inalco
Institut national
des langues
et civilisations orientales

aday

Documents structurés

Master 2 TAL

Rime ABROUGUI

Data Scientist,
Chercheuse en TAL

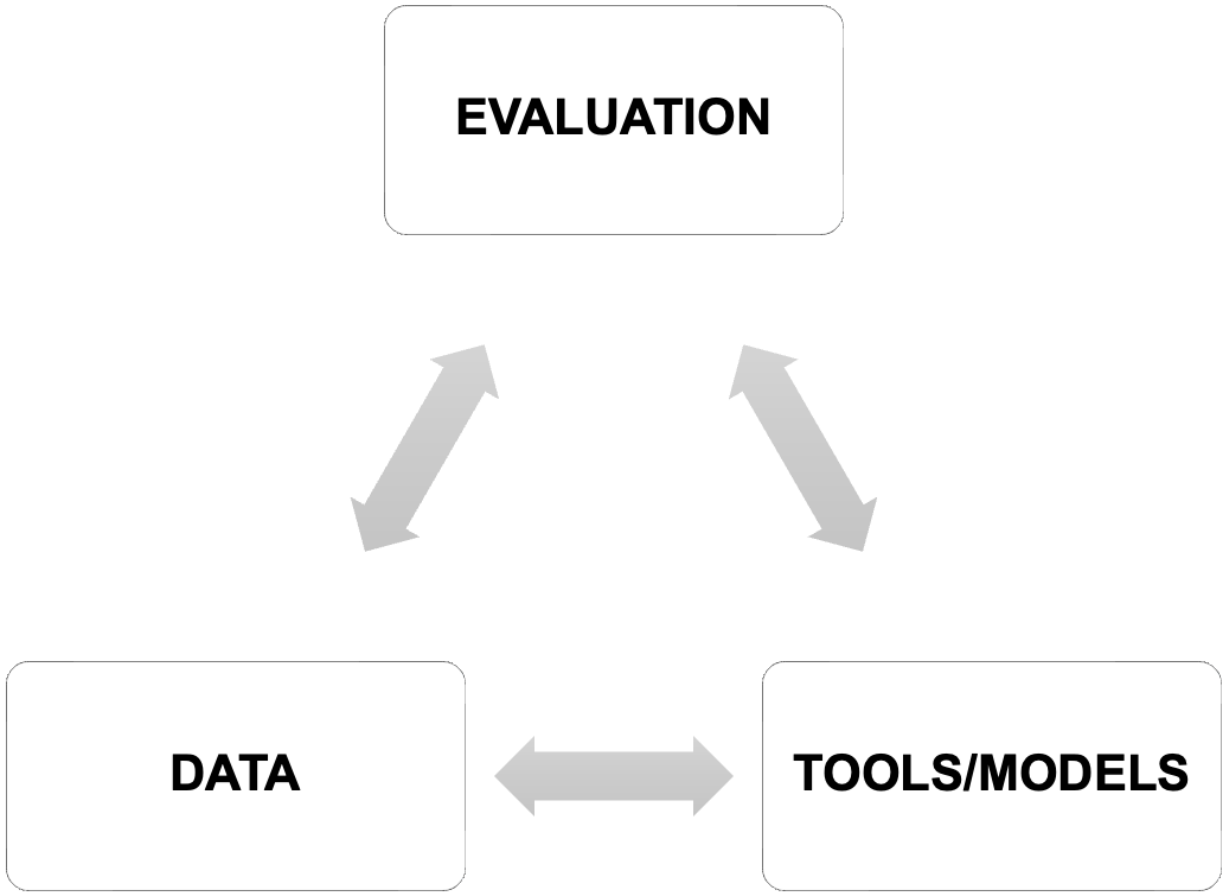
Semestre 1

Objectifs

- Développer une vision critique des outils de TAL existants (Spacy, Hugging-Face, etc.)
- Analyser des données et les structurer
- Acquérir des compétences en évaluation et réentraînement des modèles sur des données
- S'initier à la réplicabilité des résultats
- Se familiariser avec la lecture d'articles et la rédaction scientifiques

Introduction

- Avant de commencer un projet NLP :
  - Nécessité d'un **ensemble de données** adapté
  - Choix d'un outil



Introduction

- **Choix de l'outil et du modèle** :
  - Choix basé sur les besoins du projet :
    - ⋆ Type de tâche (NER, classification, traduction, ...)
    - ⋆ Volume et complexité des données (corpus de grande taille ou non)
    - ⋆ Langues supportées
  - Impact sur la qualité des résultats :
    - ⋆ Performance en termes d'accuracy, précision, rappel, F1-score...
    - ⋆ Vitesse d'exécution (temps d'inférence, optimisation des pipelines)
    - ⋆ Facilité d'intégration dans des systèmes réels ou production
  - Possibilité du ré-entraînement pour adapter un modèle aux données spécifiques (langue, domaine)

Outils en TAL

Outils: Natural Language Toolkit (NLTK)

- Open-source pour les tâches de base en TAL
- Fonctionnalités :
  - Tokenisation, POS tagging, parsing, stemming...
  - Accès à plusieurs corpus linguistiques (Brown, Gutenberg, WordNet, etc.)
  - Applications académiques ou de recherche de petite échelle
- Limitations :
  - Moins performant sur les tâches complexes comme l'interprétation sémantique
  - Industrialisation limitée

