Sorbonne Nouvelle
université des cultures

inalco
Institut national des langues et civilisations orientales

Université Paris Nanterre

Corpus arborés et parsing

Cours 11

Santiago Herrera
s.herrera@parisnanterre.fr

Tout ça pour quoi ?

- Pour comprendre le fonctionnement de la langue
- Disposant d'un modèle de syntaxe motivé linguistiquement, des phrases structurée et annotée, sur lesquelles on peut faire des opérations et de calculs…
- On peut donc faire des prédictions…

PARSING

Pourquoi le parsing automatique ?

Automatiser l'analyse syntaxique permet d'obtenir rapidement la structure syntaxique de grands corpus.

Pour :

- Des **outils** qui ont besoin d'une analyse fine de la phrase pour fonctionner
  (ex : génération de brevets)

- Des **linguistes** qui étudient les constructions d'une langue

- Des **apprenants** qui veulent comprendre en détail une phrase

- Des **professeurs de langues** qui construisent leurs cours en requêtant des corpus

- Des modèles de traduction qui utilisent la structure syntaxique pour produire la traduction

Analyse syntaxique basée sur des règles manuelles

Rule-based parsing

On définit manuellement un groupe de règles auquel on va s'y référer lors de l'analyse du texte.

Règles de Base :

- `S → NP VP` : Une phrase (S) est composée d'un groupe nominal (NP) suivi d'un groupe verbal (VP).
- `NP → Det N` : Un groupe nominal (NP) est formé d'un déterminant (Det) suivi d'un nom (N).
- `VP → V NP` : Un groupe verbal (VP) est constitué d'un verbe (V) suivi d'un groupe nominal (NP).
- `Det → "le" | "la" | "les"` : Un déterminant (Det) peut être "le", "la" ou "les".
- `N → "chat" | "chien" | "poisson"` : Un nom (N) peut être "chat", "chien" ou "poisson".
- `V → "mange" | "observe"` : Un verbe (V) peut être "mange" ou "observe" *

‘le chien mange’

Cela nécessite donc un travail coûteux de répertorisation de tous les Verbes, Adjectifs, etc…
On n'est pas encore dans de la "vraie" analyse automatique

Analyse syntaxique basée sur des transitions

On analyse la phrase dans le sens de la lecture, et pour chaque nouveau mot, on choisit s'il est dépendant d'un mot déjà analysé. On obtient donc les trois choix suivants :

- il n'est pas dépendant d'un mot de la pile => mettre le mot dans la pile (SHIFT)
- il est dépendant d'un mot de la pile => on note la relation et on met le mot de coté (LEFT|RIGHT ARC)
- il est gouverneur d'un mot de la pile => on note la R, on met mot le dans la pile et on y sort le dépendant

mots restants (buffer)	mot en cours	associations possibles (pile, stack)	action
[lui, donne, un, conseil]	Je	[ ]	mettre dans pile
[donne, un, conseil]	lui	[Je]	mettre dans pile
[un, conseil]	donner	[Je, lui]	gouverneur de Je et lui, on les sort de la pile et on met donner dans pile
[conseil]	un	[donner]	mettre dans pile
[ ]	conseil	[donner, un]	gouverneur de un (qu'on sort de la pile) , dépendant de donner
[ ]		[donner]	donner est la racine

Analyse syntaxique basée sur des transitions

Transition-based parsing

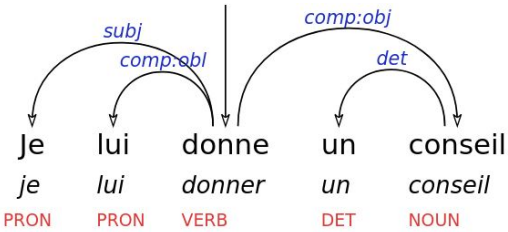
Cette méthode a deux problèmes :

