères les plus fréquentes
WordPiece	BERT, CamemBERT	Similaire à BPE, mais maximise la probabilité du corpus
SentencePiece / Unigram	T5, mBART, LLaMA 2+	Traite le texte brut sans pré-tokenisation par espaces

Illustration tokenizer



- Le vocabulaire d'une langue est de très grande taille
- Pour diminuer cette taille nous utilisons une stratégie construite sur des sous-mots, ainsi l'ensemble des mots d'une langue peuvent être couverts avec un ensemble limité de « sous-mots » (par exemple le modèle bert-small utilise seulement 30K sous mots pour couvrir l'ensemble de la langue anglaise). Chaque « sous-mot » est associé à un identifiant.



- Dans le modèle les identifiants sont associés à un vecteur (*embedding*). Selon les modèles ces vecteurs seront sommés à d'autre vecteurs (comme les *positions embeddings)*
- Ce sont ces vecteurs qui sont utilisés par les « blocs d'attentions » dans le modèle
- Notons ces *embeddings* forme une matrice de grande taille : [taille du vocabulaire]x[[dimension du vecteur]. En entrée du modèle il faudra multiplier encore par la taille du modèle

Illustration mécanisme d'attention

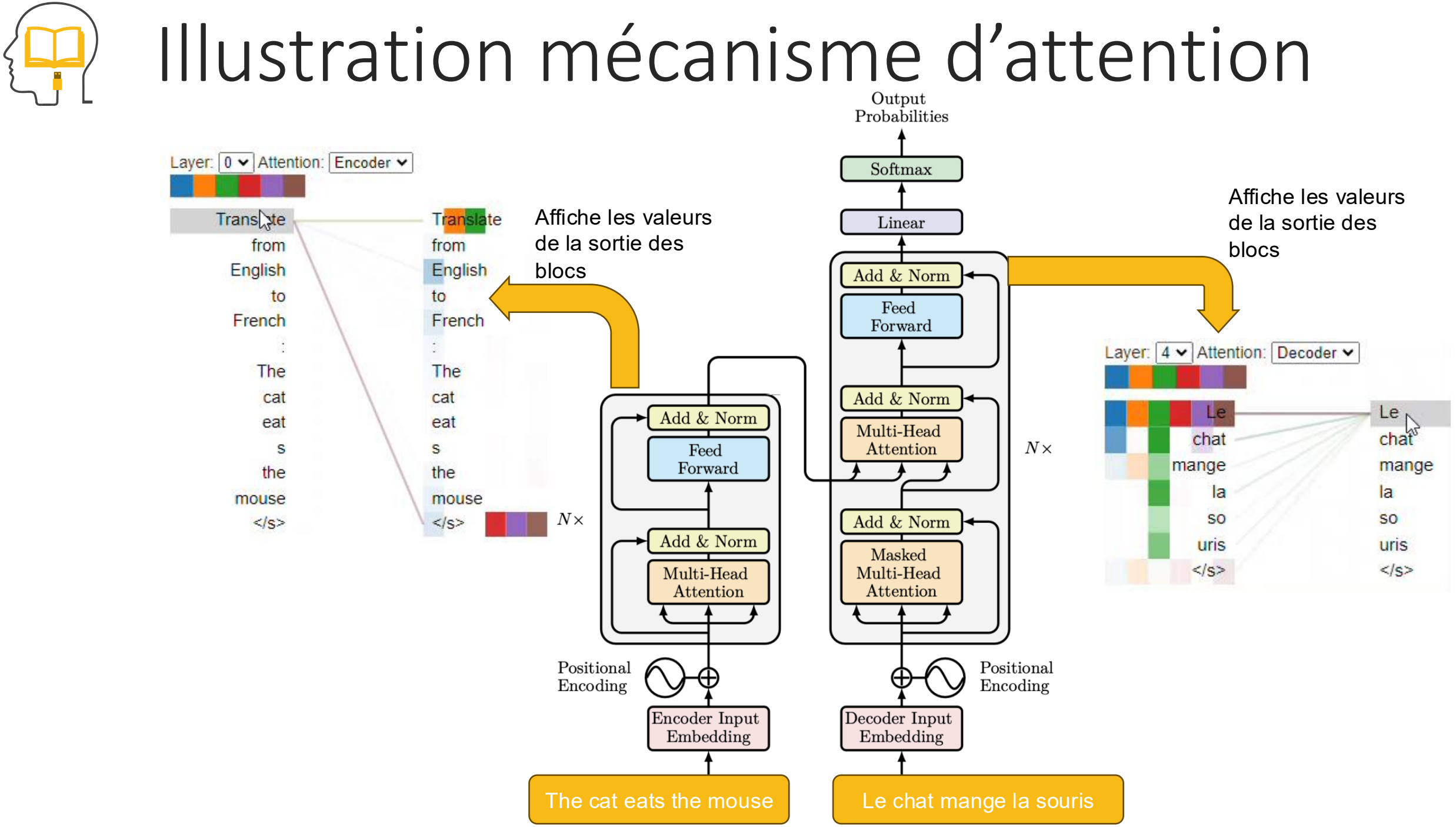
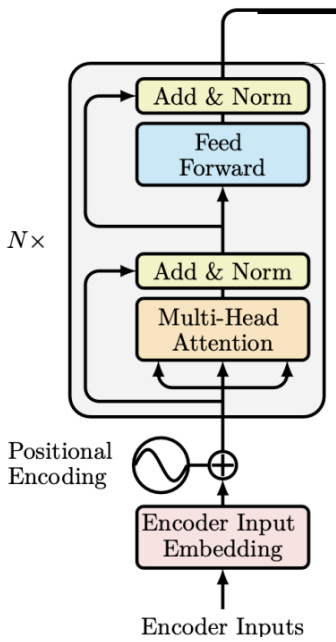
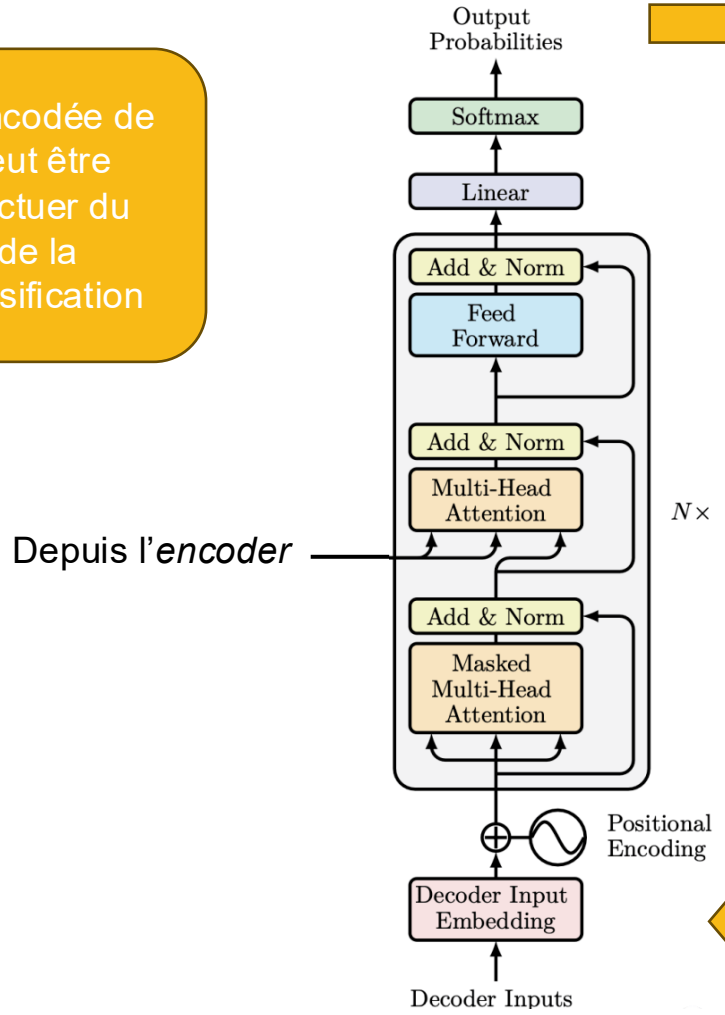


Illustration transformer



Représentation encodée de l'entrée : elle peut être utilisée pour effectuer du clustering ou de la classification classification



Dans le cadre de la génération plusieurs stratégies de recherche des mots suivants:

- ***Greedy*** : sélection de la probabilité maximale
- ***Sampling*** : sélection se fait selon on loi de probabilité (multinomiale par exemple)
- ***Beam search*** : un ensemble de n-probabilité est conservé et on cherche à maximiser une séquence

Le Prompt Engineering

01 Fondamentaux du prompting

02 Les Zero-shot et Few-shot

03 La Chain-of-Thought

04 Techniques avancées

Qu’est-ce que le Prompt Engineering ?

- Definition : l'art de formuler des instructions textuelles (prompts) pour guider un LLM vers la sortie souhaitee, sans modifier les poids du modele. Competence transversale qui remplace parfois un fine-tuning couteux.
- Anatomie d'un prompt : Persona/Role ('Tu es un expert en...'), Contexte (informations pertinentes), Instruction (la tache), Format de sortie attendu (JSON, liste, tableau...), Exemples illustratifs.
- Importance : un prompt mal formule peut diviser par 3 la qualite d'une reponse. Un prompt bien concu peut egaler les performances d'un modele fine-tune sur des donnees specifiques.
- Outils de l'ecosysteme : LangSmith (debogage et traces), PromptFlow (Microsoft, orchestration), DSPy (Stanford, optimisation automatique de prompts), PromptLayer (versionning et monitoring).

Les stratégies de prompting

- **Zero-shot** : le modele repond sans exemple. Efficace pour les LLMs puissants sur des taches simples. Ex: 'Traduis ce texte en anglais : [texte]'.
- **Few-shot** : on fournit 2 a 5 exemples input->output dans le prompt. Ameliore la precision de 20 a 40% pour les taches specialisees ou peu communes.
- **Chain-of-Thought (CoT)** : 'Reflechis etape par etape'. Le modele explicite son raisonnement intermediaire. Reduit les erreurs de 30 a 50% sur les problemes complexes de maths et de logique.
- **Self-Consistency** : generer N raisonnements CoT differents et voter pour la reponse la plus frequente. Tres efficace sur les problemes de raisonnement mathematique.
- **System Prompt / Role Prompting** : definir le role et les contraintes du modele en amont. 'Tu es un expert juridique francais. Reponds uniquement en droit francais et cite tes sources.'
- **Prompt chaining** : decomposer une tache complexe en sous-taches. La sortie du prompt 1 devient l'entree du prompt 2. Base de toutes les architectures agents.

Techniques avancées de prompting

- RAG-prompting : injecter des documents recuperes dans le contexte. 'Sur la base des extraits suivants : [doc1]...[doc2]..., reponds a la question en citant tes sources.' Reduit drastiquement les hallucinations.
- Structured Output / JSON Mode : 'Reponds uniquement en JSON avec les champs { entite, type, confiance }'. La plupart des LLMs modernes supportent le mode JSON natif pour garantir le format de sortie.
- Tree-of-Thought (ToT) : explorer plusieurs chemins de raisonnement en parallele. Le modele evalue et selectionne le meilleur raisonnement, efficace pour la planification et les problemes combinatoires.
- Function Calling / Tool Use : le modele genere des appels structures a des fonctions externes (API meteo, recherche web, base de donnees, calculatrice). Socle des architectures agents.
- Prompt Injection et securite : risque majeur en production - une entree malveillante peut detourner le comportement du modele. Toujours valider et filtrer les entrees utilisateurs dans les applications LLM.

Le Fine-tuning moderne

01 RLHF et alignement

02 Les LoRA et QLoRA

03 La Instruction tuning

04 Quand fine-tuner ?

L’alignement des LLMs : RLHF et DPO

Aligner un LLM signifie lui apprendre a suivre les instructions humaines, a etre utile, honnete et sans danger - au-dela de la simple prediction du prochain token.

- **SFT (Supervised Fine-Tuning)** : premiere etape de l'alignement. Le modele est fine-tune sur des exemples de conversations (instruction, reponse ideale) rediges par des humains.
- **RLHF - Reward Model** : des annotateurs classent plusieurs reponses du modele par preference. Un Reward Model est entraine sur ces comparaisons pour predire le score d'une reponse.
- **RLHF - Optimisation PPO** : le LLM est optimise avec l'algorithme PPO (Proximal Policy Optimization) pour maximiser le score du Reward Model. C'est ce qui a cree ChatGPT.
- **DPO (Direct Preference Optimization, 2023)** : alternative plus simple au RLHF. Apprend directement depuis des paires (reponse preferee, reponse rejetee) sans Reward Model explicite. Plus stable.
- **Constitutional AI (Anthropic)** : le modele s'auto-critique selon un ensemble de principes (la 'constitution'). Reduit le besoin d'annotations humaines pour l'alignement.
- **RLAIF** : Reinforcement Learning from AI Feedback. Un LLM plus puissant joue le role d'annotateur pour generer les preferences d'entrainement a grande echelle.

Fine-tuning efficace : LoRA et QLoRA

- Probleme : fine-tuner un LLM de 70 milliards de parametres en pleine precision necessite plusieurs dizaines de GPU A100. C'est inaccessible pour la grande majorite des organisations.
- LoRA (Low-Rank Adaptation, 2022) : au lieu de modifier tous les poids, on injecte de petites matrices de faible rang (A x B) dans les couches d'attention. Seulement 0.1 a 1% des parametres sont entraines, pour un cout reduit de 95%.
- QLoRA (2023) : LoRA + quantification 4-bit du modele de base. Permet de fine-tuner Llama 3 70B sur un seul GPU de 48 Go, accessible sur une seule machine professionnelle.
- PEFT (Parameter-Efficient Fine-Tuning) : famille de techniques incluant LoRA, Prefix Tuning, Prompt Tuning et les Adapters. Bibliotheque HuggingFace standardisee pour tous ces methodes.
- Cas d'usage concret : adapter un LLM generaliste a un domaine metier (juridique, medical, cybersecurite, code) avec seulement quelques milliers d'exemples annotes, en quelques heures sur du materiel accessible.

Hallucinations et Evaluation

01 Definition et types

02 Les Stratégies d'atténuation

03 La Évaluation des LLMs

04 En production

Les hallucinations : causes et stratégies d’atténuation

- Definition : un LLM 'hallucine' lorsqu'il genere des informations factuellement incorrectes mais formulees avec confiance et coherence. L'une des limites les plus critiques pour les applications reelles.
- Types : factuelles (faits inventes - dates, citations, references), contextuelles (incoh erence avec le contexte fourni), intrins eques (contradiction interne dans la meme reponse).
- Causes : le modele est entraine a predire le token le plus probable, pas a 'dire la verite'. Il extrapole et interpole au-dela de ses connaissances, surtout sur des sujets rares ou des details precis.
- Strategies d'attenuation : RAG (ancrer la reponse dans des sources verifiees), auto-critique ('verifie ta reponse etape par etape'), citations obligatoires avec sources, temperature=0 pour les taches factuelles.
- Detections : NLI (Natural Language Inference) pour verifier la coherence, SelfCheckGPT (generer plusieurs fois et comparer), outils specialises comme Guardrails AI, LLM-Guard ou RAGAS faithfulness.

Évaluer un LLM en production

L'evaluation des LLMs est un domaine en pleine evolution. Les metriques classiques sont insuffisantes pour capturer la qualite des reponses generatives.

- **Metriques classiques (BLEU/ROUGE)** : mesurent le chevauchement de n-grammes entre la reponse generee et une reference. Adaptees au resume et a la traduction, mais blind aux paraphrases semantiquement correctes.
- **LLM-as-a-Judge** : utiliser un LLM puissant (GPT-4, Claude 3.5) pour evaluer la qualite d'un autre LLM. Mesure coherence, fidelite, pertinence, utilite. Correle bien avec les preferences humaines.
- **RAGAS** : framework specialise pour les pipelines RAG. Mesure : faithfulness (la reponse est-elle fidele aux sources ?), answer relevancy, context precision, context recall.
- **Chatbot Arena (LMSYS)** : humains comparent deux modeles anonymes et choisissent le meilleur. Classement Elo continu. Reference non gamifiable pour l'evaluation en conditions reelles.
- **Red Teaming et tests adversariaux** : equipe qui cherche a faire echouer, biaiser ou faire deriver le modele intentionnellement. Indispensable avant tout deploiement en production sensible.
- **Metriques metier specifiques** : F1/precision/rappel pour la NER, EM et F1 pour le Q&A extractif, CodeBLEU pour la generation de code, COMET pour la traduction. Adapter l'evaluation au cas d'usage reel.