Outils en TAL: SpaCy

- Open-source pour les applications TAL
- Fonctionnalités :
  - Modèles pré-entraînés pour NER, POS tagging, dépendances syntaxiques...
  - Supporte le réentraînement sur des tâches spécifiques
  - API simple à utiliser
- Limitations :
  - Modèles fournis pas toujours efficaces
  - Difficulté pour personnaliser et ajouter de nouvelles tâches

Outils en TAL: Transformers via Hugging Face

- **Hugging Face:** plateforme et bibliothèque open-source facilitant l'accès aux Transformers
- Modèles pré-entraînés sur de larges corpus (BERT, GPT, T5, etc.)
- Fonctionnalités :
  - Fine-tuning pour des tâches de classification (NER, POS, analyse de sentiments...) et des tâches de génération (traduction, résumés...)
  - Flexibilité pour la recherche et la production
- Limitations :
  - Complexité pour les débutants
  - Couteux en ressources (mémoire, temps de calcul)

Les données

L'importance des données dans le TAL

- **Essentielles pour l'entraînement et l'évaluation des modèles** :
  - Modèles dépendants des données **train**
  - Évaluation avec les données **test**
    - ⋆ Qu'est-ce qu'on veut évaluer ?
    - ⋆ Données test doivent couvrir l'ensemble des caractéristiques
    - ⋆ Importance d'avoir une bonne qualité d'annotation (cf gold & silver annotation)
- **Impact sur les résultats** :
  - Qualité des données → performance des modèles
  - Diversité des données → capacité de généralisation

Éthique des données

- Utilisation responsable des données :
  - Respect de la vie privée et des droits individuels (cf. loi RGPD)
- Problème de filtrage
  - Filtrage des contenus nuisibles (langage inapproprié, biais toxiques, etc.)
- Importance de la transparence dans la collecte et le traitement des corpus

Annotations : Choix et impact

- Choix des labels : déterminer quelles informations annoter
- Constitution d'une ontologie pour définir les catégories et relations
- Méthodes d'annotation :
  - Automatique ou manuelle
  - Expert vs crowd-sourcing
- Impact sur les performances : annotations imprécises peuvent fausser l'évaluation

Texte original	**Annotation correcte**	**Annotation erronée**
"Je suis allée à **Paris**"	`Ville`	O
"**Paris** est magnifique"	`Ville`	`Organisation`
"Les lumières de **Paris**"	`Ville`	O

Qualité et étude du corpus avant l'évaluation

- Qualité d'input
  - Tokenisation
  - Ponctuation...
  - Dépend aussi de la tâche
- Analyser et quantifier les cas problématiques
  - Longueur des séquences
  - Manque de représentativité des variations linguistiques correspondant à la tâche
  - Annotations ambiguës
  - OOV et OOD
  - Distribution inégales des labels
- Identifier et corriger les biais et les erreurs
  - Une bonne analyse d'erreur
  - Amélioration du modèle

Exemple de problèmes des données : OOV et OOD

- **OOV (Out of Vocabulary)** :
  - Les mots non vus pendant l'entraînement ne sont pas reconnus par le modèle
  - Problème important pour les corpus spécialisés par exemple
- **OOD (Out Of Domain)** :
  - Définition difficile à cerner, car elle dépend de la tâche
  - Exemple: Modèles formés sur un domaine particulier (**Presse**) performants uniquement dans ce domaine
  - Importance de tester les modèles sur des données hors domaine pour évaluer leur robustesse

Exemple de problèmes des données : distribution inégale

- Distribution déséquilibrée des labels :
  - Certaines classes sont sur-représentées, d'autres sous-représentées
- Impact direct sur les performances:
  - Biais en faveur des classes majoritaires
  - Exemple : modèle de classification de sentiments catégorisant tous les exemples en "*positif*" :
    - ⋆ Raison possible : la classe "*positif*" domine dans le corpus
- Vérification des statistiques des données (nombre de tokens, nombre des énoncés, fréquence des labels, etc.)

Pré-processing des données

- Division du corpus :
  - Création de jeux d'entraînement, validation et test
- Gestion des caractères spéciaux et des éléments non pertinents
- Tokenization :
  - Découpage du texte en unités significatives (mots, sous-mots)
- Vectorisation / Encodage :
  - Conversion des mots ou du texte en une représentation numérique (vecteurs de nombres) pour être utilisable par les modèles et les algorithmes

Annotations et Structures dans le TAL

Annotations et Structures dans le TAL

- Représentation des informations linguistiques de manière compréhensible pour les machines
- Transformation du texte en une forme exploitable par les algorithmes
- Progrès dans le domaine avec les réseaux de neurones ouvrant la voie à des structures plus complexes

Les Annotations

- **Définition des annotations** :
  - Des marquages ajoutés aux données textuelles pour ajouter des informations linguistiques (par ex. parties du discours, relations syntaxiques, entités nommées)

- **Rôle dans le TAL** :
  - Faciliter le traitement automatique du langage en précisant les entrées des modèles (*inputs*) et leurs sorties (*outputs*)
  - Fournir des indications sur la structure et le sens du texte
  - Interprétation des données textuelles de manière plus précise