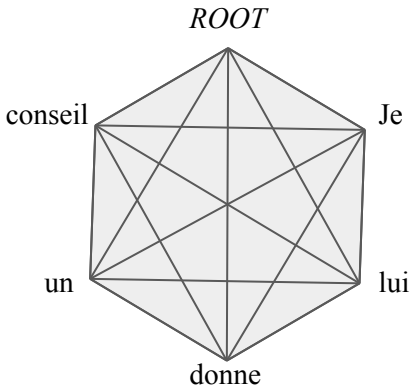
- les **choix sont locaux** ce qui diminue les performances pour les longues dépendances
- il n'est pas possible de produire des arbres **non projectifs** (cf DM)

Analyse syntaxique basée sur les graphes

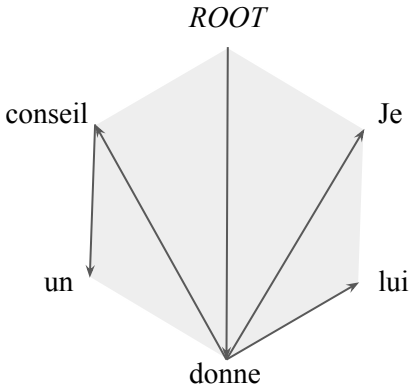
Graph-based parsing

- Chaque mot dans une phrase est considéré comme un nœud dans un graph.
- Des arcs (ou liens) potentiels sont établis entre chaque paire de mots. Ces arcs représentent les relations syntaxiques possibles.
- Chaque arc se voit attribuer un poids
- On cherche l'arbre qui maximise la somme de ses arcs (MST)





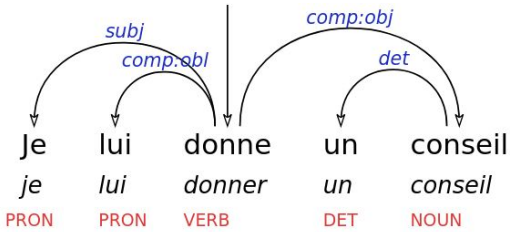
Maximum Spanning Tree

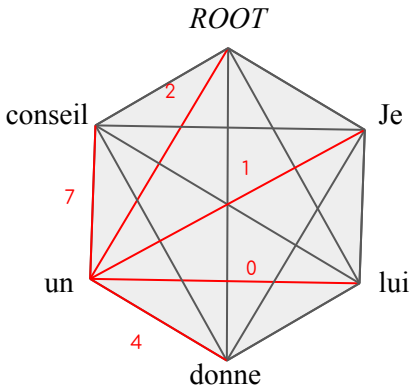


Analyse syntaxique basée sur les graphes

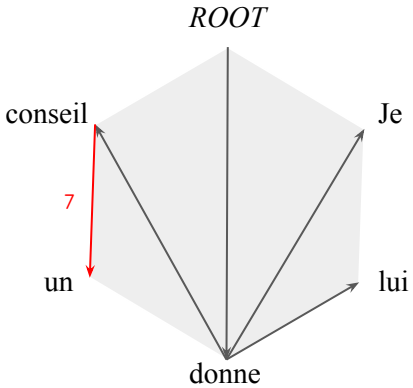
Graph-based parsing

- Chaque mot dans une phrase est considéré comme un nœud dans un graph.
- Des arcs (ou liens) potentiels sont établis entre chaque paire de mots. Ces arcs représentent les relations syntaxiques possibles.
- **Chaque arc se voit attribuer un poids**
- On cherche l'arbre qui maximise la somme de ses arcs (MST)