L’extraction d’information

01 La morphosyntaxe

02 Les entités nommées

03 La syntaxe

04 Les mots clés

L’ étiquetage morphosyntaxique

Je	**mets**	**les**	**mains**	**dans**	**le**	**moteur**	**.**
pronom	verbe	déterminant	nom	adposition	déterminant	nom	ponctuation
je	*mettre*	*le*	*main*	*dans*	*le*	*moteur*	*.*

Appliquer des modèles pour associer une étiquette grammaticale à chaque «*token* »

Les etiquettes grammaticales

Utiliser un dictionnaire pour extraire le lemme associé au couple (*token(s)*=forme fléchie, *Part Of Speech Tag*= Etiquette *grammaticale*)

Les lemmes

Morphologie

- Analyse morphologique traite de l'analyse des morphèmes (unité linguiste porteuse de sens). Notez qu'un morphème peut avoir plus graphie (allomorphe) :

  Par exemple : *tu t'assois / tu t'assieds ; il s'assoit / il s'assied ; je paye / je paie etc. (source: wikipedia)*

- Plusieurs opérations peuvent être appliquées sur les morphèmes
  - Flexion : conjugaison, mise au féminin, au pluriel …
  - Dérivation : ajout d'affixe

prefixes		radical	suffixes		
anti	con-	-stitu-	-tion	-nelle	-ment

(source: wikipedia)

  - Composition, par exemple « bonhomme » : bon + homme

Analyse morphologique

- Racinisation (stemming) : extraire la racine d'un mot en supprimant sa terminaison :

- Lemmatisation : retrouver la forme canonique du morphème

Forme Fléchie	Lemme	« racine »
Chant	Chant	Chant
Chante	Chanter	**Chant**

La racinisation peu avoir quelques biais, mais elle est utilisée intensivement dans les moteurs de recherche ou encore les systèmes de classification documentaire

- Décomposition: segmenter un mot en unités le composant

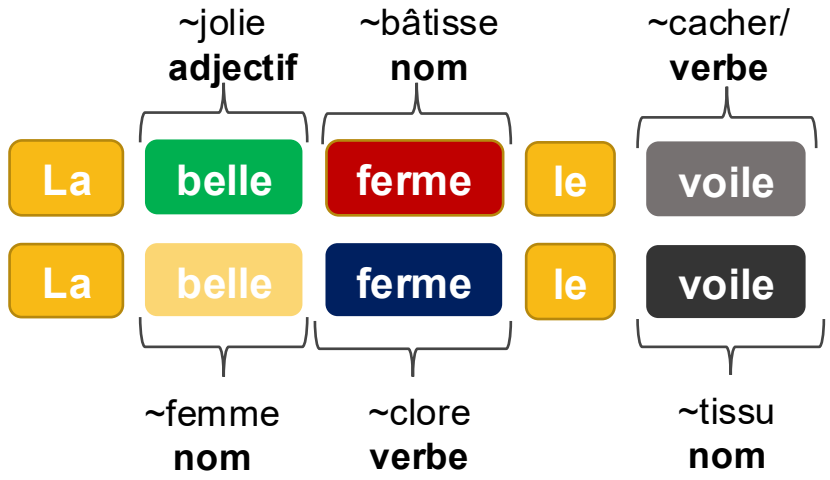
La morphosyntaxe

La morphosyntaxe traite de l'étiquetage de mots par des étiquettes grammaticales : Nom, Verbe, Adjectifs …

- Permet le calcul de lemmes (plus précis que la racinisation
- Facilite la détection des mots « outils » : plus précis qu'une liste de mots outils (*stopwords)*
- 2 classes d'étiquettes
  - Catégories fermées : les déterminants, conjonctions, les prépositions, les nombres, les auxiliaires
  - Catégories ouvertes : noms, verbes, adjectifs et les adverbes
- Un mot peu avoir plusieurs étiquettes grammaticales et donc plusieurs formes canoniques possible (cela peut induire des sens différents)

Forme Fléchie	Lemme	« racine »
Chant (Noun)	Chant	Chant
Chante (Verb)	Chanter	**Chant**

Mot	Categorie	« StopListé »
Or	Déterminant	OUI
Or (metal)	Nom	**NON**



Un premier transformer pour l'étiquetage

Entrée
'Le chat mange une souris et boit du lait.

Tokenizer
</s> _Le _chat _mange _une _so uris _et _bo it _du _lait . </s>

Notons que les '_' au début des tokens donne le premier « sous-mots ». Ainsi le verbe « boit » est composé de 2 sous-mots : '_bo' et 'it'

Modèle
Le modèle fourni une matrice (de logits) ou pour chaque ligne il faudra prendre la valeur maximale. L'indice de la colonne correspond à une étiquette

Token	ADJ	ADP	ADV	AUX	CCONJ	DET	INTJ	NOUN	NUM	PART	PRON	PROPN	PUNCT	SCONJ	SYM	VERB	X
_Le	0,251	1,181	-0,889	-1,459	-0,578	**10,096**	-0,703	-1,193	0,487	-1,071	0,066	-0,524	-1,342	-0,637	-0,944	-1,655	-1,047
_chat	1,205	-1,857	-0,413	-1,356	-1,252	-1,225	-1,044	**9,499**	-1,163	-1,490	-0,405	2,031	-1,426	-1,886	-0,497	-0,539	0,270
_mange	1,726	-0,724	-1,334	0,453	-1,622	-2,377	-1,027	1,103	-1,244	-1,132	-1,247	-0,090	-2,115	-1,593	-1,197	**9,277**	-0,350
_une	0,166	1,631	-0,773	-1,287	-0,480	**9,437**	-0,832	-1,754	1,273	-0,899	0,847	-1,375	-1,297	-0,518	-0,971	-1,861	-1,218
_so	1,493	-1,459	-1,242	-1,538	-1,536	-1,103	-1,305	**8,893**	-0,800	-1,713	-0,270	1,973	-1,647	-1,912	-0,535	0,267	0,639
uris	2,162	-1,641	-0,976	-1,743	-1,575	-1,326	-1,176	**8,428**	-0,595	-1,791	-0,480	2,149	-1,664	-1,930	-0,587	0,165	0,247
_et	-1,368	-0,282	0,320	0,012	**9,623**	-0,470	-0,419	-0,988	-1,319	-0,596	-0,426	-1,482	-1,071	0,027	0,234	-0,841	-0,365
_bo	1,577	-0,846	-1,330	0,313	-2,277	-2,249	-1,260	1,306	-1,004	-1,547	-1,087	0,242	-1,851	-1,340	-1,107	**8,913**	-0,147
it	0,933	-0,713	-1,116	1,922	-1,077	-2,331	-1,084	0,795	-1,397	-1,083	-0,585	-0,633	-1,663	-1,583	-1,464	**9,037**	-0,673
_du	0,362	2,493	-0,841	-1,634	-0,603	**9,095**	-1,093	-1,475	0,956	-1,122	0,179	-1,384	-1,293	-0,653	-0,740	-1,145	-0,840
_lait	1,288	-1,817	-0,394	-1,576	-0,977	-1,393	-0,977	**9,200**	-1,203	-1,324	-0,451	1,347	-1,724	-2,081	-0,419	0,255	0,346
,	-0,462	0,090	-0,007	-0,783	0,136	0,137	-1,159	0,015	-0,677	-0,836	0,115	-1,262	**7,744**	-0,219	1,427	-0,920	-0,677

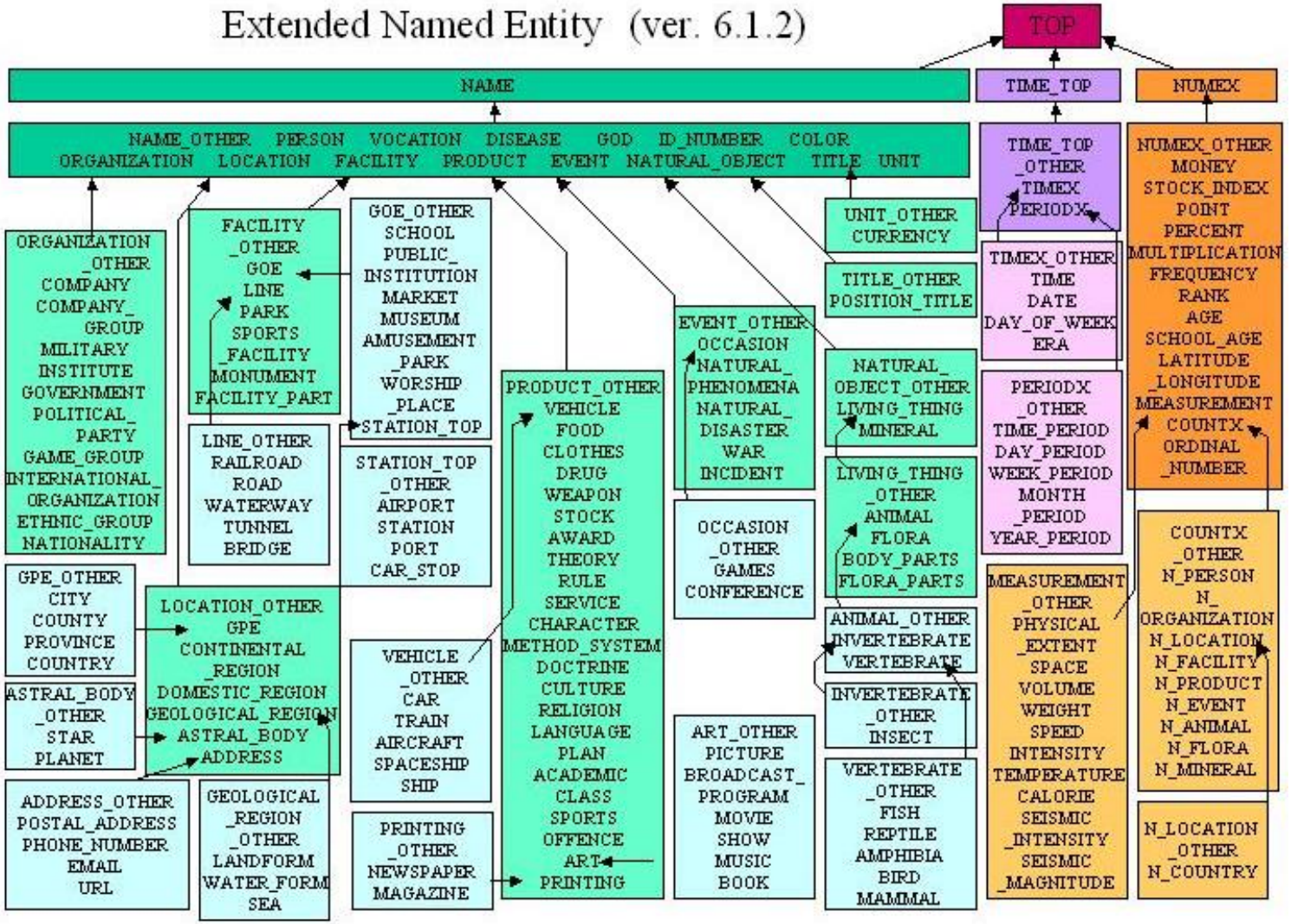
L’Extraction d’entités nommées

« À l'instar de Roberto Martinez PER , il nous reste un goût amer dans la bouche. « C’est vraiment dommage qu’on n’ait pas un stade national pour pouvoir célébrer cette incroyable génération de joueurs pendant l’Euro… ». Et le journal de poursuivre : « C’est même particulièrement navrant. Tête de série, la Belgique LOC aurait pu revendiquer au minimum deux matchs à domicile dans un « Eurostadium LOC » flambant neuf, sis sur le parking C du Heysel LOC . Un projet qui a coûté plus de 25 millions d’euros MONEY pour, in fine, échouer lamentablement, entre querelles à différents niveaux de pouvoir et à relents communautaires. « Retirée par l’UEFA des villes hôtes en décembre 2017 DATE , Bruxelles LOC n’aura donc pas de stade national… et n’en aura sans doute jamais. Aucun gouvernement – à condition déjà d’en avoir un – ne semble près à s’accorder à ce niveau. (…) Bref, une véritable « histoire belge », notre royaume étant, à ce stade, la risée de l’Euro(pe). À charge désormais des Diables LOC de nous éviter de l’être sur le terrain. Y compris à l’autre bout du continent », conclut le journal.

Difficultés de l’extraction d’entités nommées

- **Variabilité des noms propres** : les noms propres sont souvent sujets à des variations orthographiques, à des abréviations, à des acronymes et à des formes alternatives. Cela demande une capacité à reconnaître et à généraliser ces variations.
- **Ambiguïtés contextuelles** : certains mots peuvent être utilisés à la fois comme noms propres et comme mots ordinaires. Distinguer ces cas nécessite une compréhension fine du contexte.
- **Entités nouvelles ou rares** : les modèles doivent être capables de gérer des entités qui ne sont pas présentes dans leur jeu de données d'entraînement.
- **Entités ambiguës** : certains noms propres peuvent être associés à plusieurs entités. Le bon choix dépend du contexte, de la géographie et des connaissances générales.
- **Langues à flexion libre** : dans certaines langues, comme l'anglais, les noms propres n'ont pas de marques morphologiques spécifiques. Cela signifie que les modèles doivent s'appuyer davantage sur le contexte pour les identifier.
- **Étiquetage des frontières d’entités** : il faut déterminer précisément les débuts et les fins des entités nommées dans un texte. Cela peut être difficile lorsque les entités se chevauchent ou sont mal formatées.
- **Séparation d’entités composites** : certains noms propres sont composés de plusieurs mots. Distinguer ces entités des séquences de mots non entières est essentiel.
- **Manque de standardisation** : les entités nommées peuvent être exprimées de différentes manières dans des domaines spécifiques. Par exemple, les produits technologiques peuvent avoir des noms de marque variés qui peuvent être difficiles à catégoriser de manière cohérente.

Type d’entités nommées



Hiérarchie de Sékine

Un premier transformer pour l'extraction d'entités nommées

Entrée
'Raymond Devos est un humoriste français'

Tokenizer
<s> _Raymond _de vos _est _un _ humoriste _français . </s>'

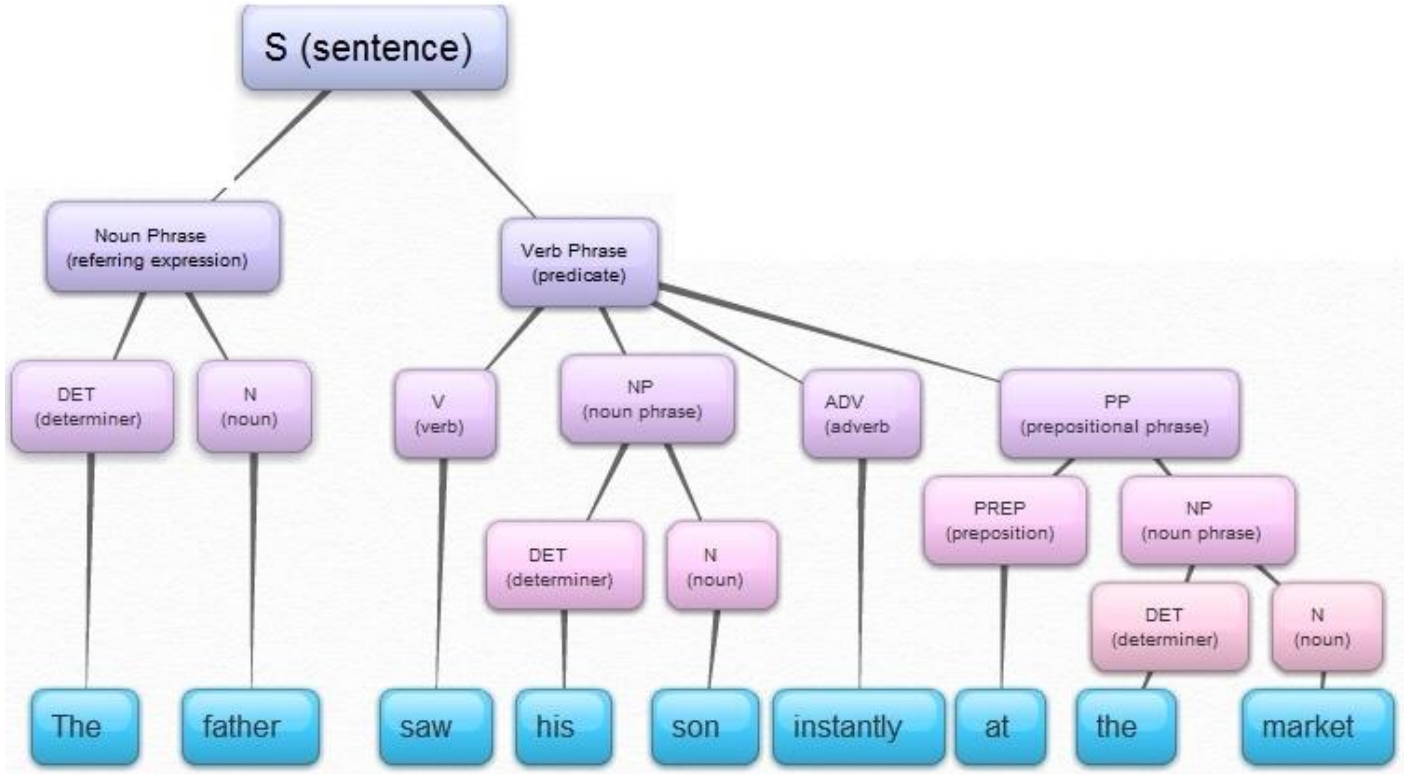
Modèle
Le modèle fourni une matrice (de logits) ou pour chaque ligne il faudra prendre la valeur maximale. L'indice de la colonne correspond à une étiquette

Tokens	O	I-LOC	I-PER	I-MISC	I-ORG
_Raymond	-0,566	-2,405	**6,723**	-0,532	-2,085
_De	-0,949	-2,001	**7,008**	-0,735	-1,760
vos	-0,973	-1,955	**7,017**	-0,733	-1,749
_vit	6,596	-2,597	-4,161	-2,448	-3,794
_à	5,125	-0,427	-4,402	-2,104	-4,149
_Chev	-2,430	**6,479**	-1,016	-0,812	-1,815
r	-2,378	**6,480**	-1,043	-0,854	-1,859
euse	-2,390	**6,490**	-1,048	-0,850	-1,831
.	6,321	-2,912	-2,702	-3,514	-2,883

Pas vraiment de surprise, le principe est exactement identique à l'étiquetage morphosyntaxique.