Annotations Structurées

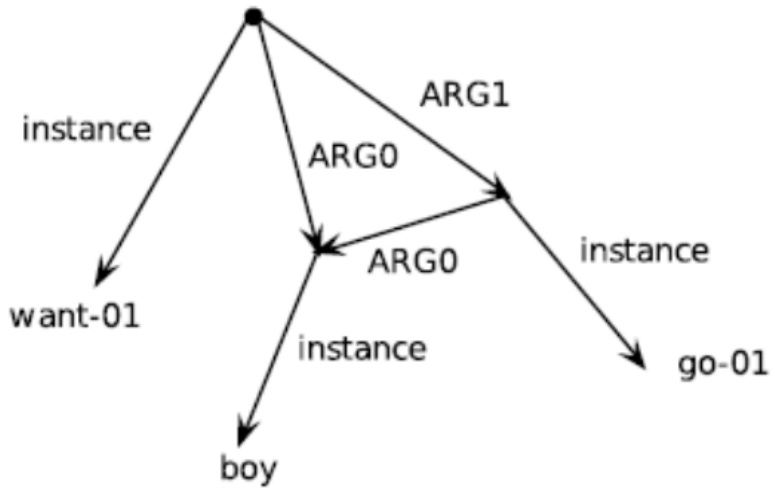
- **Définition** : représentation de la hiérarchie et les relations complexes entre les éléments d'un texte (par ex. dépendances syntaxiques, relations sémantiques entre les entités)
- **Évolution** :
  - Initialement sous-utilisées en raison de l'absence de modèles capables de traiter ces structures
  - Plus pertinentes avec l'arrivée de modèles capables de traiter des représentations hiérarchiques
- **Exemples d'utilisations** : Analyse syntaxique des dépendances, extraction de relations entre entités, etc.

Annotations Structurées: exemple AMR (Abstract Meaning Representation)

AMR format (based on PENMAN):

```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01
           :arg0 b))
```

GRAPH format:



Annotations à Plat

- **Définition** : représentation linéaire des annotations où chaque élément est traité de manière indépendante, sans structure hiérarchique
- **Histoire** :
  - L'une des premières formes d'annotation en TAL
  - Prise en charge par des formats tabulés comme CoNLL, qui listent les mots et leurs annotations ligne par ligne
- **Rôle** : cncore très utilisé pour des tâches comme l'étiquetage des parties du discours ou la reconnaissance d'entités nommées

Annotations à Plat: exemple du coprus Sequoia

Lemme	**Étiquette**
nous	PRON
avoir	AUX
noter	VERB
que	SCONJ
le	DET
production	NOUN
de	ADP
électricité	NOUN
correspondre	VERB
à	ADP
30	NUM
%	NOUN
.	PUNCT

Annotations Semi-Structurées

- **Définition** : Représentations qui tentent d'intégrer une structure hiérarchique tout en restant dans un format à plat, facilitant leur traitement
- **Évolution** :
  - Utilisation des formats où les données sont partiellement organisées en objets imbriqués
  - Réponse à la nécessité de capturer une certaine hiérarchie sans complexité excessive
- **Utilisation** : Cas d'analyse sémantique et syntaxique de phrases complexes, reconnaissance d'entité nommées, etc.

Annotations Semi-Structurées: exemple du corpus MEDIA

Words	Mode	Attribute Name
donnez-moi	+	null
le	?	reflink-coref
tarif	?	object
puisque	+	connectProp
je voudrais	+	null
une chambre	+	number-room
qui coûte	+	object
pas plus de	+	comparative-payment
cinquante	+	payment-amount-integer-room
euros	+	payment-unit

Outils en Python

- ‘lxml.etree‘ : manipuler et analyser les fichiers XML
- ‘BeautifulSoup‘ : utile pour extraire et manipuler des données textuelles à partir de fichiers HTML ou XML
- ‘nltk‘ : des fonctionnalités pour le traitement des arbres syntaxiques et l’analyse de texte
- ‘pandas‘ : utile pour lire et manipuler les fichiers CSV ou CoNLL
- ‘json‘ : fournit des outils pour manipuler et extraire des données à partir de fichiers JSON
- etc.

Evaluation

Évaluation

- Mesurer la performance d'un modèle ou d'un outil sur une tâche spécifique
- Comparaison entre modèles, algorithmes, méthodes, etc.
- Déterminer à quel point un modèle est capable de produire des résultats corrects en comparant ses sorties à des données de référence (gold standard)
- Différentes mesures (F-score, Accuracy...)
- Identification des erreurs et des zones d'amélioration

Méthodes d'évaluation

- **Mesures automatiques** :
  - Calculées par le système et basées sur des formules mathématiques
- **Mesures humaines** :
  - Jugements par des experts ou utilisateurs

- Importance de combiner les deux types de mesures pour une évaluation complète
  - **Évaluation qualitative et quantitative**

Exemples de métriques par tâche

- **Classification Binaire (e.g. Analyse de sentiment)** :
  - AUC-ROC
  - Accuracy
- **Multi-classification (e.g. étiquetage de séquence)**:
  - Précision, rappel, F1-score (Micro, Macro, Weighted Avg)
  - Accuracy
- **Tâche de Génération (e.g. traduction)** :
- ROUGE
- BLEU
- BERTScore...