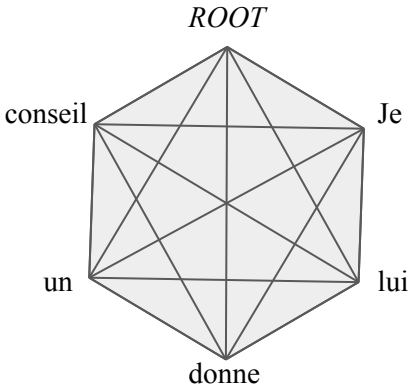




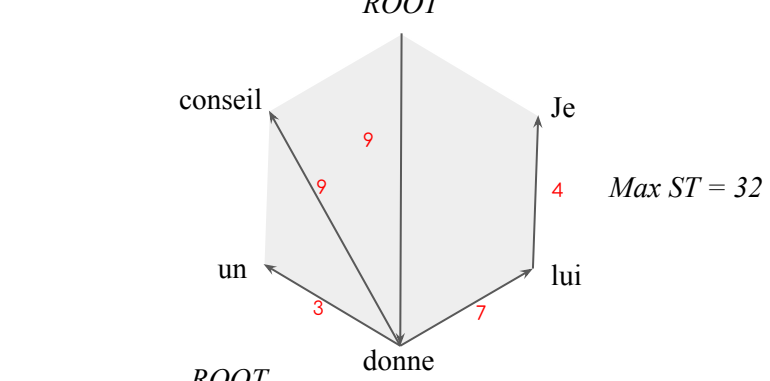
Maximum Spanning Tree



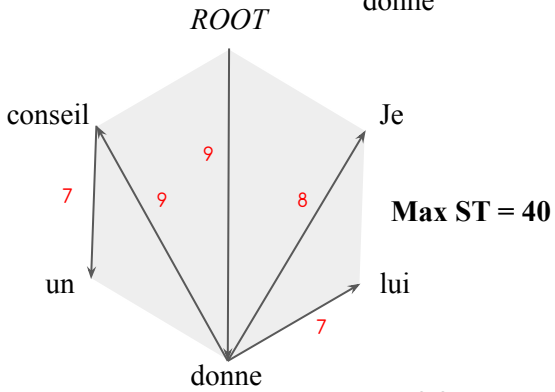
Maximum Spanning Tree



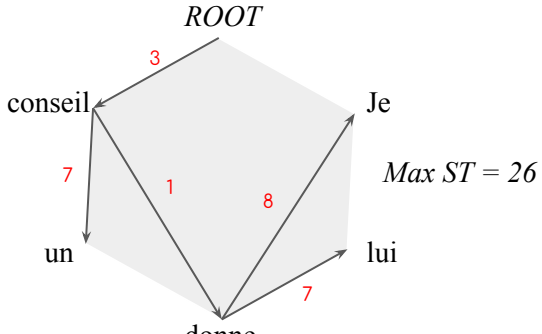
Maximum Spanning Tree



Max ST = 32



Max ST = 40



Max ST = 26

Apprentissage automatique

Machine learning

Apprentissage supervisé :

Les algorithmes apprennent à partir d'un ensemble de données étiquetées, essayant de faire des prédictions ou des classifications.

On cherche à apprendre F(X) = Y

exemples : prévision météo, traduction automatique, parsing, Speech-To-Text

Apprentissage non supervisé :

Les algorithmes explorent des données non étiquetées pour trouver des structures cachées ou des regroupements.

On cherche un nouvel espace qui définit nos données

exemples : algo de recommandation, détection d'anomalies

La fonction de coût

La fonction de coût **mesure l'erreur** ou le coût d'un modèle prédictif **par rapport aux données réelles**.

C'est elle qui va permettre de déterminer automatiquement comment modifier les paramètres du modèle

La descente du gradient

Gradient descent

1 - On évalue le modèle sur des données d'évaluation

2 - On entraine sur des données d'entrainement

3 - On réévalue le modèle sur les données d'évaluation

- si le coût **diminue** => **garder** la direction
- si le coût **augmente** => **inverser** la direction

4 - On fait les étapes 2 et 3 en boucle jusqu'à ce que le modèle "converge"