Ici un étiquetage avec une notion de début (**B-**) et d'élément interne (**I-**) sont mis en oeuvre

L’ étiquetage syntaxique



POS tagging

Segmentation et/ou morphosyntaxe

Regroupement hiérarchique

Les etiquettes syntaxique

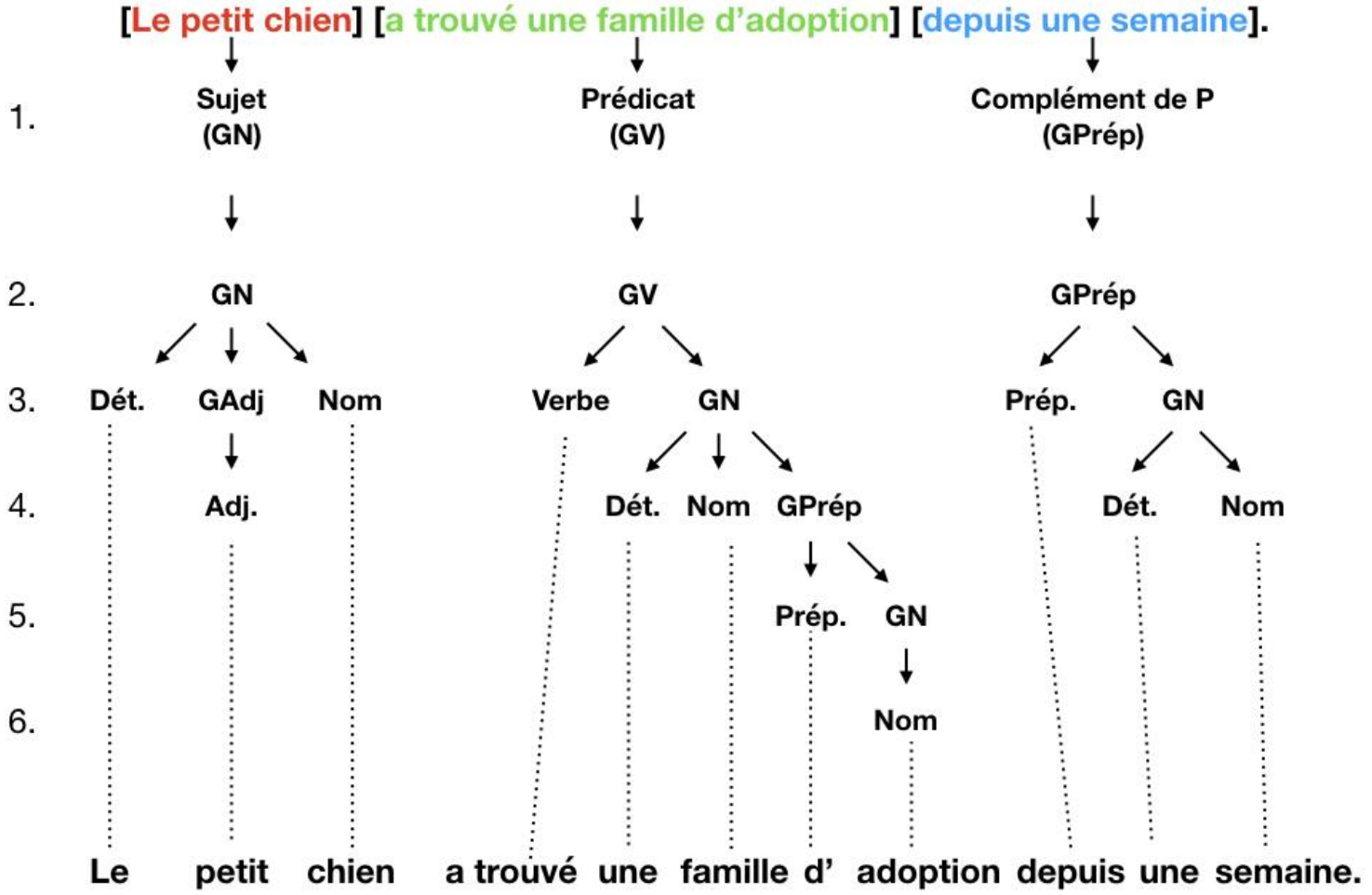
Analyse Syntaxique superficielle (*chunking*)

- L'analyse syntaxique de surface se concentre sur la structure apparente des phrases.
- Elle décompose les phrases en unités syntaxiques visibles immédiatement.
- Elle identifie les sujets, verbes, objets et autres éléments syntaxiques évidents.
- Utilise des méthodes statistiques et des modèles d'apprentissage automatique.
- Se base sur les parties du discours et les dépendances entre les mots.
- Génère des arbres syntaxiques peu profonds pour décrire la structure de surface.

Type de syntagmes	Noyau	Exemple *(source: wikipedia)*
Syntagme nominal (SN)	Nom	[*Le petit* ***chien***$^{\text{Nom}}$ *blanc de mon voisin*]$^{\text{SN}}$ *a aboyé* toute la nuit.
Syntagme verbal (SV)	Verbe	Sophie [***aime***$^{\text{Verbe}}$ *beaucoup le chocolat*]$^{\text{SV}}$
Syntagme adjectival (SA)	Adjectif	Cette île est [la plus **belle**$^{\text{adjectif}}$ du monde]$^{\text{SA}}$
Syntagme adverbal(Sadv)	Adverbe	Ils ont dû payer une amende [***conformément***$^{\text{Adverbe}}$ *à la loi*]$^{\text{Sadv}}$.

Analyse syntaxique (parsing, *syntax analysis*)

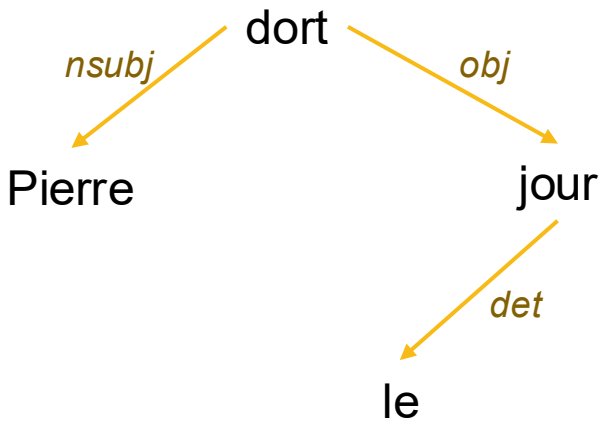
Analyser la phrase pour construire un arbre syntaxique, relier des syntagmes et créer des groupement hiérarchiques complexes



(source: http://www.alloprof.qc.ca/BV/Pages/f1126.aspx)

Analyse Dépendances (dependencies analysis)

Analyser la phrase pour construire un arbre dont les nœuds sont les mots. Les liens entre ces nœuds étant typés. Cet arbre met en évidence les relations entre les mots d'une phrase.



lien	description
obj	object
nsubj	sujet nominal
det	déterminant

Extraction de mots clés

Fonctions

- Identifie les termes pertinents d'un texte.
- Résume et catégorise le contenu de manière concise.
- Représentent les sujets principaux et l'essence du texte.

Les techniques incluent l'analyse statistique et l'utilisation de modèles de langage; ces dernier, construits avec des Transformers, améliorent la précision.

Leur utilisation facilite la recherche, la classification et la compréhension des documents.

Un premier transformer pour l'extraction de mots clés

Entrée
'Raymond Devos vit à Chevreuse.'

Tokenizer
</s> _Raymond _De vis _vit _à _Chev r euse . </s>

Modèle
Le modèle fourni une matrice (de logits) ou pour chaque ligne il faudra prendre la valeur maximale. L'indice de la colonne correspond à une étiquette

Token	O	B-KEY	I-KEY
_Raymond	-1,178	**3,804**	-1,844
_de	-1,797	-1,103	**3,312**
vos	-1,479	-2,157	**3,766**
_est	**4,716**	-3,386	-3,552
_un	**4,642**	-3,033	-3,775
_	-1,847	**3,981**	-1,110
humoriste	-1,562	-2,035	**3,768**
_français	**1,404**	0,276	-2,504
.	**4,092**	-3,152	-2,907

Pas vraiment de surprise, le principe est exactement identique à l'étiquetage morphosyntaxique

Recherche d'information

01 Base de données relationnelles

02 Les bases graphes

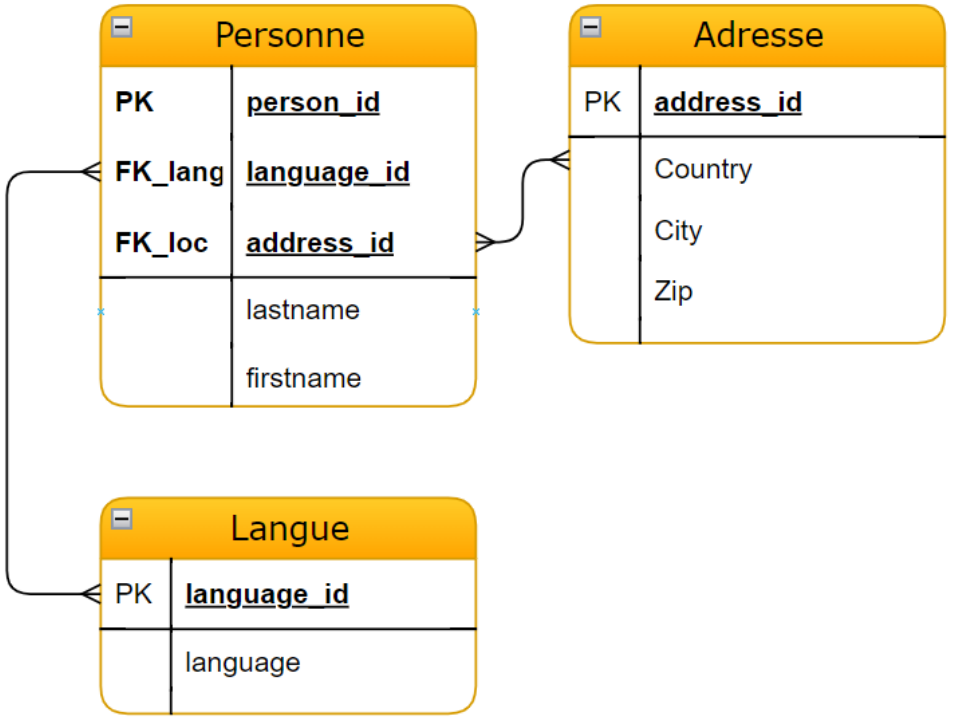
03 Les index inversés

04 Les bases vectorielles

05 Les agrégateurs et les méthodes de fusion

Les bases de données relationnelles

- **Les tables** organisent les données.
- **Les clés primaires** identifient de manière unique chaque enregistrement.
- **Les clés étrangères** établissent les relations entre les tables.
- **Les jointures** permettent de combiner les données de plusieurs tables en fonction des relations établies à l'aide des clés primaires et étrangères
- **Les opérations de requête** sont écrites en SQL (Structured Query Language) et sont utilisées pour interroger et extraire des données de la base de données



Des transformers aux langages structurés comme le SQL

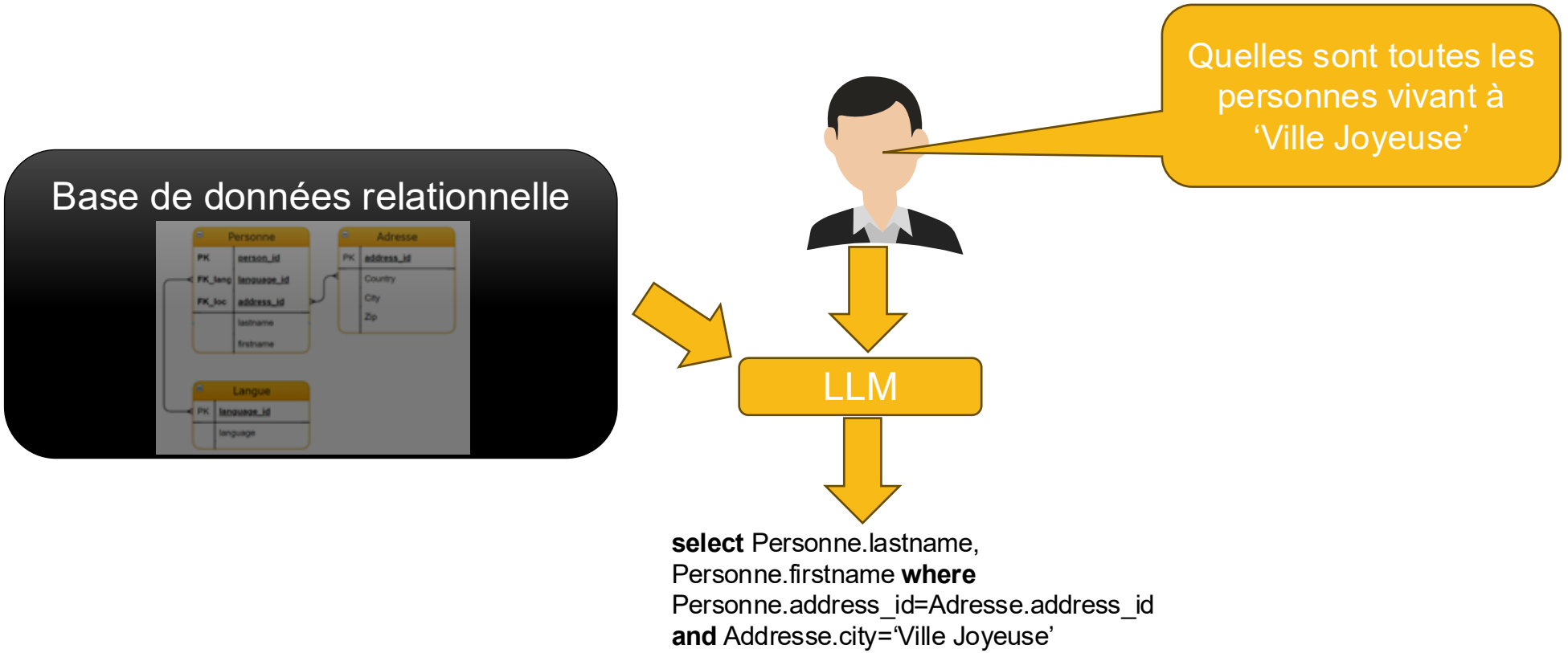
Les Transformers ouvrent de nouvelles perspectives pour l'analyse et l'exploitation de données structurées. Ils peuvent comprendre et traiter des requêtes complexes formulées en langage naturel, permettant une interaction plus intuitive et puissante avec les bases de données.