Exemple d'évaluation humaine

- Pour des tâches où la qualité de la sortie est subjective ou difficile à quantifier automatiquement
- Evaluation générale des aspects comme la fluidité, la cohérence, la pertinence et l'exactitude des réponses
- Exemple :
  - **Notation sur une échelle** (par exemple, de 1 à 5) pour juger la qualité de la sortie
  - **Comparaison par paires**: comparaison des sorties de différents modèles
  - **Tâches d'annotation**: identification des erreurs et leurs types

Le corpus Sequoia (Candito,M. & Seddah, D. (2012))

- Projet de recherche financé par l'ANR
- Quatre sous-corpus annotés syntaxiquement
- Objectifs:
  - Fournir un jeu de données pour l'analyse syntaxique afin de tester et d'améliorer la robustesse des modèles
  - Développer et évaluer des méthodes d'adaptation à d'autres domaines que celui du corpus d'entrainement
  - Utilisation libre pour les recherches en TAL et les études linguistiques

Le corpus Sequoia: annotation

- Inspiration du schéma d'annotation syntaxique proche de celui du FTB (French Treebank)
- Annotation manuelle
- Vérification et validation des méthodes semi-automatiques
- Deux formats:
  - Un format parenthésé annoté en constituants
    - `((SENT (NP (NPP Gutenberg))))`
  - Un format tabulé CoNLL (c.f. TP)

Le corpus Sequoia: statistiques

- **Origines des Données**
  - Europarl (les débats parlementaires de l'Union européenne)
  - Le journal régional l'Est Républicain
  - Wikipedia en français
  - Documents de l'Agence Européenne du Médicament
- **3204 phrases**
  - 69 246 tokens
- les plus longues $\geq 29 tokens$
- **16 labels**
  - Distributions inégales

Le corpus Sequoia: Evaluation des prédictions Spacy

- **Matrice de confusion**:
  - Tableau récapitulant les vraies et fausses prédictions
- **Accuracy**:
  - Proportion de prédictions correctes sur l'ensemble des instances évaluées
- **F-mesure:**
  - Moyenne harmonique entre la précision et le rappel
  - Mesure utilisée pour les ensembles de données déséquilibrés

Exemple POS Tagging

- Phrase : "la consommation énergétique"
  - Référence: `DET NOUN ADJ`
  - Prédiction : `DET NOUN NOUN`
  - Erreur sur ADJ
- Phrase : "Je me pose également des questions"
  - Référence: `PRON PRON VERB ADV DET NOUN`
  - Prédiction: `PRON PRON VERB ADV DET NOUN`
  - Tous les labels sont corrects

Accuracy

- **Définition :** mesure de la proportion de prédictions correctes parmi toutes les prédictions
- **Formule :**

$$\text{Accuracy} = \frac{VP + VN}{VP + FP + FN + VN}$$

- **Interprétation :**
  - Valeurs allant de 0 à 1
  - 1 représente une classification parfaite
- **Utilisation :** mesure simple, mais peut être trompeuse pour les classes déséquilibrées

Calcul de l'Accuracy : Exemple POS Tagging

- **Formule :** $\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total d'échantillons}}$

- **Nombre total des labels** : 9
- **Total prédictions correctes** : 8 sur 9
- **Accuracy** : $\frac{8}{9} \approx 0.888$ (soit 88.8%)

- Si l'échantillon est la phrase, il faut voir si toutes les prédictions d'une seule phrase sont correctes ($\frac{1}{2} = 50\%$)

La F-mesure (F1-score)

- **Définition :** Moyenne harmonique entre la précision et le rappel.
- **Éléments clés :**
  - **Précision :**

$$\text{Précision} = \frac{VP}{VP + FP = TotalPrédit}$$

  - **Rappel :**

$$\text{Rappel} = \frac{VP}{VP + FN = TotalRéel}$$

- **Calcul de la F-mesure :**

$$\text{F-mesure} = \frac{2 \times \text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

F-mesures

- **Différents Types :**
  - **F1-macro :** moyenne des F1-scores de toutes les classes
    - ⋆ Traite toutes les classes de manière égale
    - ⋆ Plus sensible aux performances sur les petites classes
  - **F1-micro :** calcul basé sur le total des VP, FP, et FN
    - ⋆ Accorde plus de poids aux grandes classes
    - ⋆ Idéal lorsque vous voulez évaluer la performance globale sans être influencé par les déséquilibres de classes
  - **Weighted Average :** moyenne pondérée des F1-scores selon la fréquence de chaque classe
    - ⋆ Combine les avantages des deux
    - ⋆ Tenir compte à la fois de la taille des classes et des performances individuelles
- **Valeurs de la F-mesure :** de 0 (mauvaise performance) à 1 (excellente performance)

F1-weighted

- Supposons trois classes : NOUN (A), ADJ (B), et VERB (C) avec les scores F1 suivants :
  - Classe NOUN : F1-score = 0.8 (50 instances)
  - Classe ADJ : F1-score = 0.6 (30 instances)
  - Classe VERB : F1-score = 0.9 (20 instances)
- Calcul de la F1-weighted :

$$\text{F1-weighted} = \frac{(F1_A \times nb_A) + (F1_B \times nb_B) + (F1_C \times nb_C)}{nb_{total}}$$

Outils Python pour l'évaluation

- **Scikit-learn** : métriques de classification (précision, rappel, F1-score, ROC-AUC)
- **SpaCy** : évaluation des tâches de NER, POS tagging, parsing
- **Hugging Face Transformers** : évaluation des modèles pré-entraînés (BLEU, ROUGE, etc.)
- **Seqeval** : évaluation des tâches de séquence, comme NER
- **rouge-score** : calcul des scores ROUGE pour l'évaluation des résumés automatiques
- ...