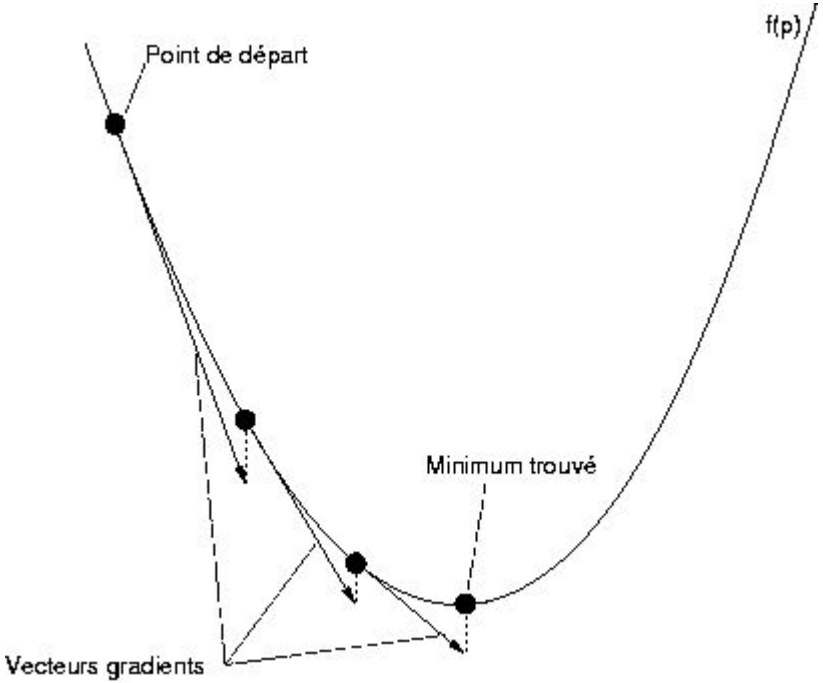
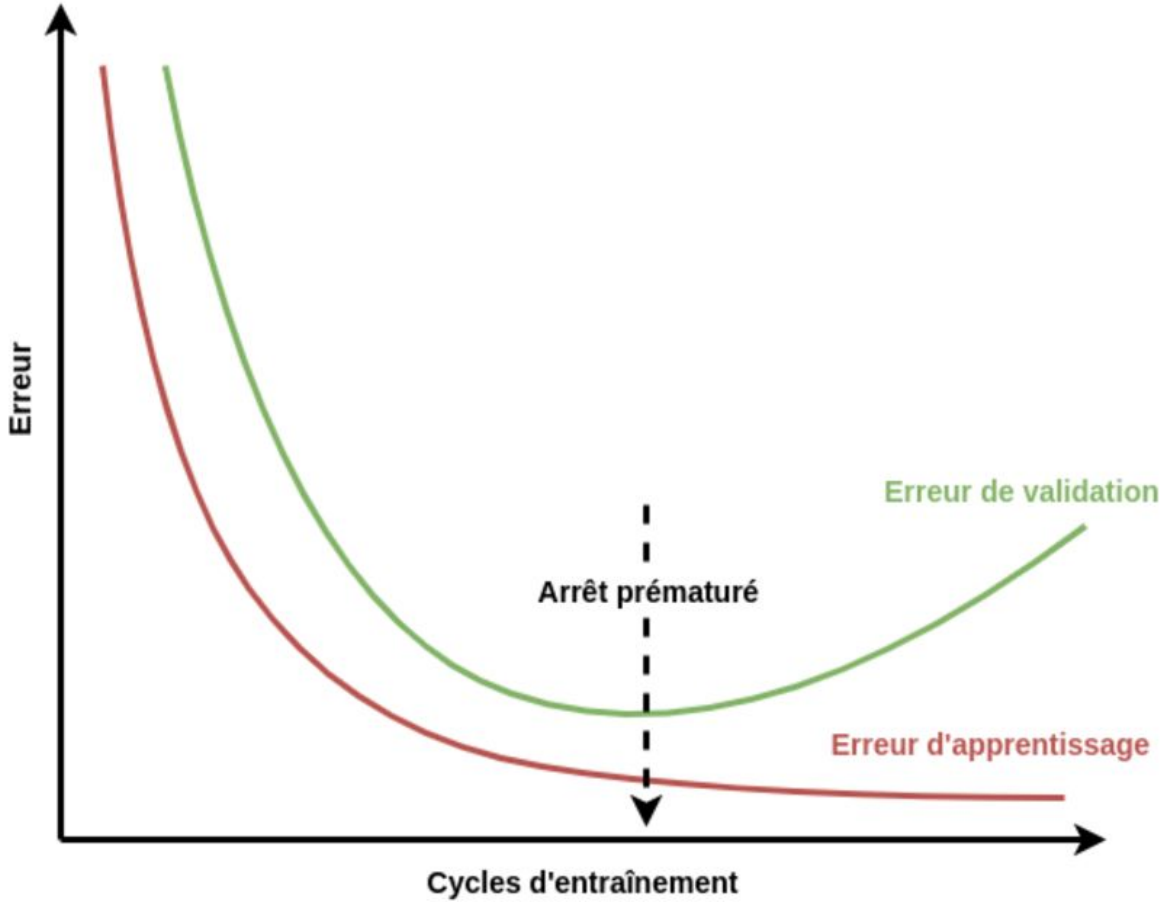


