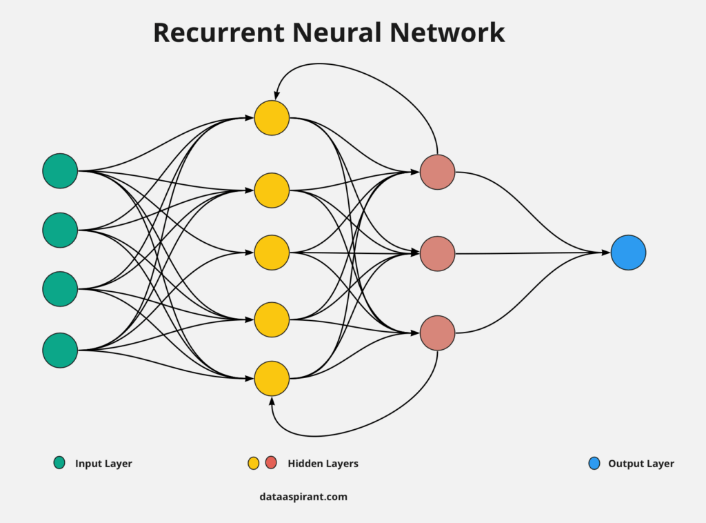
# TP8 : Prédictions sur le dataset IMDB avec un RNN

Ce TP vise à vous apprendre à appliquer et à évaluer un modèle de réseau neuronal récurrent (RNN) pour la classification de sentiments sur le dataset IMDB, tout en visualisant les résultats pour une meilleure compréhension des performances du modèle.



**Objectifs**

* Comprendre les étapes nécessaires pour préparer les données.
* Construire et entraîner un modèle RNN pour la classification de texte.
* Évaluer les performances du modèle.
* Visualiser les résultats.

**Matériel**

* Un ordinateur avec Python 3.x installé.
* Les bibliothèques Python installées : Pandas, numpy, scikit-learn, keras (tensorflow), matplotlib.

**Prérequis**

* Connaissances de base en Python.
* Notions fondamentales en deep learning.

**Données**

* Le dataset IMDB est un ensemble de 50 000 critiques de films extraites du site Internet Movie Database.
* Chaque critique est accompagnée d'une étiquette de sentiment : positive ou négative.
* Le dataset est équilibré avec 25 000 critiques positives et 25 000 critiques négatives.
* Il est divisé en deux ensembles (25 000 critiques d’entraînement, 25 000 critiques de tests)
* Vous pouvez l’importer directement depuis keras :

from tensorflow.keras.datasets import imdb

**Parties**

1. **Chargement des données (Importation):**
   * Importation des Bibliothèques: (keras.datasets.imdb et keras.preprocessing.sequence)
   * Configuration des Paramètres: (max\_features pour limiter le nombre de mots uniques = 10000) et (maxlen pour la longueur maximale des séquences = 500).
   * Chargement des Données:
     1. Utiliser imdb.load\_data(num\_words=max\_features) pour charger les données d'entraînement et de test.
   * Prétraitement des Données:
     1. Utiliser sequence.pad\_sequences pour tronquer/pader les séquences.
   * Afficher quelques exemples de données pour comprendre la structure
2. **Construction du Modèle RNN :**
   * Importation des Bibliothèques (Sequential, Embedding, SimpleRNN, et Dense de keras.models et keras.layers).
   * Créer une instance du model en Sequential.
   * Ajouter une couche Embedding pour convertir les mots en vecteurs.
   * Ajouter une couche SimpleRNN avec 128 unités pour traiter les séquences.
   * Ajouter une couche Dense avec une activation sigmoid pour la classification.
3. **Compilation du Modèle:**
   * Utiliser model.compile avec optimizer, loss et metrics.
4. **Entraînement du Modèle :**
   * Entraînement :
     1. Utiliser model.fit pour entraîner le modèle sur les données d'entraînement.
     2. Nombre d'époques = 10 et la taille de batch=32.
     3. Inclure une validation sur 20% des données d'entraînement.
5. **Évaluation du Modèle:**
   * Évaluation sur les Données de Test:
     1. Utiliser model.evaluate pour évaluer le modèle sur les données de test.
   * Affichage des Résultats:
     1. Afficher la perte (loss) et la précision (accuracy) des tests.
6. **Visualisation des Résultats :**
   * Importer matplotlib.pyplot.
   * Visualisation des Courbes de Perte ‘loss’:
     1. Tracer les courbes de perte pour les données d'entraînement et de validation.
   * Visualisation des Courbes de Précision ‘accuracy’:
     1. Tracer les courbes de précision pour les données d'entraînement et de validation.
7. **Prédictions:**
   * Utiliser model.predict pour faire des prédictions sur les données de test.
   * Interprétation des Prédictions:
     1. Interpréter et afficher les résultats des prédictions.

**Travail à rendre :**

Appliquer le modèle RNN sur la dataset reuters.

\* Elle contient 11 228 articles de presse provenant de Reuters.

\* Réparties en 46 catégories (finance, commerce, etc.).

\* A importer en utilisant :

from tensorflow.keras.datasets import reuters