Évaluation de la Structure

- Évaluer la capacité du modèle à détecter la structure et la hiérarchie des annotations
- Difficulté potentielle d'application des outils, parfois impossible
- Nécessité d'un script d'évaluation :
  - Parser les sorties des modèles (post-processing)
  - Comparer les résultats avec les références

Évaluation de la Structure - Exemple

- *Send everyone with a birthday today a happy birthday message*
  - ref: `[IN:send_message [SL:recipient [IN:get_contact [SL:date_time today ] ] ] [SL:content_exact happy birthday ] ]`
  - hyp: `[IN:send_message [SL:recipient everyone ] [SL:content_exact birthday today ] [SL:content_exact happy birthday ]`

Introduction à l'Apprentissage Automatique (Machine Learning)

Introduction à l'Apprentissage Automatique (Machine Learning)

- Sous-domaine de l'Intelligence Artificielle (IA)
- Utilisation des approches mathématiques et statistiques pour permettre aux ordinateurs d'« apprendre » à partir de données
- Conception et optimisation de modèles visant à minimiser l'erreur

L'Intelligence Artificielle (IA)

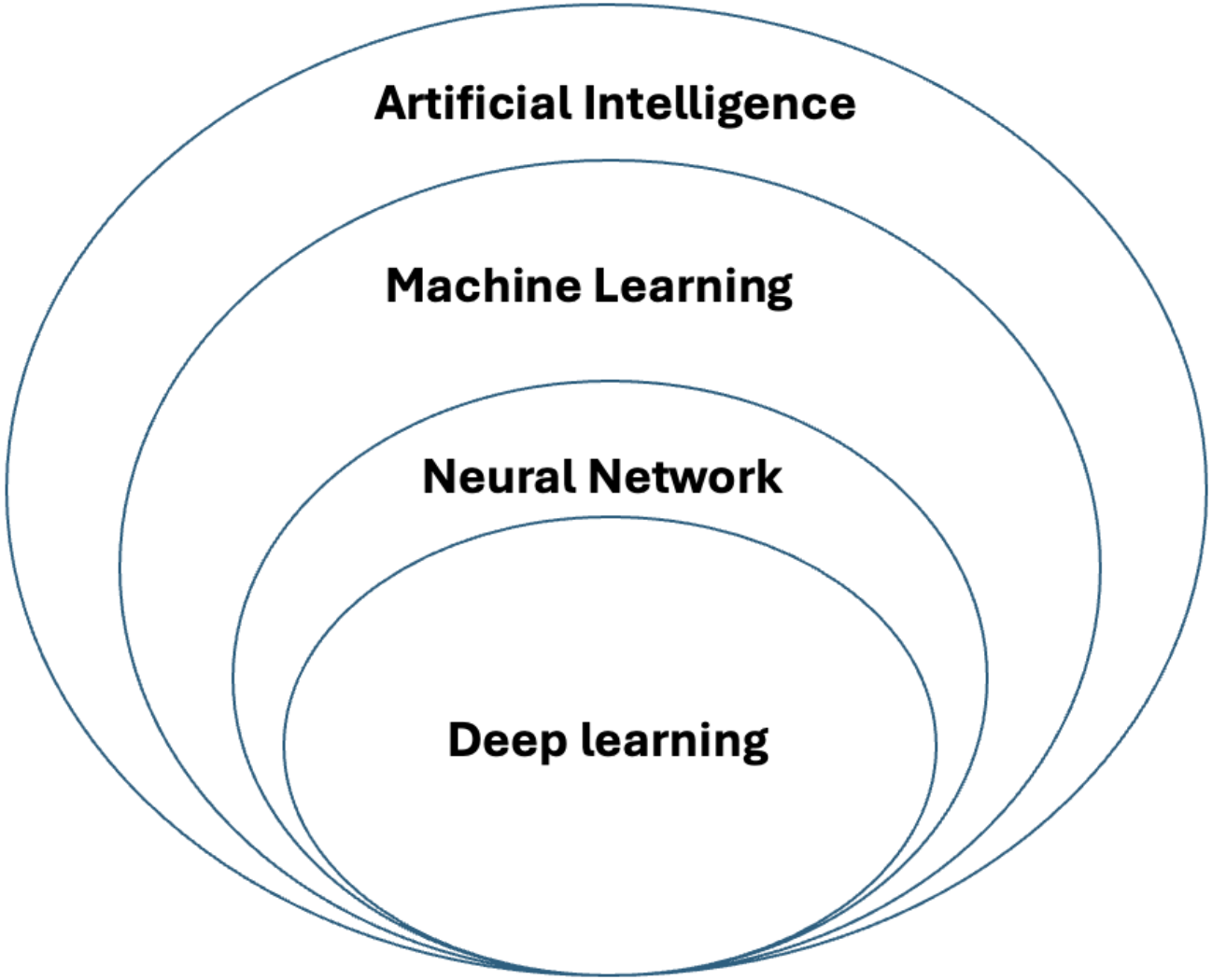
- L'IA: branche de l'informatique visant à simuler des comportements intelligents, y compris la capacité d'apprendre et de résoudre des tâches complexes
- Objectif: création des systèmes intelligents capables de résoudre des problèmes
- Capacité à effectuer des tâches nécessitant l'intelligence humaine
- Exemple : systèmes d'IA imitant le raisonnement humain