Fig : La descente du gradient (lien url)


Erreur
Erreur de validation
Arrêt prématuré
Erreur d'apprentissage
Cycles d'entraînement

Apprentissage automatique

Machine learning

Pour faire du ML, il nous faut des données numériques !

Apprentissage automatique

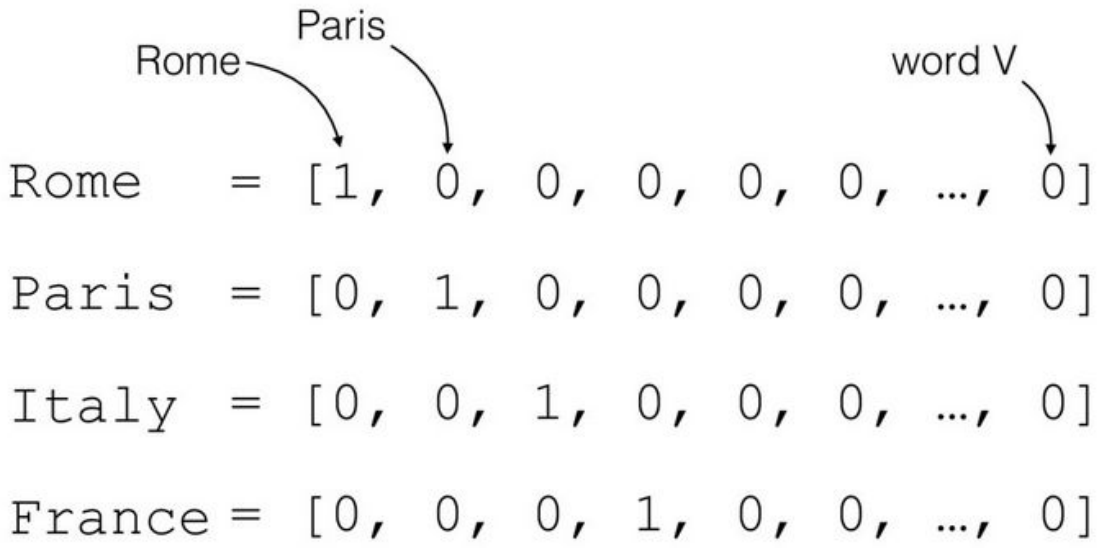
Machine learning

Pour faire du ML, il nous faut des données numériques !

(et il nous faut un objectif)

Comment transformer les mots en vecteurs ?