Cette convergence technologique offre

- Des requêtes plus naturelles et une meilleure compréhension des intentions de l'utilisateur.
- Une réponse plus précise aux demandes complexes.
- Des possibilités d'automatisation des rapports, de prédiction de requêtes et d'optimisation des performances.

Exemple avec ChatGPT 3.5

Base de données relationnelles & *Large Languages Models (LLM)*

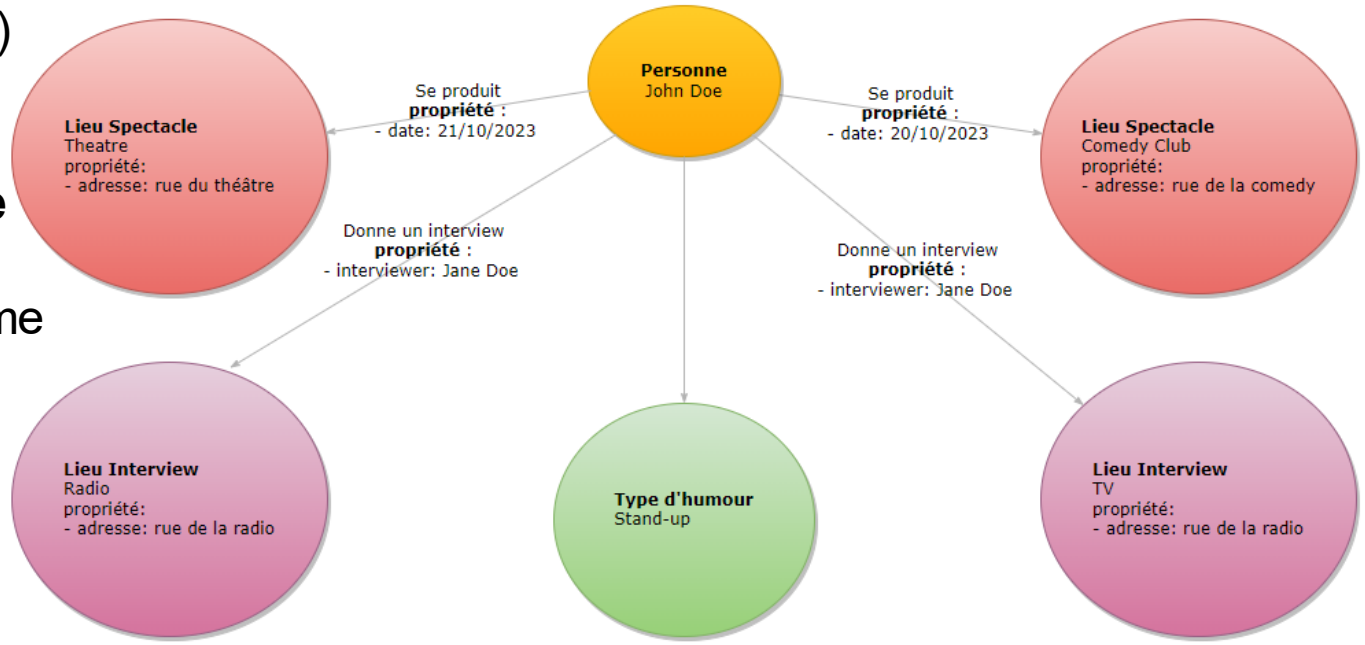


Les bases graphes

Une base de données graphe représente et le stock de données sous forme de graphes. Ces derniers sont des ensembles de nœuds (entités) reliés par des arêtes (relations)

Composants d'une Base de Données Graphe

- **Les nœuds** représentent les entités du système à modéliser (ex. : personnes, lieux).
- **Les arêtes** établissent des liens entre les nœuds, représentant ainsi les connexions ou relations entre les entités (ex. : amitiés entre personnes).
- Chaque nœud et arête peuvent posséder des **propriétés** (ex. : "nom" et "âge" pour un nœud "personne").



D’une base graphe au langage naturel

La génération de langage naturel à partir d'une base de données graphe, en utilisant des modèles génératifs tels que les Transformers, représente une avancée significative dans le domaine de la génération de texte. Ce processus permet de convertir les données stockées dans la base graphe en descriptions fluides et informatives.

Cela trouve des applications particulièrement intéressantes dans la génération automatique de descriptions d'événements, de produits et autres informations basées sur des données structurées.

Exemple Concret
Supposons que nous disposions d'une base graphe représentant des artistes humoristes, les lieux de leurs représentations, les lieux d'interviews et les types d'humour associés. En utilisant un modèle génératif basé sur les Transformers, nous pouvons fournir cette base graphe en entrée, et le modèle générera une description textuelle cohérente de ces relations.

Base Graphe :

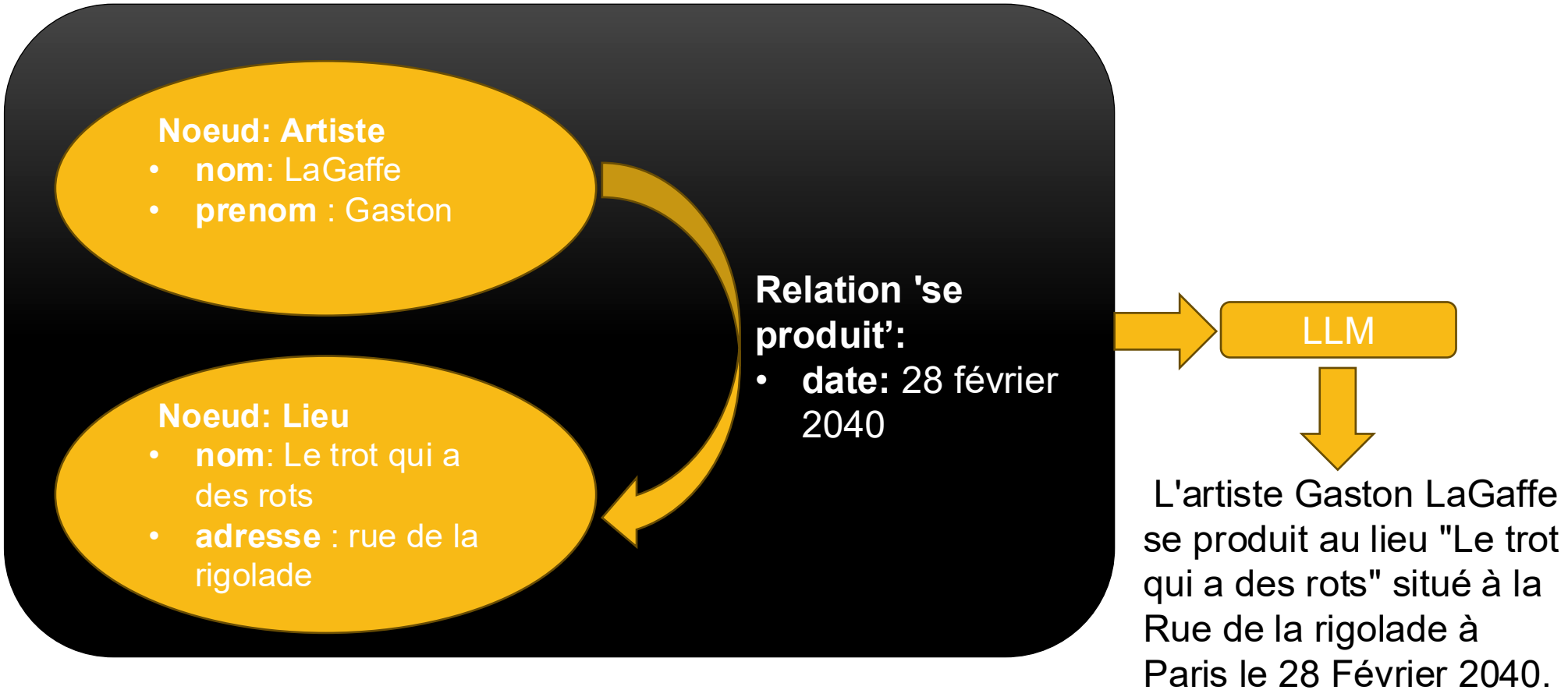
markdown — Copy code

```
Artiste Humoriste : John Doe
  - Âge : 35
  - Type d'Humour : Stand-up

Lieux de Spectacle : Comedy Club XYZ
  - Adresse : 123 Rue de la Comédie
  - Capacité d'Accueil : 150 personnes

Se Produit : John Doe au Comedy Club XYZ
  - Date du Spectacle : 10/11/2023
  - Heure du Spectacle : 20h00
  - Prix du Billet : 20€
```

Texte Généré :

"John Doe, un humoriste de 35 ans, se produira au Comedy Club XYZ le 10 novembre 2023 à 20h00. Les billets pour le spectacle seront au prix de 20€."

Base Graphe & *Large Languages Models*

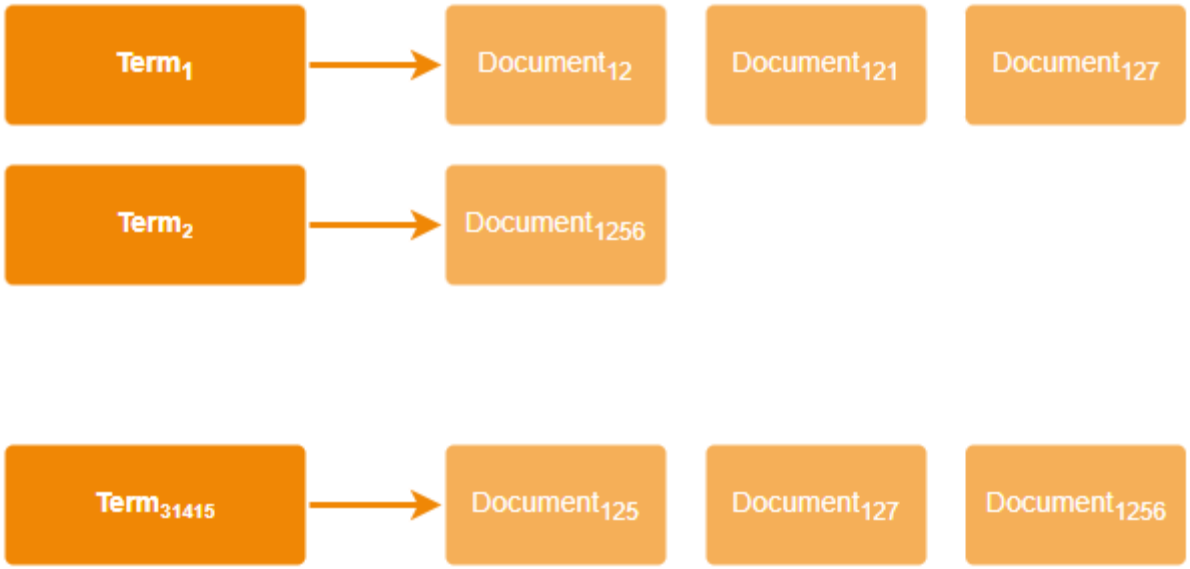


Les Index inversés

Avec la prodigieuse quantité de données générées chaque jour, les moteurs de recherche sont devenus des incontournables pour naviguer dans cet océan d'informations.

Au cœur de ces systèmes se trouve un élément essentiel : l'index inversé. Cette structure de données sophistiquée permet un accès rapide à l'information.

Un index inversé fonctionne comme un vaste catalogue de mots-clés, mais avec une subtilité ingénieuse : il pointe vers les mots et les documents où ils apparaissent.



Ces moteurs de recherche basés sur les index inversés sont l'ossature des recherches en ligne que nous utilisons au quotidien. Ils permettent des requêtes complexes, englobant des milliers de termes, en quelques millisecondes.
L'efficacité de ces moteurs repose sur une combinaison complexe d'algorithmes d'indexation, de traitement du langage naturel, et d'optimisation des requêtes.

L’apport des Transformers

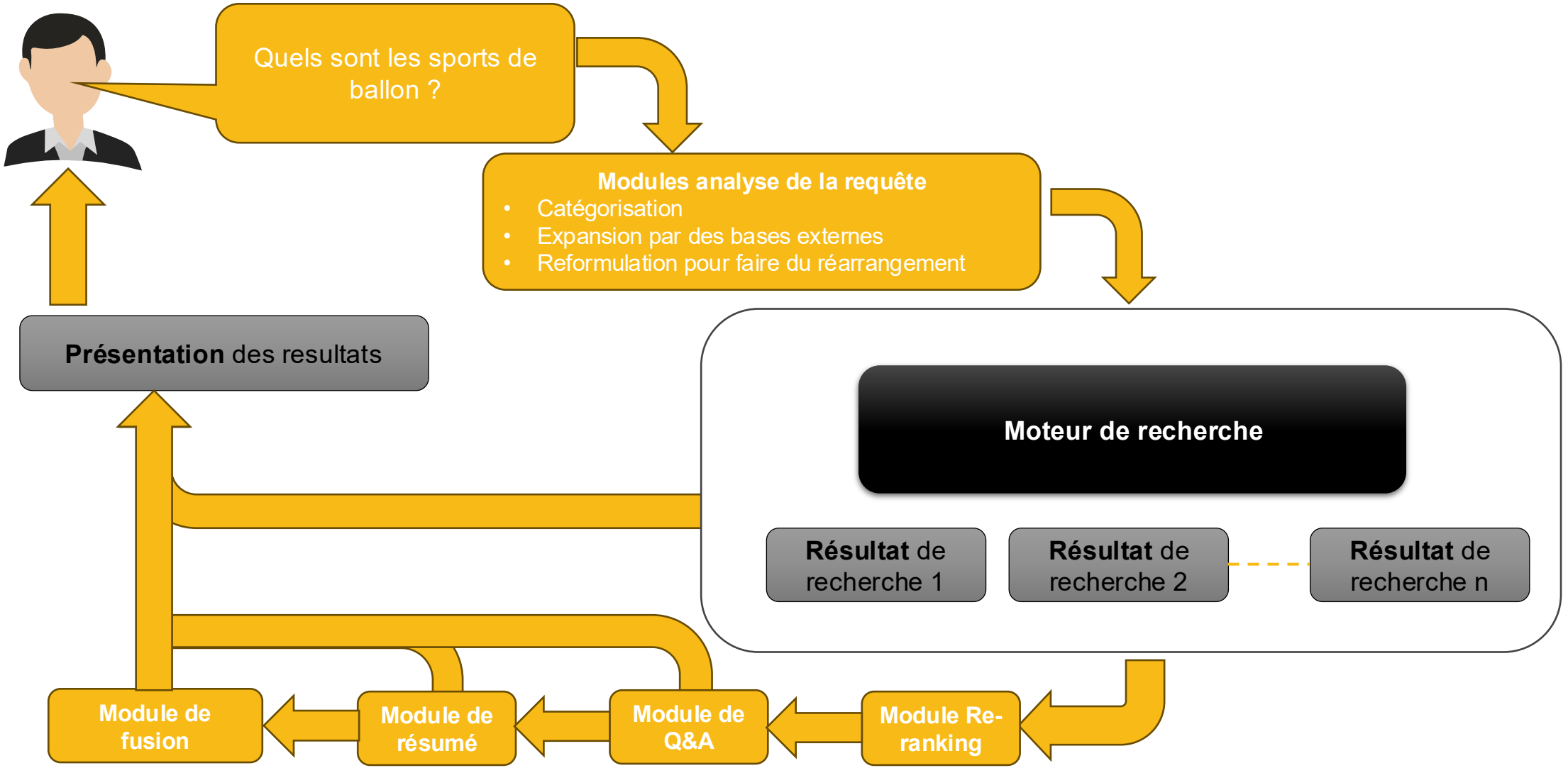
Pré-traitement avec les transformers

- **Analyse sémantique avancée** : les Transformers améliorent l'analyse sémantique des requêtes en comprenant les nuances du langage.
- **Traitement des synonymes et des variantes** : les Transformers gèrent les synonymes et les variantes de mots. Ils identifient des mots similaires ou équivalents, élargissant la recherche au-delà des termes exacts.
- **Correction d’orthographe et gestion des erreurs** : grâce à leur compréhension contextuelle, les Transformers corrigent les erreurs de saisie et les fautes d'orthographe dans les requêtes

Post-traitement avec les transformers

- **Raffinement des résultats** : les Transformers permettent de raffiner les résultats en fonction du contexte de la requête, en prenant en compte des éléments tels que la proximité des mots ou l'importance relative des termes, améliorant ainsi la pertinence des résultats.
- **Adaptation au contexte dynamique** : les Transformers s'adaptent en temps réel aux changements dans le contenu indexé.
- **Optimisation des résultats** : Les Transformers aident à classer les résultats en fonction de leur pertinence réelle pour l'utilisateur. Les résultats sont triés en fonction de leur importance dans le contexte de la requête, plutôt que simplement en fonction de la fréquence des mots.