L'Intelligence Artificielle (IA)

- **Types d'IA :**
  - IA faible : résout des problèmes spécifiques avec des données et une aide humaine
  - IA forte : machines disposant de capacités cognitives proches de celles des humains (encore en recherche)

- **Historique de l'IA :**
  - 1950 : Alan Turing propose le test de Turing pour évaluer la capacité des machines à imiter l'intelligence humaine
  - 1959 : premier programme de machine learning capable de jouer aux dames
  - 1997 : Deep Blue d'IBM bat le champion du monde d'échecs

Intelligence Artificielle et Sous-Domaines



Introduction à l'Apprentissage Automatique (Machine Learning)

- Phases de l'Apprentissage Automatique
  - **Entraînement** : le modèle apprend à partir de données d'exemples
    - Apprentissage supervisé
    - Apprentissage non supervisé
    - Apprentissage par renforcement
  - **Inférence** : le modèle généralisé est appliqué à de nouvelles entrées pour effectuer des prédictions
    - Evaluation
- Un bon modèle de machine learning: un modèle qui généralise

Apprentissage Supervisé

- Le modèle apprend à partir de données étiquetées
- Pour chaque entrée, une sortie correcte est fournie pendant l'entraînement
- Objectif : prédire la sortie correcte pour des données inconnues

Exemples d'algorithmes d'apprentissage supervisé

- **Régression linéaire** : prédiction de variables continues, ex : score de sentiment dans les analyses de texte
- **SVM (Support Vector Machines)** : classification de textes pour identifier les spams ou catégorisation de documents
- **Random Forest** : combinaison d'arbres de décision pour la classification
- **Naive Bayes** : modèle probabiliste pour la classification de texte, souvent utilisé pour la détection de spams
- **CRF (Conditional Random Fields)** : utilisé pour les tâches de séquençage comme l'étiquetage de parties du discours (POS tagging)

Apprentissage Non Supervisé

- Le modèle apprend à partir de données non étiquetées
- Il identifie des structures ou des motifs cachés dans les données
- Principalement utilisé pour le regroupement (clustering) ou la réduction de dimensionnalité

Exemples d'algorithmes pour l'apprentissage non supervisé

1. **Clustering**
   - **K-Means** : partitionne les données en K clusters
   - **DBSCAN** : identifie des clusters basés sur la densité
2. **Règles d'Association**
   - **FP-Growth** : construit un arbre compact pour identifier les ensembles fréquents
3. **Réduction de la Dimensionnalité**
   - **t-SNE** : visualise des données à haute dimension en 2D ou 3D
   - **Autoencodeurs** : réduit la dimensionnalité en apprenant une représentation comprimée

Apprentissage Semi-Supervisé

- Combinaison d'apprentissage supervisé et non supervisé
- Utilise un petit ensemble de données étiquetées et un grand ensemble non étiqueté
- Permet de réduire les coûts de labellisation des données tout en améliorant les performances du modèle

Apprentissage par Renforcement

- L'agent apprend par interaction avec son environnement
- Système de récompenses et pénalités pour guider l'agent vers le bon comportement
- Utilisé dans des domaines où l'apprentissage par essai-erreur est nécessaire
- **Exemples** :
  - **Systèmes de trading automatique**
  - **Jeux vidéo**

Types d'Algorithmes : Classification vs Génération

- **Classification** : prédire la catégorie à laquelle appartient une donnée (ex: reconnaissance d'images, détection de spams)
- **Génération** : créer de nouvelles données similaires à celles observées (ex: génération de texte ou d'images)

La Classification

- **Principe** : prendre des entrées et les assigner à une classe parmi un ensemble de classes possibles
- **Algorithmes courants** : régression logistique, SVM, réseaux de neurones (LSTM, CNN, BERT...)
- Utilisée dans divers domaines comme la reconnaissance d'images, la classification de textes, détection de fraudes, détection du contenu haineux, etc.

Exemple 1 : Classification d'Images

- Utilisation de **réseaux de neurones convolutifs (CNN)** pour identifier des objets dans des images
- Entraîné sur des ensembles de données d'images étiquetées (ex: CIFAR-10, ImageNet)
- Capable de classer des images dans des catégories comme "chien", "chat", "voiture", etc.