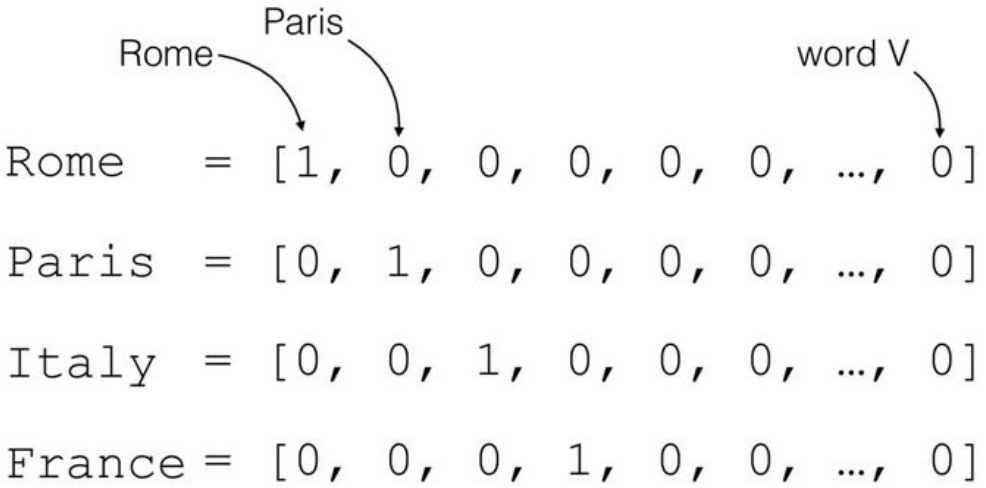
Pour une langue donnée de V mots, on attribue à chaque mot un vecteur unitaire de la forme :



Comment transformer les mots en vecteurs ?

Problèmes :

- Ça prend beaucoup de place (VxV, sachant qu'une langue a plusieurs dizaines de milliers de mots). Ce qui augmente donc la complexité des modèles se basant sur ces vecteurs
- On n'a pas de notion de "similarité" (le produit scalaire est nul pour chaque pair de mots)



$$sim(Rome, Paris) = sim(Rome, poire) = 0$$

Les plongements vectoriels statiques

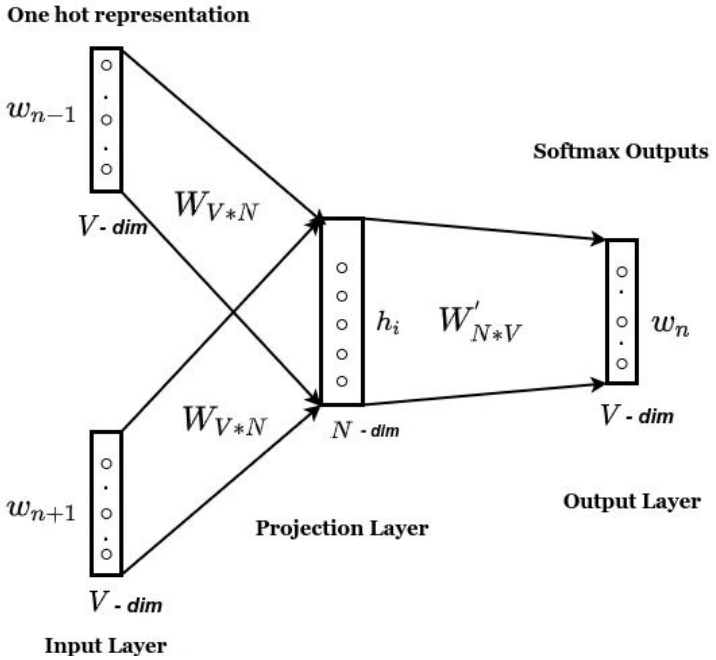
Word Embeddings

L'equipe de Mikolov (google) en 2013 a eu l'idée d'utiliser un perceptron (réseau de neurones) qui prend en entrée les vecteurs des mots d'une fenêtre entourant un mot masqué et qui prédit en sortie le vecteur unitaire de ce mot caché.