Search engine with NLP pipeline based on transformers



Les bases vectorielles

Ces systèmes sont optimisés pour stocker et interroger des données sous forme de vecteurs numériques.
Utiles en vision par ordinateur, traitement du langage naturel, recommandation de produits, etc.

Techniques d'Indexation

- KD-Tree
- LSH (Locality-Sensitive Hashing)
- Product Quantization
- R-Trees
- Random Projection
- Hierarchical Navigable Small World

Impact des Transformers

- Extraction de caractéristiques avancées
- Contextualisation avancée
- Traitement des requêtes complexes
- Amélioration de la pertinence des résultats

Vector store avec les espaces latents des transformers

Documents à stocker

Vecteur « documents » ajouter à la base vectorielle

Vector store (FAISS, Chroma, …)

Transformer pour extraire des embeddings

Recherche de document proches de la requête

Transformation de la requête en vecteur (embeddings)

Où est l'avion ?

Vector store avec des espaces latents d'images

Documents à stocker

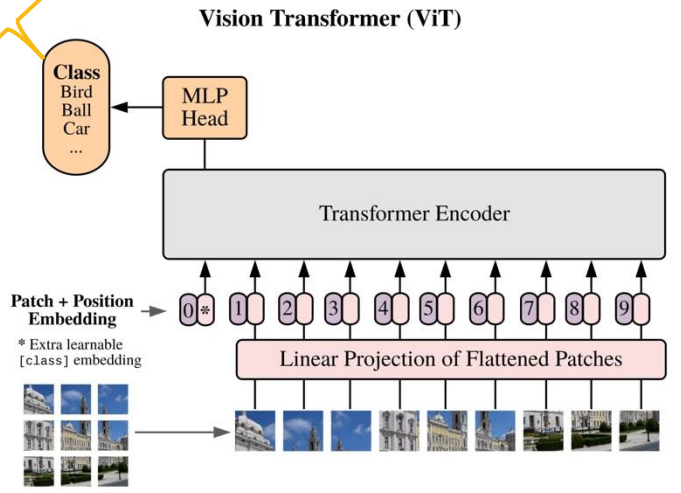
Vecteur « documents » ajouter à la base vectorielle

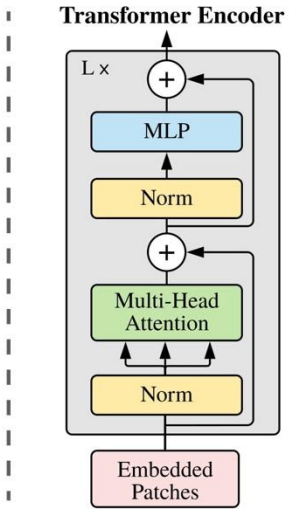
Vector store (FAISS, Chroma, …)

Transformer pour extraire des embeddings

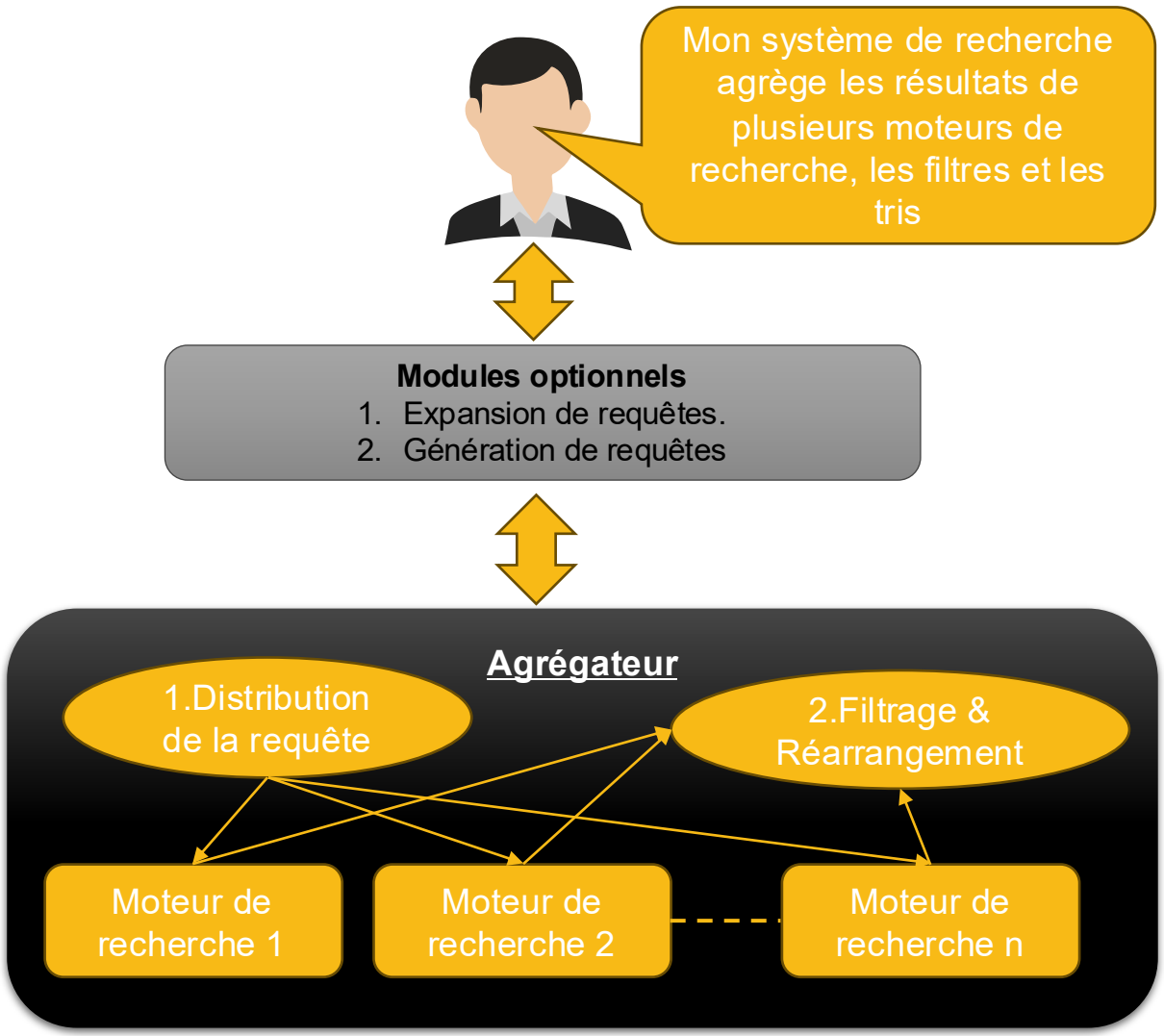
Recherche de document proches de la requête

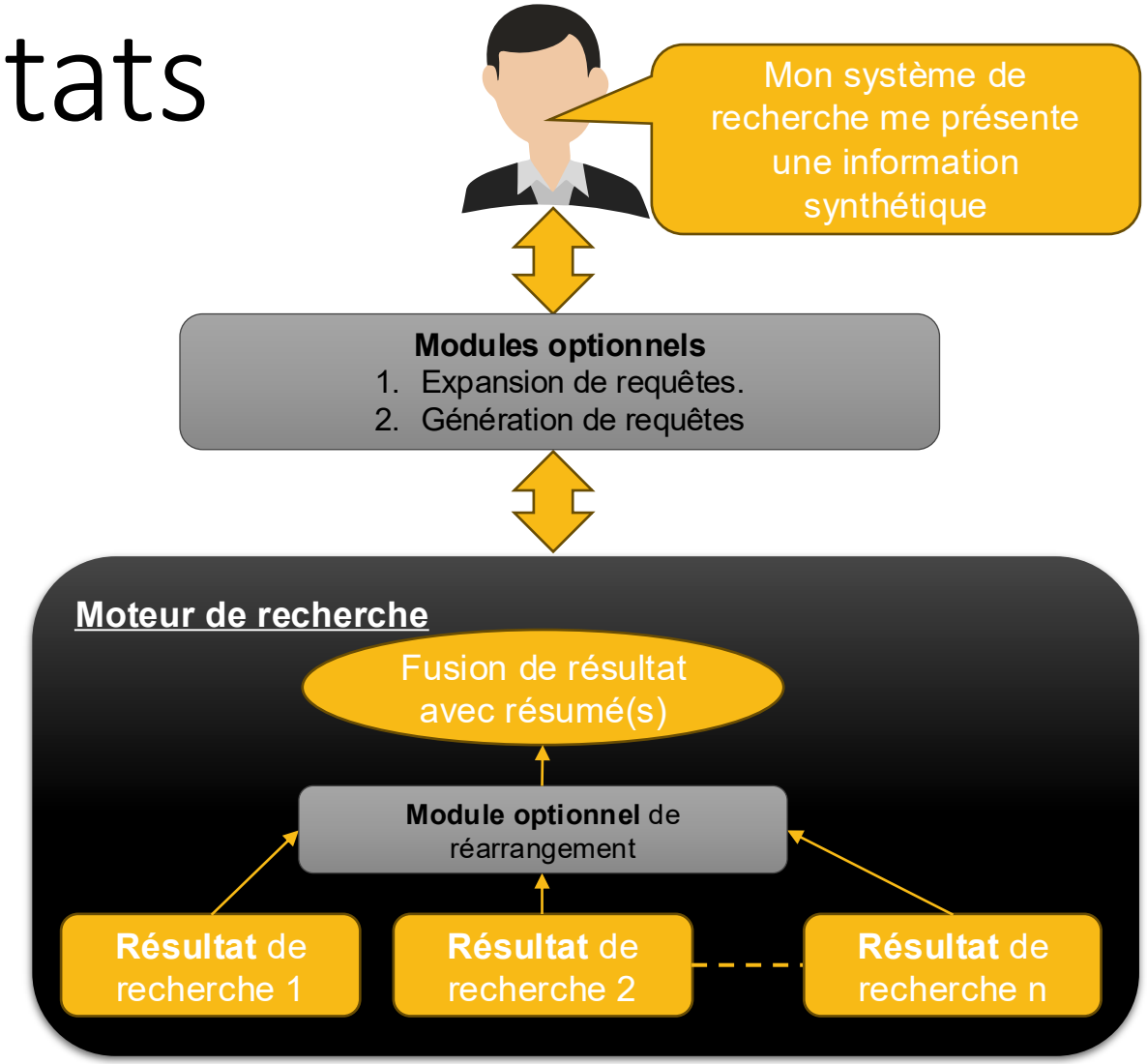
Transformation de la requête en vecteur (embeddings)





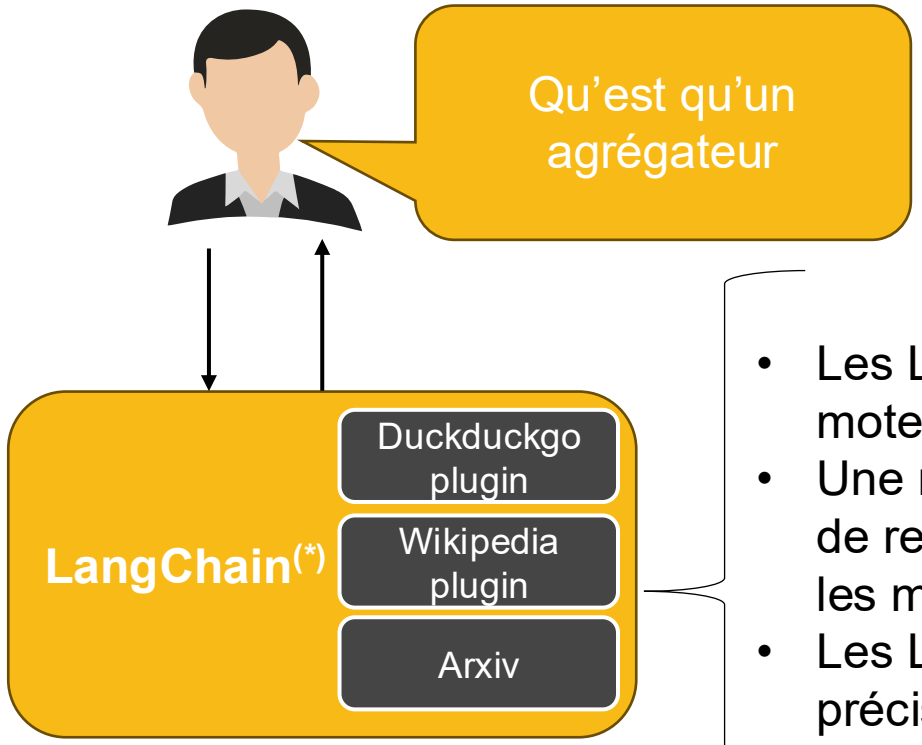
Agrégation de résultat & la fusion de résultats





Avec le module de génération de requêtes & celui de réarrangement, cette architecture porte le nom de **R**etrieval **A**ugmented **G**eneration Fusion (RAG-Fusion)

Agrégateurs un peu évolué – Exemple dans langchain



- Les LLMs sont utilisé pour sélectionner le(s) « bon moteur(s) de recherche »
- Une requête est soumise à un agrégateur de moteurs de recherche (par exemple ***searx***) en sélectionnant les moteurs définis par le LLM
- Les LLMs sont utilisés pour formuler une réponse précise à l'utilisateur
- Il est possible de récupérer la source de la réponse produite

(*) **Langchain** est un framework de manipulation de LLM – il sera détaillé plus précisément dans le chapitre des agents conversationnels

RAG — Retrieval-Augmented Generation

Les LLMs ont une connaissance figée (date de coupure) et hallucinent sur les données propriétaires.

- Au moment de l'inférence, les documents pertinents sont récupérés depuis une base vectorielle (FAISS, ChromaDB, Pinecone…) et injectés dans le contexte du prompt.
- **Pipeline** :
  - *(1) Indexation* : découper les documents en chunks → encoder en vecteurs (modèle d'embedding) → stocker.
  - *(2) Requête* : encoder la question → recherche par similarité cosinus → récupérer les top-k chunks → construire le prompt augmenté → générer.
- **Avantages** : sources citables, mise à jour sans réentraînement, réduction des hallucinations.
- **Outils** : LangChain, LlamaIndex, Haystack.

Les systèmes de questions / réponses

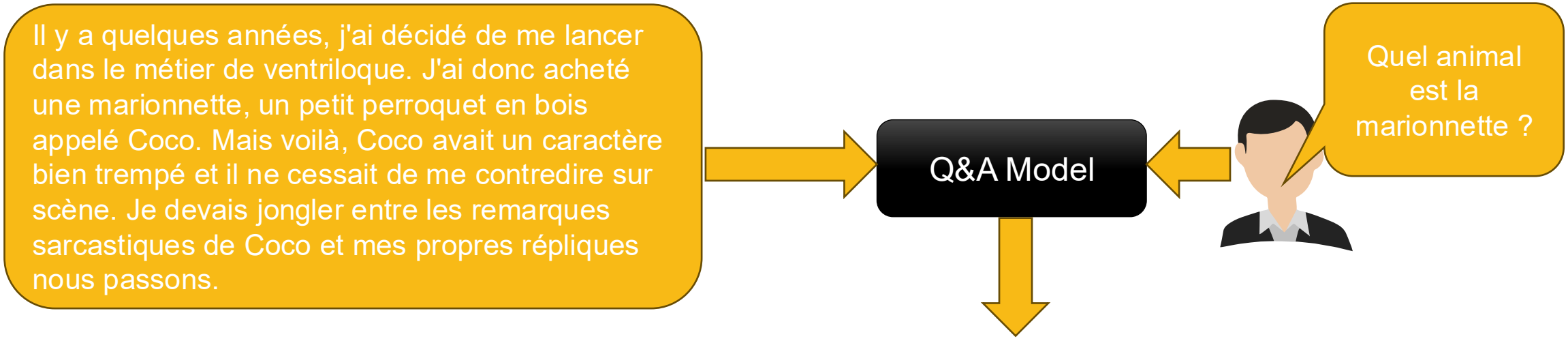
Utilisation des transformers pour les systèmes de question-réponse

Les systèmes de question-réponse exploitent la puissance des Transformers, une architecture d'apprentissage automatique spécialisée dans le traitement de données textuelles. Voici comment ils fonctionnent :

- **Encodage de la question et du contexte** : la question de l'utilisateur et le contexte sont encodés en représentations vectorielles. Les Transformers découpent le texte en "tokens" et les encodent séquentiellement.
- **Création de la représentation ensembliste** : la question et le contexte sont combinés pour créer une représentation ensembliste complète, fournissant au modèle une vue d'ensemble de la requête et des données.
- **Prédiction de la réponse** : le modèle prédit la réponse en identifiant la partie pertinente du texte grâce à des mécanismes d'attention.
- **Extraction de la Réponse** : la portion pertinente est extraite pour la présenter comme réponse.
- **Post-traitement (en option)** : une étape de post-traitement peut affiner la réponse extraite, en supprimant des éléments superflus par exemple.