La Génération

- **Principe** : générer des données à partir d'un modèle appris sur de données d'entraînement
- Utilisée dans des tâches comme la génération de texte, d'images ou de vidéos
- **Exemples de modèles** : GANs (pour la génération d'images) ou T5 (pour la génération de texte)

Exemple 1 : T5

- **T5 (text-to-text model)** : un modèle de langage génératif basé sur un encodeur-decodeur
- Entraîné sur plusieurs tâches de TAL en reformulant chaque tâche en un problème de génération de texte
- Applications variées :
  - **Résumés automatiques**
  - **Traduction automatique**
  - **Réponse à des questions**

Exemple 2 : GANs (Generative Adversarial Networks)

- **GANs** : utilisé pour générer des images réalistes (ex: visages synthétiques, œuvres d'art)
- Un réseau de génération crée des images, tandis qu'un réseau discriminateur évalue si elles sont réalistes ou non
- Ces modèles sont utilisés dans des applications comme l'amélioration d'images, la création d'avatars et la conception artistique

Importance des Hyperparamètres

- **Définition** : les hyperparamètres sont des paramètres fixés avant l'entraînement du modèle, contrairement aux paramètres qui sont appris

- Exemples : le taux d'apprentissage, le nombre de couches, la taille des batchs

Courbe d'Apprentissage

- **Principe** : montre la performance du modèle en fonction du nombre d'itérations ou du temps d'entraînement
- Utilisée pour diagnostiquer le surapprentissage (overfitting) ou sous-apprentissage (underfitting)
- Permet d'ajuster les hyperparamètres ou de décider si plus de données sont nécessaires

Problèmes rencontrés lors de l'Apprentissage

- **Insuffisance des données** : trop peu de données entraînent un sous-apprentissage
- **Coût en mémoire** : modèles complexes nécessitent des ressources importantes pour le stockage et l'entraînement
- **Surapprentissage (Overfitting)** : le modèle s'ajuste trop aux données d'entraînement, et généralise mal sur des données non vues

Méthodologie pour l'Apprentissage Automatique

Premier Constat : Tâche et Données

- **Tâche** : définir le type de problème (classification, régression, clustering, etc.)
- **Données** : analyse qualitative (types de données) et quantitative (taille, distribution)
- Identifier des tendances initiales ou des anomalies potentielles avant de commencer l'entraînement

Split des Données : Train, Dev, Test

- **Entraînement (Train)** : 70-80% des données pour ajuster les paramètres du modèle
- **Développement/Validation (Dev)** : 10-15%, utilisé pour ajuster les hyperparamètres
- **Test** : 10-15%, permet d'évaluer les performances du modèle sur des données jamais vues

Étude Statistique des Données d'Entraînement

- Analyse approfondie des données d'entraînement pour détecter des biais ou des problèmes spécifiques
- Calcul des statistiques descriptives (moyenne, écart-type, distribution des classes, etc.)
- Utilisation de visualisations (histogrammes, heatmaps) pour comprendre la structure des données
- Comparaison par rapport aux données Test (relever les OOV et les OOD par exemple)

Entraîner le Modèle

- Choisir l'algorithme d'apprentissage automatique en fonction de la tâche
- Ajuster les hyperparamètres pour optimiser les performances (ex: taux d'apprentissage, batch size, epochs, sequence-length...)
- Utiliser des techniques pour éviter le surapprentissage

Évaluation Globale et sur Sous-Parties Difficiles

- **Évaluation globale** : mesurer les performances du modèle sur l'ensemble du corpus test
- **Sous-parties difficiles** : évaluer sur des séquences spécifiques comme les plus longues ou les OOV
- Cela permet de mieux comprendre les limites du modèle et d'identifier des domaines d'amélioration

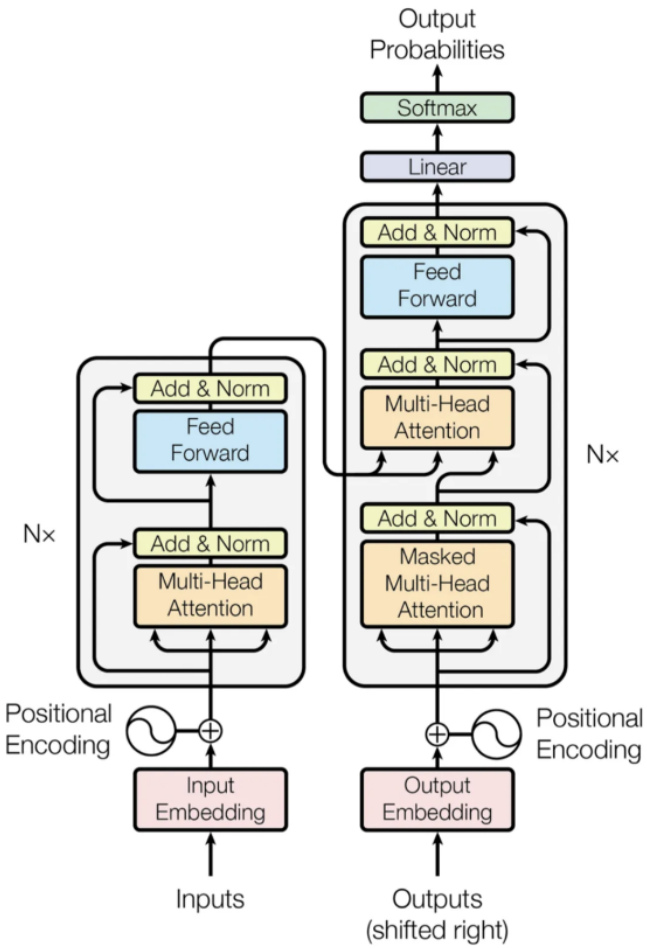
L'Apprentissage Automatique avec Hugging Face

- Des outils pour l'entraînement et l'évaluation des modèles NLP
- Intégration facile de modèles pré-entraînés pour des tâches de classification, traduction, génération de texte, etc.
- Utilise des modèles basés sur des architectures comme BERT, T5...