sim(Rome, Paris) = 0.8

sim(Rome, poire) = 0.1

sim(Rome, olive) = 0.3



Les plongements vectoriels statiques

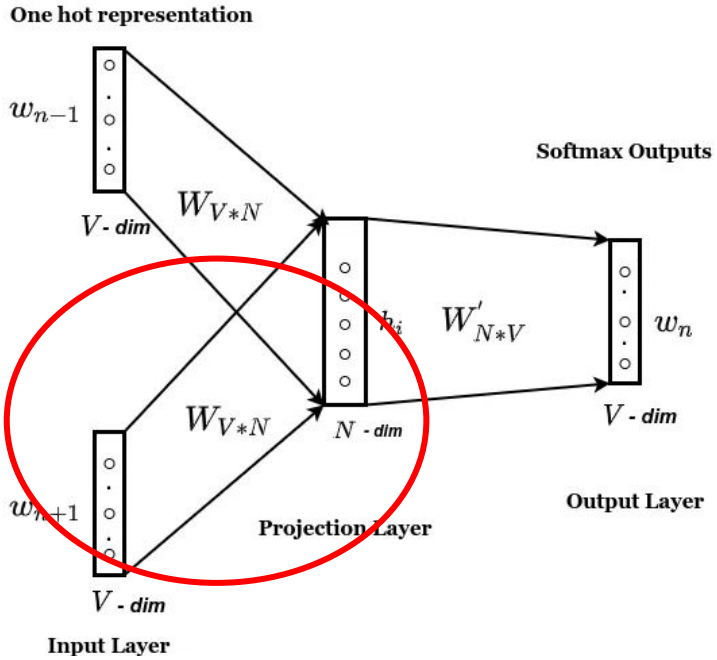
Word Embeddings

L'equipe de Mikolov (google) en 2013 a eu l'idée d'utiliser un perceptron (réseau de neurones) qui prend en entrée les vecteurs des mots d'une fenêtre entourant un mot masqué et qui prédit en sortie le vecteur unitaire de ce mot caché.

sim(Rome, Paris) = 0.8

sim(Rome, poire) = 0.1

sim(Rome, olive) = 0.3



Les plongements vectoriels

Word Embeddings

Les deux révolutions du NLP de ces 20 dernières années proviennent des word embeddings :

Word2Vec, Mikolov, 2013

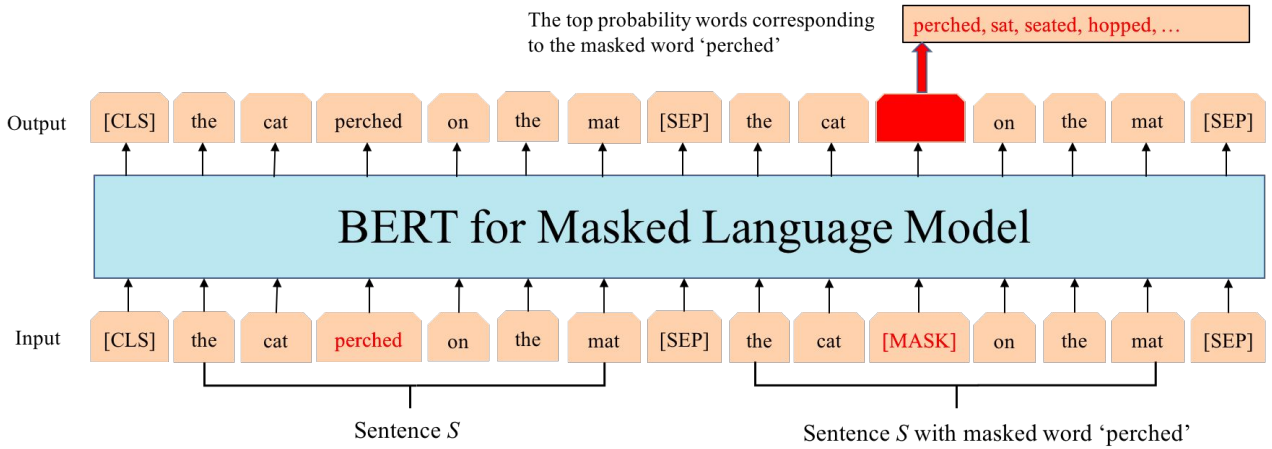
On transforme chaque token d'une langue en un vecteur fixe. Cela nous permet donc d'utiliser les algorithmes de machine learning déjà existants et utilisés dans d'autres domaines : clustering, classifieur, recherche de similarité