Les Transformers excellent dans la compréhension des relations complexes entre les mots et les phrases, en faisant d'eux des candidats idéaux pour traiter des questions en langage naturel. Leur capacité à capturer des informations détaillées dans des données séquentielles en fait des outils puissants pour les systèmes de question-réponse.

Système de question-réponse avec les transformers

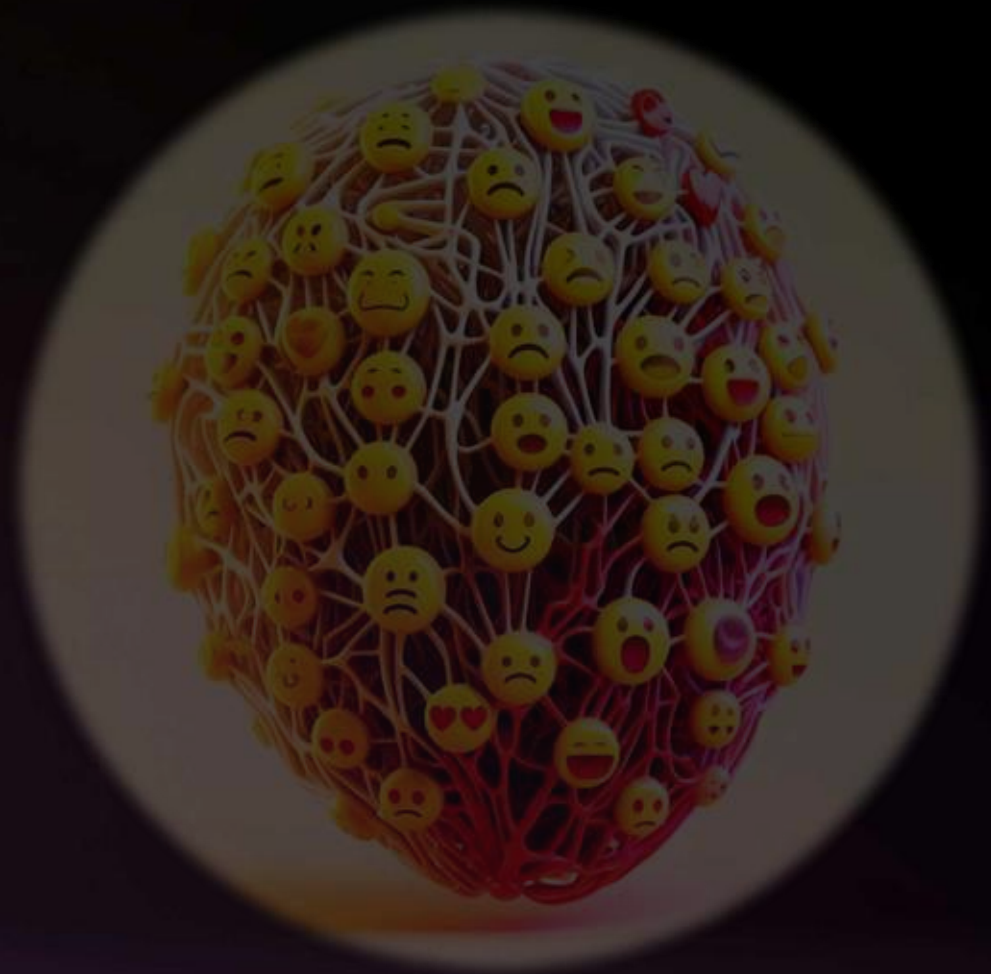


Depuis le pipeline huggingface:

```
{
  'score': 0.1441538780927658,
  'start': 113, 'end': 166,
  'answer': ' un petit perroquet en bois appelé Coco.'
}
```

La sortie du modèle est composée de deux vecteurs de la taille de la séquence : un vecteur contenant la probabilité de début de la réponse et un autre la probabilité de fin de réponse. Le calcul du « span » de réponse se fait entre les deux maximums.

Analyse de sentiments



Utilisation des transformers l’analyse de sentiments

Analyse de Sentiments
- Évalue l'orientation émotionnelle d'un texte.
- Utilise des modèles Transformers pour une compréhension fine du langage.

Génération de Texte avec Sentiments
- Crée du contenu textuel imprégné d'une tonalité émotionnelle spécifique.
- Modèles Transformers comprennent nuances sémantiques et émotionnelles.
- Ouverture à une communication plus riche et humaine avec les machines.

Analyse de sentiments avec les transformers (classification de token)

Entrée
‘Le chat mange de très mauvaises croquettes.

Tokenizer
</s> _Le _chat _mange _de _très _mauvaises _croquette . </s>

Modèle
Le modèle fourni une matrice (de logits) ou pour chaque ligne il faudra prendre la valeur maximale. L’indice de la colonne correspond à une étiquette

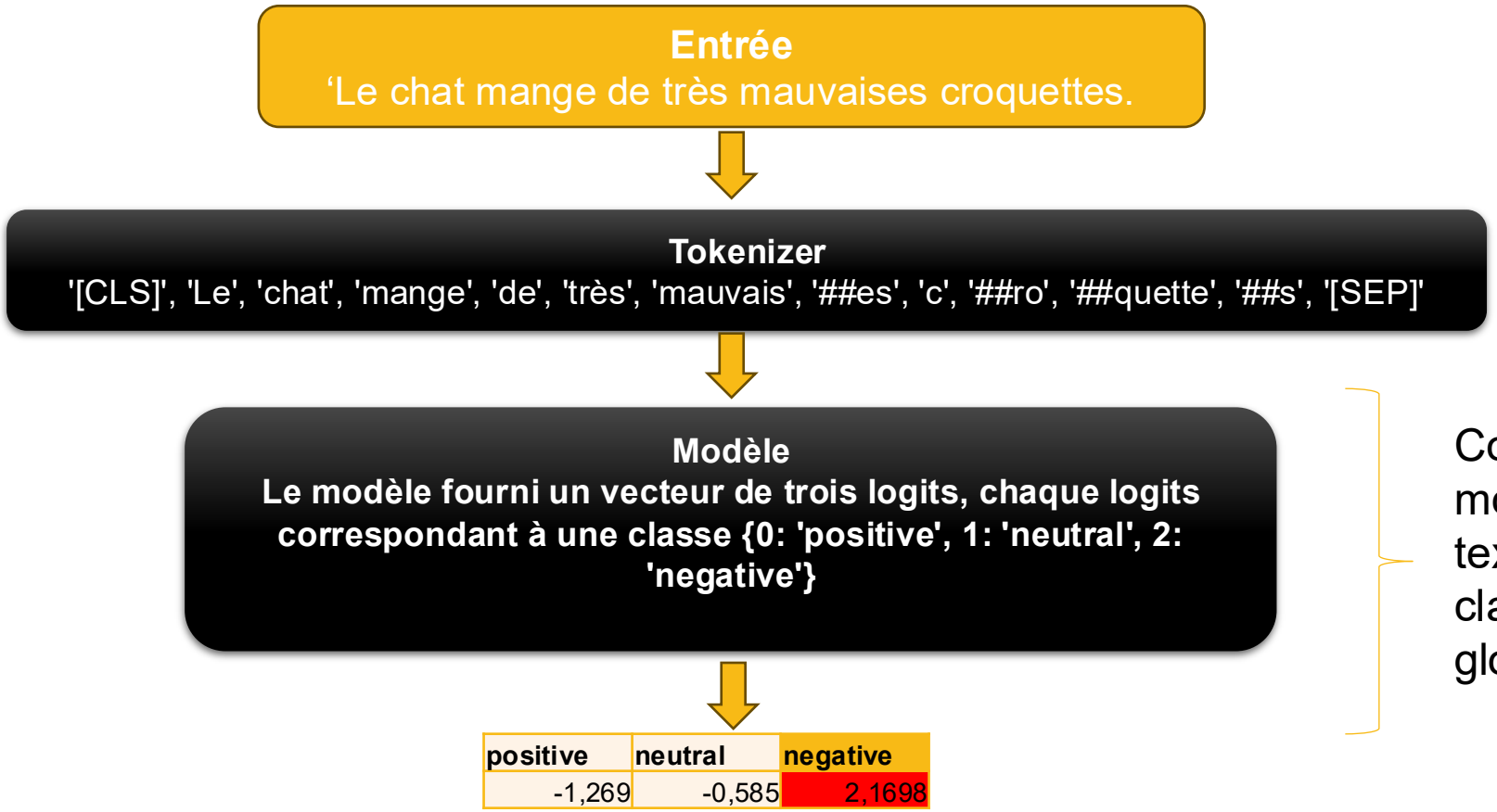
Token	1 star	2 star	3 star	4 star	5 star
_Le	-0,543	0,341	-0,332	0,579	-0,238
_chat	-0,453	0,239	-0,107	0,154	-0,208
_mange	0,308	0,037	-0,430	0,382	-0,300
_de	0,554	0,351	-0,347	-0,368	-0,218
_très	0,526	0,512	-0,209	-0,264	-0,379
_mauvaises	-0,166	0,649	-0,061	0,015	-0,332
_croquettes	0,110	0,190	-0,640	0,160	-0,194
.	-0,061	-0,205	0,118	0,964	0,043

Pas vraiment de surprise, le principe est exactement identique à l’étiquetage morphosyntaxique, l’extraction d’entités nommés ….

Un autre exemple avec un modèle différent et d’autres classes :

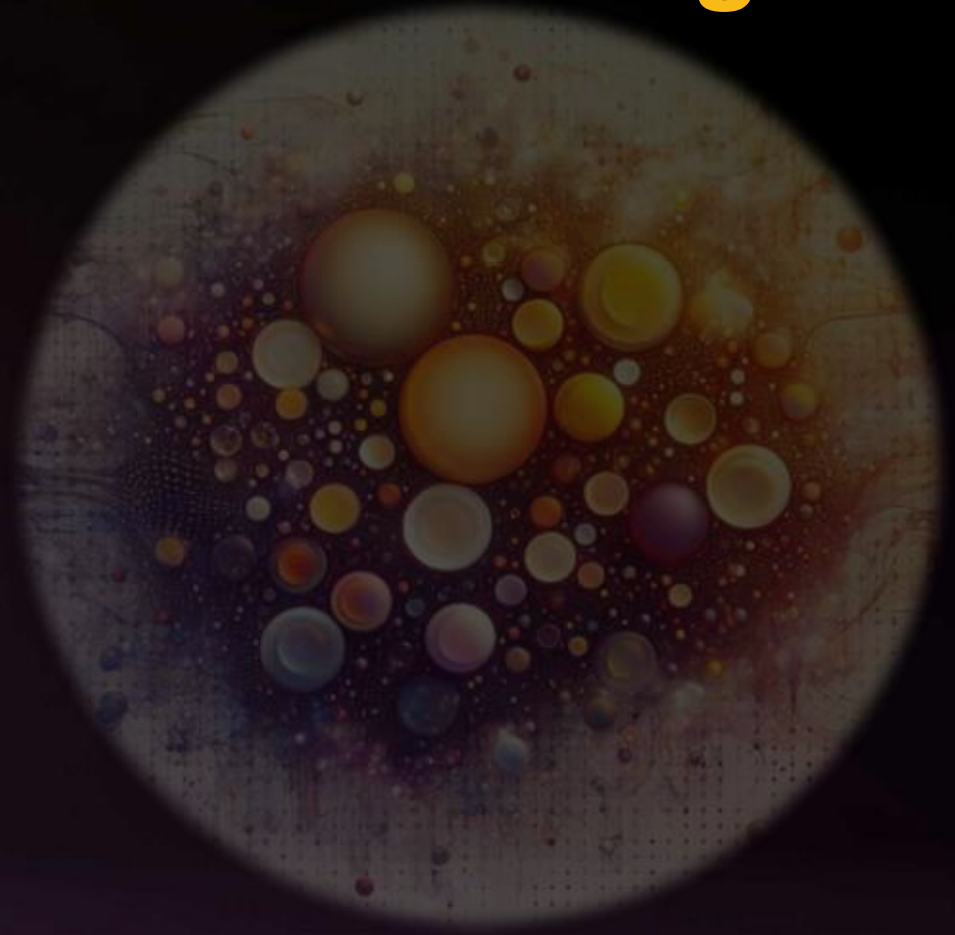
Token	MIXED	NEGATIVE	OBJECTIVE	POSITIVE
Le</w>	3,210	2,524	0,957	-1,152
chat</w>	2,749	1,138	4,408	3,591
mange</w>	3,530	1,454	2,931	3,338
de</w>	3,436	0,455	3,328	-0,426
très</w>	3,688	4,117	2,887	1,922
mauvaises</w>	-1,104	3,227	-0,051	1,261
croquettes</w>	2,310	1,201	-0,065	0,702
.</w>	-10,212	17,151	13,014	15,011

Analyse de sentiments avec les transformers (classification du texte)



Contrairement aux modèles précédents, le texte en entré est classifié dans sa globalité

Classification & Clustering



Utilisation des transformers pour la classification documentaire

Modèles Standards

- choix d'un modèle pré-entraîné :** BERT, GPT-2, etc., captent informations sémantiques et contextuelles complexes.
- Fine-tuning :** Adapte modèle pré-entraîné à la tâche spécifique.

Approches "Zeroshot classification

- Utiles quand catégories possibles sont connues à l'avance, même si non présentes dans les données d'entraînement.
- Modèles transformer avancés comme BART comprennent subtilités du langage, conduisant à des classifications précises même pour des catégories non vues auparavant.

Limites de Taille dues à l'architecture basée sur l'attention

- Troncature ou troncature dynamique.
- Sliding window.
- Stratégies hiérarchiques.
- Utilisation de modèles spécifiques comme Longformer ou BigBird.

Utilisation des transformers pour le regroupement de documents

Avantages des Transformers pour le clustering de documents

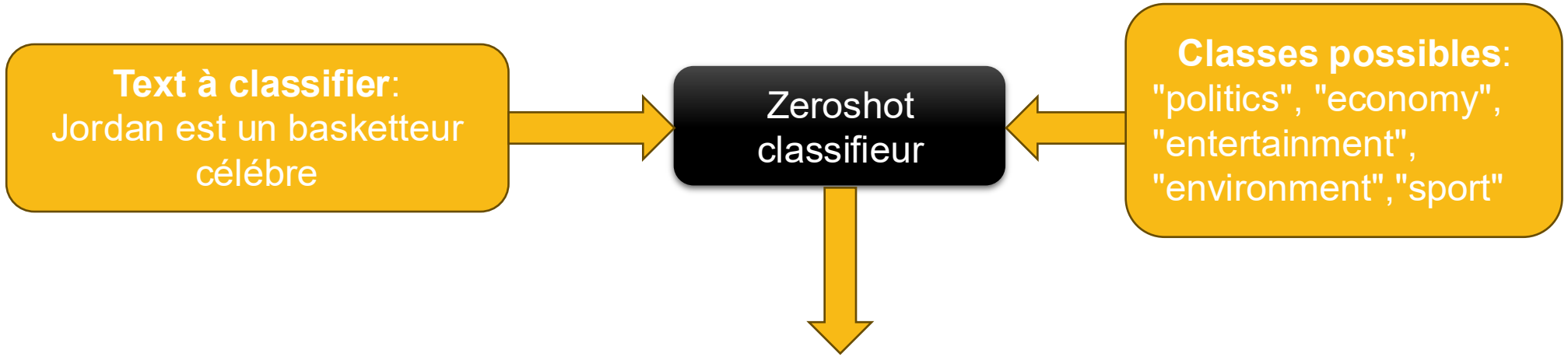
- Capture d'informations sémantiques complexes
- Regroupements plus précis et significatifs

Étapes:

- **Représentation en vecteurs** :conversion des documents en vecteurs denses via des modèles comme BERT, capturant sémantique et contexte.
- **Réduction de dimensionnalité** : utilisation de techniques comme PCA, t-SNE ou UMAP pour gérer la haute dimensionnalité.
- **Algorithme de clustering** : application de méthodes telles que K-Means, DBSCAN, ou Hierarchical Clustering.
- **Évaluation des clusters** : Utilisation de métriques comme l'indice de silhouette pour mesurer la qualité des regroupements.
- **Interprétation des clusters** : compréhension des caractéristiques communes des documents dans chaque cluster.