Modèle BERT (Bidirectional Encoder Representations from Transformers)

- **BERT** : la partie encodeur d'un Transformer, bidirectionnel qui lit un texte à la fois de gauche à droite et de droite à gauche
- **Transformers** : architecture qui capture les relations entre tous les mots d'une phrase via des mécanismes d'attention
- **Paramètres** : BERT-base contient 110M de paramètres, BERT-large 340M. Ajustables selon la tâche (fine-tuning)

Transformer



Transformer Model Architecture From 'Attention Is All You Need' by Vaswani et al.

BERT: Pré-apprentissage

- **Tâches** :
  - **Masked Language Modeling (MLM)** : masque 15% des tokens dans une phrase et apprend à les prédire en tenant compte du contexte bidirectionnel (gauche et droite)
  - **Next Sentence Prediction (NSP)** : prédire si deux phrases se suivent dans un texte pour capturer les relations entre phrases
  - Entraînement sur des données massives comme *BooksCorpus* et *Wikipedia* pour aider le modèle à développer une certaine "compréhension" générique du langage

BERT: Pré-apprentissage

- **Tokenization dans BERT** :
  - **WordPiece Tokenization** : décomposition des mots en sous-unités (par exemple, "playing" devient "play" et "##ing")
  - **Caractères inconnus** : les tokens rares ou inconnus sont décomposés en sous-mots pour gérer des mots hors-vocabulaire (OOV)

- **Attention Multi-têtes**
  - Utilisation de plusieurs têtes d'attention pour capturer différentes relations entre les tokens d'une séquence
  - Chaque tête d'attention se concentre sur différentes parties du texte, permettant au modèle d'apprendre plusieurs aspects du contexte en parallèle

Transfert-Learning

- Utilisation d'un modèle pré-entraîné sur un grand ensemble de données pour une tâche similaire
- Le modèle pré-entraîné sert d'extracteur de caractéristiques : ses couches sont gelées, et seules les nouvelles couches spécifiques à la tâche sont entraînées
- Exemple : utiliser BERT pour extraire des caractéristiques d'un texte et alimenter un classificateur de sentiments sans modifier les poids de BERT

Fine-Tuning

- Entraînement plus poussé : certaines couches du modèle pré-entraîné sont également dégelées et ajustées avec les nouvelles données
- Permet au modèle de s'adapter plus finement à la tâche spécifique, surtout si le nouvel ensemble de données est similaire à l'original
- Exemple : fine-tuning de BERT pour des tâches de reconnaissance d'entités nommées (NER) ou d'analyse des sentiments, où quelques couches internes sont ajustées

Avantages et Limites de BERT

- **Avantages** :
  - Traitement profond du contexte bidirectionnel
  - Performant sur de nombreuses tâches NLP, notamment en classification (classification des documents, POS, NER, SLU...)

- **Limites** :
  - Entraînement long et coûteux en calcul
  - Nécessite beaucoup de mémoire pour les grandes architectures
  - Assez limité par rapport aux modèles génératifs pour des tâches plus complexes

Hugging Face - Datasets

- Hugging Face propose des jeux de données publics pour l'entraînement et l'évaluation
- Les utilisateurs peuvent également charger leurs propres jeux de données via les bibliothèques **datasets**
- Format des données : JSON, CSV, ou texte brut, avec la possibilité de prétraiter les données selon les besoins de la tâche

Exemple d'Expérience : Comparaison POS Hugging Face et SpaCy

- Utiliser un modèle pré-entraîné de Hugging Face pour le **POS tagging**
- Comparer les résultats avec ceux obtenus via **SpaCy**
- Analyser les désaccords : cas où les deux modèles donnent des résultats différents, et comprendre pourquoi

Analyse des Résultats : SpaCy vs Hugging Face

- **Accords** : cas où les deux modèles sont d'accord pour identifier les étiquettes POS correctes
- **Désaccords** : cas où les modèles divergent, révélant des différences dans le traitement des données
- **Insights** : une analyse approfondie permet de mieux comprendre les forces et les limites de chaque modèle ainsi que des données

Éthique en Apprentissage Automatique

- Questions éthiques autour de l'impact des décisions prises par des algorithmes
- Transparence, équité, et responsabilité sont essentielles dans les applications d'IA
- Nécessité d'encadrer l'usage des algorithmes dans des contextes sensibles

TP-2 - Évaluation : Partie sous-corpus

- Chercher les séquences les plus longues (composées de plus de 29 tokens)
- Créer un sous-corpus à partir de ces séquences
- Conserver les **id** des séquences

TP-2 - Évaluation : POS sur tout le corpus avec SpaCy

- Générer un fichier au format :
  - `id-séquence | token | pos-ref | pos-spacy`

TP-2 - Évaluation : Sur tout le corpus

- a) Calculer Précisions, Rappels, F-mesures pour chaque classe
- b) Calculer F1-Macro, F1-Weighted à partir des F-scores de (a)

- c) Calculer F1-Micro :
  - Nombre total des prédictions correctes
  - Nombre total de la référence
  - Nombre total des prédictions pour toutes les classes
  - Calcul des Précisions, Rappels et F1-Micro

TP-2 - Évaluation : Performance du modèle sur le sous-corpus

- a) Extraire les séquences les plus longues avec leurs références et prédictions à partir des **id**
- b) Évaluer le sous-corpus