Classification de documents avec les transformers

Note: Le dernier modèle d'analyse sentiment peut être vu comme un modèle de classification



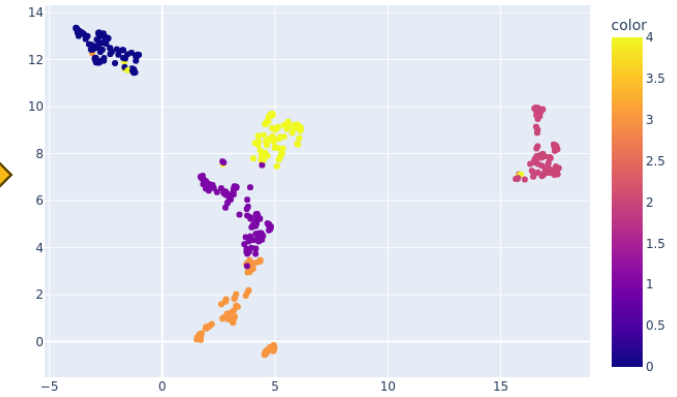
- 'labels': ['**sport**', 'entertainment', 'environment', 'economy', 'politics'],
- 'scores': [**0.8383722305297852**, 0.10435088723897934, 0.025080088526010513, 0.02190273255109787, 0.010294051840901375]

Clustering de documents avec les transformers

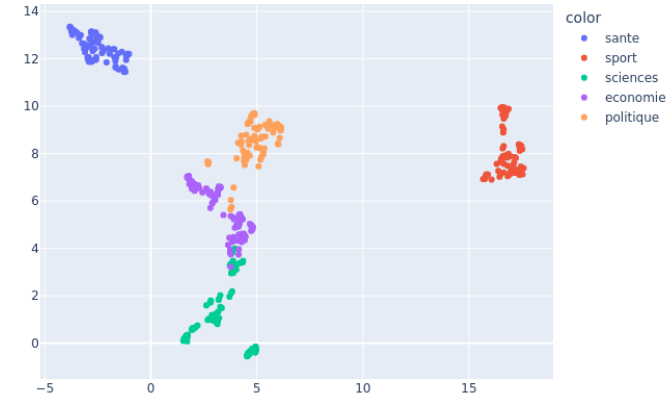
Modèle

Clustering

Clustering K-means sur des embeddings



Reference (mapping des classes connues)

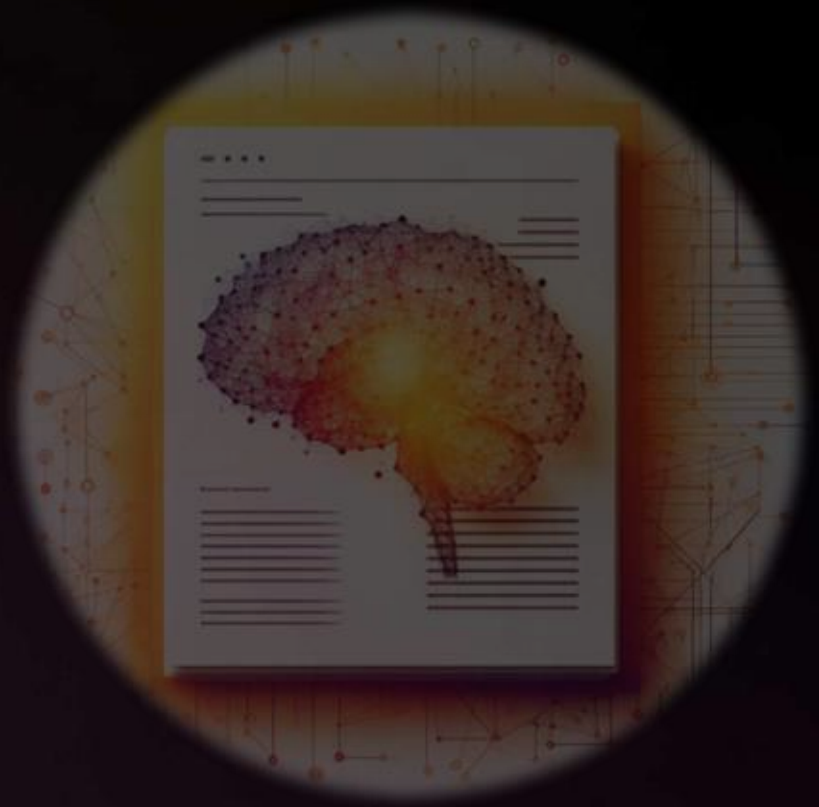


Les corpus documentaires sont souvent de tailles conséquentes avec des thématiques hétérogènes : il y a un intérêt à regrouper ces documents par thématiques pour un facilité la compréhension globale

Lorsque nous utilisons un *transformer* dans le cadre du clustering, c'est pour transformer des documents en vecteurs à des fins de comparaisons

Le clustering permet de regrouper des vecteurs proches au regard d'une distance

Résumé automatique

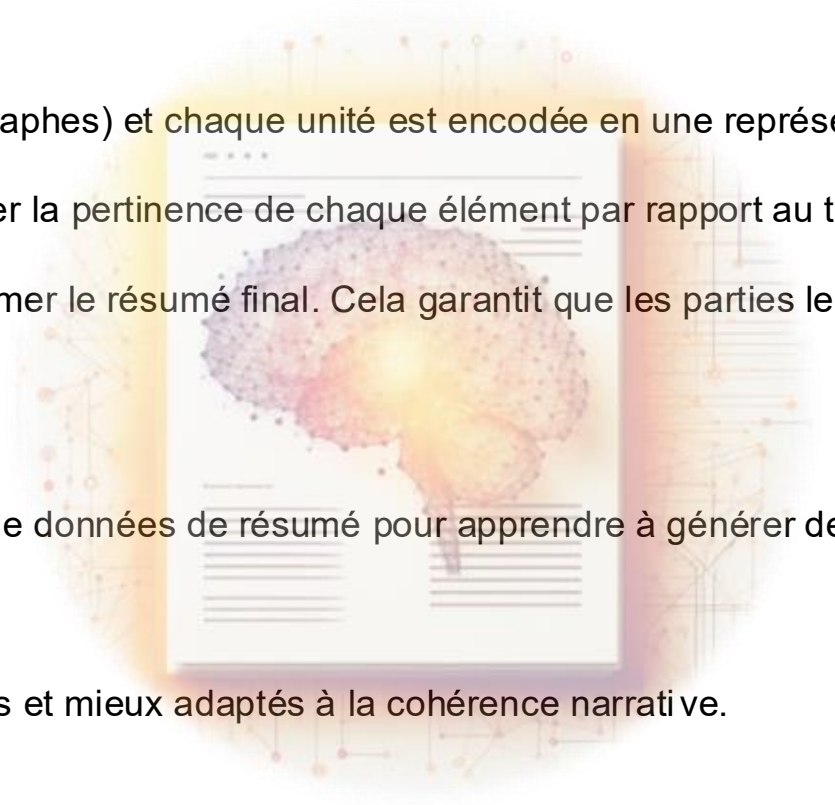


Utilisation des transformers pour le résumé automatique

Les techniques de résumé automatique avec les Transformers exploitent la puissance des réseaux neuronaux pour extraire l'essence d'un texte de manière précise et contextuellement riche. Deux principales approches sont utilisées : l'approche extractive et l'approche générative.

Approche Extractive avec les Transformers

- **Encodage du texte source** : le texte est divisé en unités (mots, phrases ou paragraphes) et chaque unité est encodée en une représentation vectorielle grâce au modèle Transformer.
- **Calcul de la pertinence** : le modèle utilise ses mécanismes d'attention pour évaluer la pertinence de chaque élément par rapport au texte global.
- **Sélection des éléments** : les éléments les mieux notés sont sélectionnés pour former le résumé final. Cela garantit que les parties les plus essentielles du texte sont conservées.



Approche Générative avec les Transformers

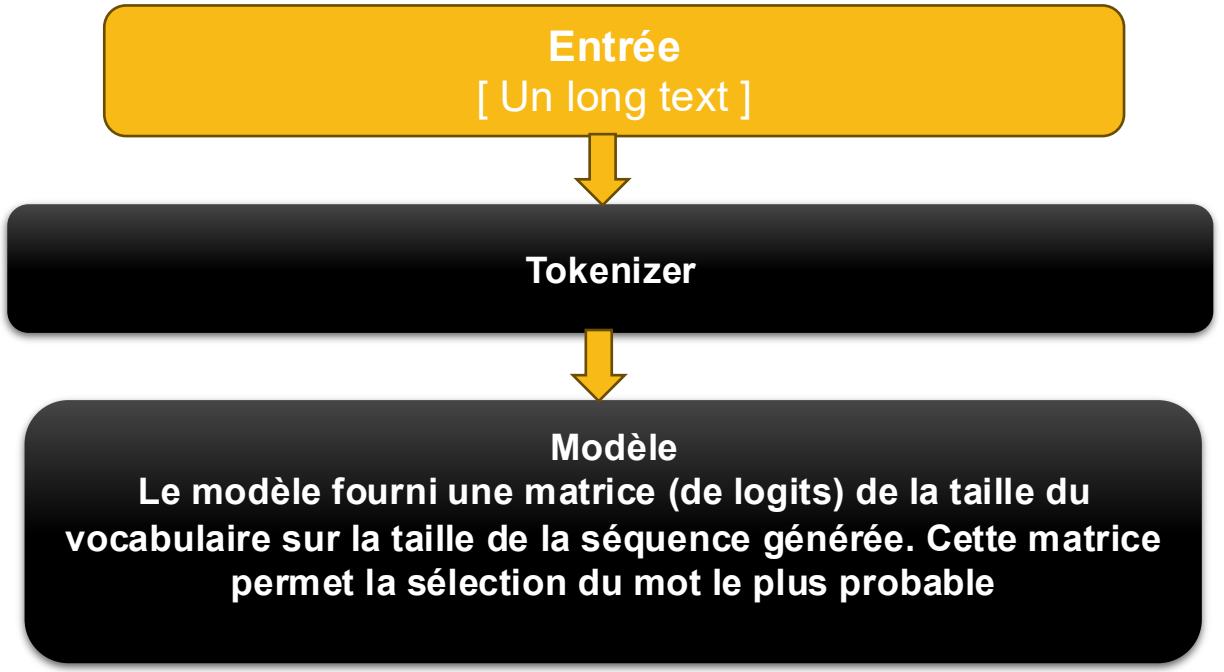
- **Entraînement sur des données de résumé** : le modèle est appris sur ensemble de données de résumé pour apprendre à générer des résumés précis.
- **Génération du Résumé**

L'approche générative avec les Transformers permet d'obtenir des résumés plus fluides et mieux adaptés à la cohérence narrative.

Combinaison des Approches

Il est également possible de combiner les approches extractives et génératives pour bénéficier des avantages de chacune. Dans cette configuration, les parties initiales du résumé sont extraites du texte source, puis le modèle génératif complète et affine le résumé.

Résumé automatique avec les transformers



La taille du texte a une importance, car l'entrée des modèles ont une taille déterminée

Les modèles de résumé, sont généralement des modèle texte vers texte avec un encodeur et un décodeur (ce qui n'est pas le cas de la plupart des modèles de classification où seul un encodeur à laquelle est ajouté une couche de classification est utilisée)

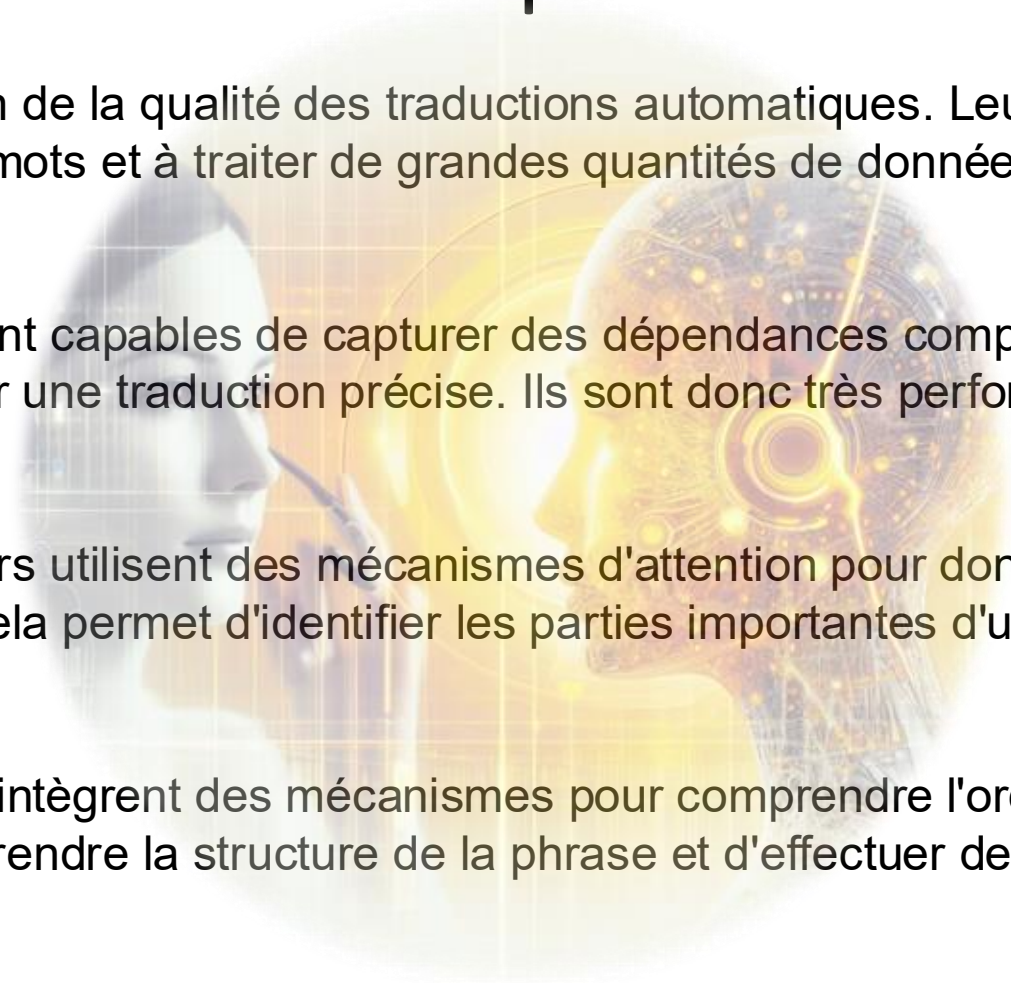
La génération du texte faite mot par mot avec des stratégies différentes:

- **Greedy** : sélection de la probabilité maximum
- **Sampling** : sélection se fait selon on loi de probabilité (multinomial par exemple)
- **Beam search** : un semble de n-probabilité est conservé et on cherche à maximiser une séquence

Traduction automatique

Utilisation des transformers pour la traduction automatique

Les transformers ont joué un rôle majeur dans l'amélioration de la qualité des traductions automatiques. Leur capacité à capturer des dépendances complexes entre les mots et à traiter de grandes quantités de données en font un choix idéal pour cette tâche linguistique spécifique.



- **Capture de dépendances complexes** : ces modèles sont capables de capturer des dépendances complexes entre les mots dans une phrase, ce qui est essentiel pour une traduction précise. Ils sont donc très performants pour cette tâche.

- **Utilisation de mécanismes d'attention** : les transformers utilisent des mécanismes d'attention pour donner plus de poids à certains mots en fonction du contexte. Cela permet d'identifier les parties importantes d'une phrase lors de la prise de décision de traduction.

- **Importance de la position des mots** : les transformers intègrent des mécanismes pour comprendre l'ordre des mots dans une phrase. Cela permet au modèle de comprendre la structure de la phrase et d'effectuer des traductions précises.

Traduction avec les transformers

Entrée
translate English to French: The cat eats the mouse

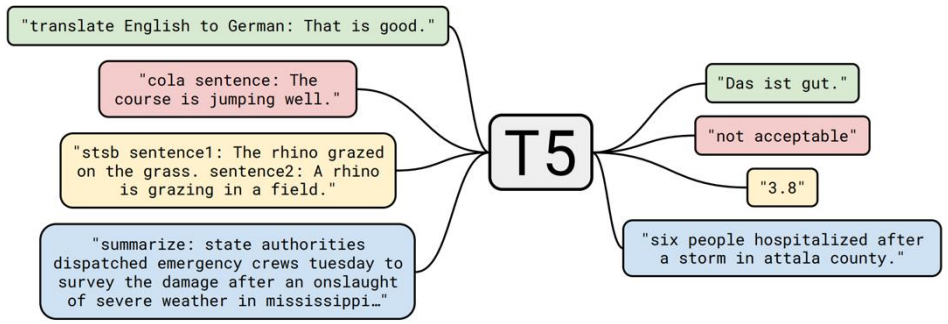
Tokenizer

Modèle
Le modèle fourni une matrice (de logits) de la taille du vocabulaire sur la taille de la séquence générée. Cette matrice permet la sélection du mot le plus probable

Sortie
Le chat mange la souris

Le prompt permet de sélectionner la tâche ici nous nous intéressons à la traduction (‘**translate English to French:**’)

Les modèles de traduction, sont généralement des modèles texte vers texte avec un encodeur et un décodeur. Au même titre que le résumé, la génération est faite selon les stratégies greedy, sampling, beam search



Les modèles multi tâches sont particulièrement intéressants : ils profitent de l’apprentissage de tâches complémentaires pour améliorer les performances

Les agents LLM

01 Architecture d'un agent

02 Les ReAct et planification

03 La Function Calling

04 Frameworks

Les agents LLM : architecture et fonctionnement

- Definition : un agent LLM est un systeme ou le modele de langage joue le role d'un 'cerveau' : il percoit un contexte, raisonne sur l'objectif, planifie les etapes et execute des actions via des outils externes.
- Composants cles : LLM (raisonnement et planification), Outils/Tools (recherche web, execution de code, APIs, bases de donnees), Memoire court-terme (historique de conversation), Memoire long-terme (base vectorielle).
- ReAct (Reason + Act) : pattern fondamental des agents. Le modele alterne Thought (raisonnement interne), Action (appel d'outil) et Observation (resultat de l'outil) dans une boucle jusqu'a obtenir la reponse finale.
- Multi-agents : plusieurs agents specialises collaborent - un agent Researcher cherche l'information, un agent Analyst la traite, un agent Writer redige. AutoGen (Microsoft) et CrewAI popularisent ce paradigme.
- Limites actuelles : boucles infinies, erreurs cumulees (compounding errors), hallucinations dans les decisions d'action, cout eleve par inference, difficile a debugger et a monitorer en production.

Les frameworks pour construire des agents

L'ecosysteme des frameworks agents a explose en 2023-2025. Chaque outil repond a un besoin specifique :

- **LangChain** : framework le plus populaire pour chainer LLMs, outils et memoires. Agents ReAct, MRKL, OpenAI Functions. Ecosysteme tres riche mais parfois complexe a debugger.
- **LlamaIndex** : spécialisé dans la construction de pipelines RAG et d'agents sur des corpus documentaires. Excellent pour l'indexation et la recherche dans de grands volumes de documents.
- **AutoGen (Microsoft)** : multi-agents conversationnels. Des agents specialises (Researcher, Coder, Reviewer) collaborent via des messages pour resoudre des taches complexes.
- **CrewAI** : orchestration d'equipes d'agents avec roles et objectifs definis. Interface plus simple qu'AutoGen pour les workflows multi-agents en production.
- **Smolagents (HuggingFace)** : agents legers bases sur des modeles open-source. Les agents ecrivent et executent du code Python comme outil de base. Tres efficace pour les taches analytiques.
- **Haystack** : framework NLP pour des pipelines de recherche et generation augmentee en production. Fort sur les architectures RAG hybrides et l'evaluation.

Les modeles multimodaux

01 Vision-Language Models

02 Les Architectures VLM

03 La Applications

04 Tendances 2025

Les transformers multimodaux : au-delà du texte

- Definition : les modeles multimodaux traitent plusieurs types de donnees simultanement - texte, image, audio, video - dans un espace de representation partage, permettant le raisonnement entre modalites.
- CLIP (OpenAI, 2021) : pre-entraine sur 400 millions de paires image-texte. Projette images et textes dans le meme espace vectoriel pour la recherche d'images par texte ou la classification zero-shot.
- GPT-4V / GPT-4o : analyse d'images, de graphiques, de schemas et de documents scannes en langage naturel. Capable de raisonner sur des contenus visuels complexes et de repondre a des questions sur des captures d'ecran.
- LLaVA (Large Language and Vision Assistant) : modele open-source Vision-Language. Combine un encodeur visuel (CLIP ViT) avec un LLM (Llama) via un connecteur projector leger. Entrainable sur une seule machine.
- Gemini 2.0 / GPT-4o : nativement multimodaux (texte + image + audio + video). Traitent des heures de contenu video dans leur fenetre de contexte et peuvent generer du contenu dans toutes les modalites.

Applications des modèles multimodaux

La multimodalite ouvre des cas d'usage impossibles avec le texte seul :

- **Extraction de donnees visuelles** : OCR avance sur factures, contrats et tableaux complexes. Analyse de graphiques et schemas techniques. Traitement de formulaires scannes sans template fixe.
- **Recherche image-texte** : retrouver une image par description textuelle ('une photo de plage au coucher de soleil avec des palmiers') ou inversement trouver des documents via une image requete.
- **Agents visuels (computer use)** : agents capables de 'voir' un ecran, de lire une interface graphique et d'interagir avec des applications comme un humain. Claude Computer Use, GPT-4V pour l'automatisation UI.
- **Accessibilite** : description automatique d'images pour malvoyants, transcription et resume de videos, generation de sous-titres contextuellement precis. Impact social majeur.
- **Medical et scientifique** : analyse d'imagerie medicale (radio, IRM) assistee par LLM, interpretation de graphiques scientifiques, lecture de publications avec figures.
- **Generation multimodale** : creer automatiquement des presentations illustrees, des rapports visuels ou des interfaces a partir de descriptions textuelles via des modeles text-to-image+texte.

Réglementation & IA: le cadre européen

Pourquoi réglementer l'IA ?

Les modèles de langage (LLM) et les systèmes d'IA générative sont désormais déployés dans des contextes critiques : santé, justice, finance, ressources humaines, infrastructures numériques. Cette diffusion massive soulève des questions fondamentales auxquelles le droit doit répondre.

Responsabilité

Qui est responsable en cas de décision erronée prise par un système automatisé ?

Protection des données

Comment protéger les données personnelles traitées par les LLM à grande échelle ?

Droits des personnes

Quels recours pour les individus affectés par une décision entièrement automatisée ?

L'Union Européenne a répondu en construisant un **cadre réglementaire cohérent et multicouche**, dont l'AI Act constitue la pièce maîtresse — complété par plusieurs textes sectoriels complémentaires.

La cartographie réglementaire européenne

Ces textes ne s'excluent pas mutuellement : **ils se superposent**. Un système d'IA déployé dans une banque peut relever simultanément du RGPD, de DORA, de NIS2 et de l'AI Act. Comprendre leur articulation est essentiel pour toute démarche de conformité.

Texte complet	Acronyme	Objet principal	En vigueur
General Data Protection Regulation	**RGPD**	Protection des données personnelles	2018
Network and Information Security 2	**NIS2**	Cybersécurité des entités critiques	2024
Digital Operational Resilience Act	**DORA**	Résilience numérique du secteur financier	2025
Cyber Resilience Act	**CRA**	Sécurité des produits connectés	2025
EU AI Act	**AI Act**	Régulation des systèmes d'IA	2024–2027

Le RGPD & les LLM : les points de friction

Le RGPD (2016/679) s'applique dès lors qu'un traitement concerne des **données à caractère personnel** — et les LLM en consomment massivement, à chaque étape de leur cycle de vie.

Entraînement

Les corpus d'entraînement peuvent contenir des données personnelles collectées sans base légale adéquate.

Inférence

Les prompts soumis par les utilisateurs constituent fréquemment des données personnelles à part entière.

Mémorisation

Les LLM peuvent restituer des données sensibles mémorisées lors de l'entraînement (*memorization attacks*).

Principes clés à respecter

→ Minimisation

Ne collecter que les données strictement nécessaires à la finalité déclarée.

→ Finalité

Ne pas réutiliser les données à des fins non prévues lors de la collect

→ Droit à l'effacement

Comment "oublier" une donnée dans un réseau de neurones ? Le *machine unlearning* reste un problème ouvert.

⚠ Le **Comité Européen à la Protection des Données (CEPD)** a publié en 2024 des lignes directrices spécifiques aux LLM, à consulter impérativement.

La DPIA : évaluer l'impact avant de déployer

La **DPIA** (Data Protection Impact Assessment, ou AIPD en français) est une **analyse d'impact obligatoire** imposée par l'article 35 du RGPD, à réaliser avant tout traitement susceptible de présenter un risque élevé pour les droits et libertés des personnes.

Quand est-elle obligatoire ?

Décision automatisée

Scoring, profilage, notation comportementale

Données sensibles à grande échelle

Santé, origine ethnique, opinions politiques…

Surveillance systématique

Espaces accessibles au public

Contenu obligatoire de la DPIA

01

Description du traitement

Finalités, nature des données, catégories de personnes concernées

02

Nécessité et proportionnalité

Justification du traitement au regard de l'objectif poursuivi

03

Identification des risques

Probabilité, gravité et sources des risques identifiés

04

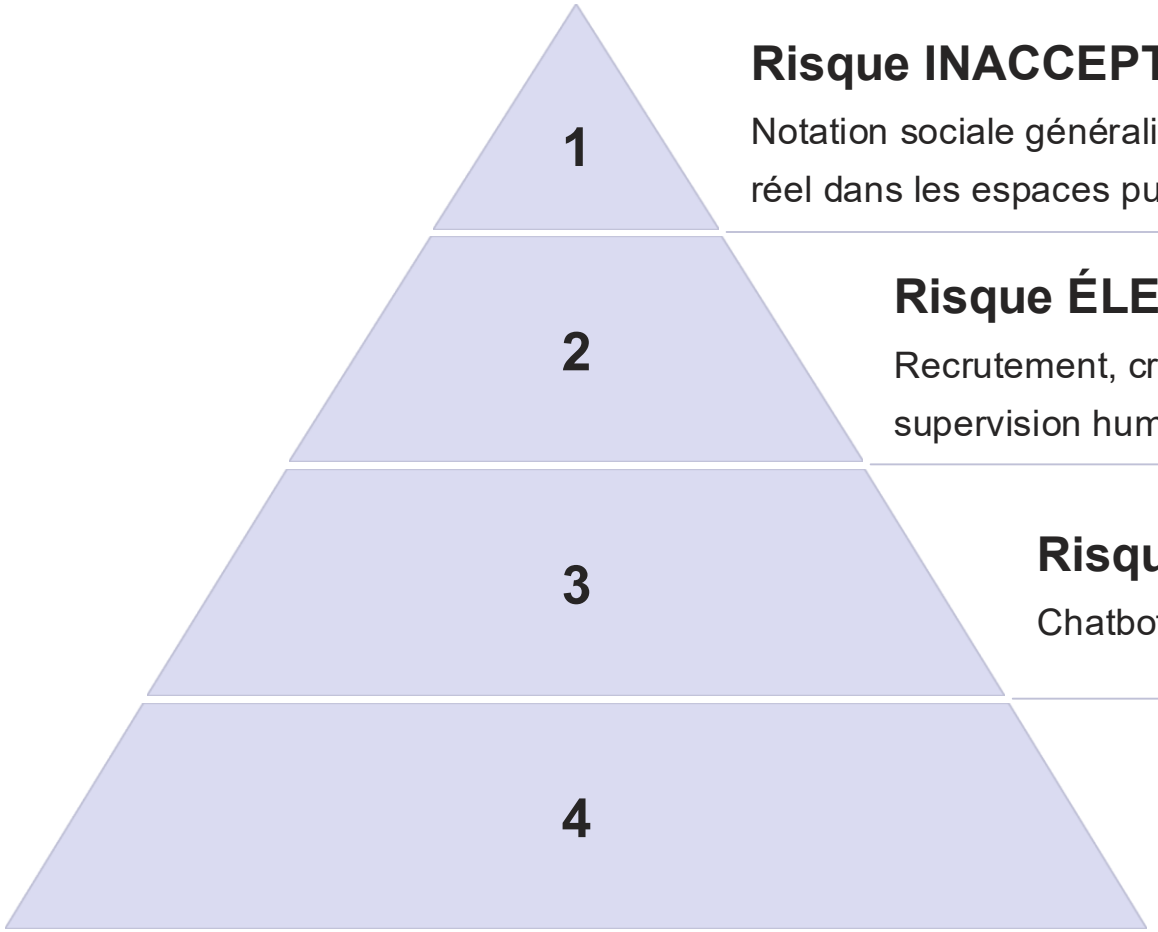
Mesures d'atténuation

Techniques et organisationnelles pour réduire les risques résiduels

En pratique : Un chatbot RH évaluant des candidats → DPIA obligatoire. Un moteur de recherche interne sur documents publics → probablement non.

L'AI Act : une approche par les risques

Le **Règlement européen sur l'IA (2024/1689)** classe les systèmes d'IA en quatre niveaux de risque. Plus le risque est élevé, plus les obligations sont contraignantes. Les premières interdictions sont effectives depuis **février 2025** ; les obligations pour les systèmes à risque élevé s'appliquent pleinement en **2027**.



Risque INACCEPTABLE → Interdit

Notation sociale généralisée, manipulation subliminale, reconnaissance faciale en temps réel dans les espaces publics

Risque ÉLEVÉ → Obligations strictes

Recrutement, crédit, justice, éducation, infrastructures → tests de robustesse, logs, supervision humaine, marquage CE

Risque LIMITÉ → Transparence obligatoire

Chatbots → l'utilisateur doit être informé qu'il interagit avec une IA

Risque MINIMAL → Pas de contrainte spécifique

Filtres anti-spam, jeux vidéo, recommandations de contenu

L'AI Act & les modèles à usage général (GPAI)

Les **GPAI** (*General Purpose AI models*) — dont GPT-4, Claude, Gemini, Llama — font l'objet d'un régime spécifique au sein de l'AI Act, reconnaissant leur nature transversale et leur potentiel d'impact systémique.

Obligations pour tous les GPAI

- Documentation technique complète et mise à jour régulière
- Politique de respect du droit d'auteur
- Publication d'un résumé des données d'entraînement

GPAI "systémiques" (> 10^{25} FLOPs)

- Évaluation des risques systémiques avant mise sur le marché
- Tests adversariaux obligatoires (*red teaming*)
- Signalement des incidents graves à la Commission européenne
- Mesures de cybersécurité renforcées

L'**AI Office**, rattaché à la Commission européenne, est l'autorité de supervision des GPAI au niveau européen.

Points de vigilance pour un praticien NLP

Au-delà de la connaissance des textes, c'est dans les pratiques quotidiennes de développement que la conformité se construit. Voici les six risques principaux à maîtriser impérativement.

Risque identifié	Texte applicable	Bonne pratique recommandée
Données personnelles dans les prompts	RGPD	Anonymisation et pseudonymisation systématiques en amont du traitement
Décision automatisée sans supervision	AI Act (risque élevé)	*Human-in-the-loop* obligatoire pour toute décision impactante
Mémorisation de données d'entraînement	RGPD + AI Act	*Differential privacy*, techniques de *machine unlearning*
Fournisseur LLM externe non audité	NIS2 / DORA	Due diligence contractuelle, SLA de sécurité, audit tiers
Biais discriminatoires dans les sorties	AI Act	Évaluation d'équité régulière, jeux de test diversifiés et représentatifs
Deepfakes et contenus synthétiques	AI Act	Marquage obligatoire des contenus générés par IA (watermarking)

Pour aller plus loin : Texte intégral de l'AI Act · Guidelines CEPD sur les LLM (2024) · AI Office européen

MERCI

Credits

Openshot
Bentsound
Allppt.com
Pixabay.com
Huggingface.com

Remerciements

Aux personnes grandes et petites qui me supportent au quotidien

Droit de reproduction