Transformer , 2017

Les vecteurs sont générés à la volée pour chaque phrase, ils sont contextuels et permettent de désambiguïser dans les cas de polysémie

Les plongements vectoriels contextualisés

Système permettant d'obtenir des embeddings contextualisés des tokens d'une phrase grâce à un mécanisme d'attention



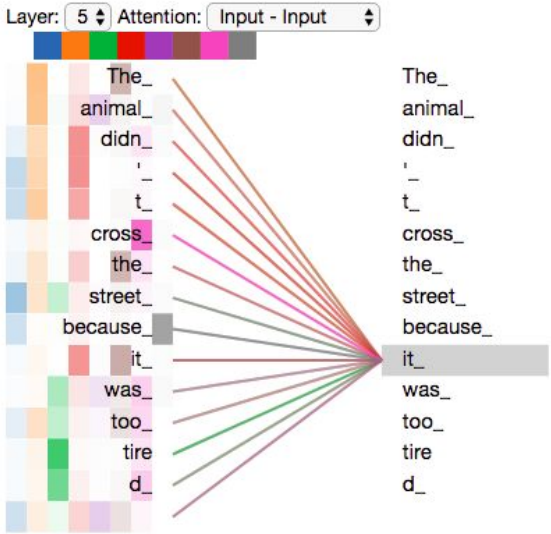


Figure : Exemple de modèle de prédiction de mots masqués

Visualisation Live

Et maintenant ?

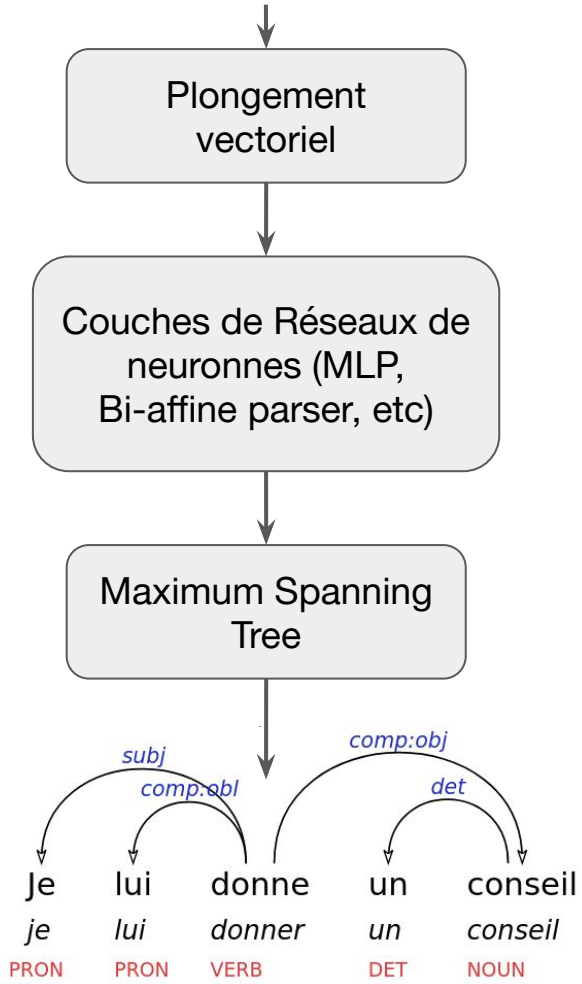
On sait :

- comment transformer les mots en vecteurs

- ~~comment obtenir une matrice de prédiction à partir de ces vecteurs~~

- comment obtenir un arbre depuis cette matrice

Je lui donne un conseil



La fonction de coût

La fonction de coût mesure l'erreur ou le coût d'un modèle prédictif par rapport aux données réelles.

C'est elle qui va permettre de déterminer automatiquement comment modifier les paramètres du modèle

L'evaluation

Pour l'analyse syntaxique, il est dur d'appréhender la performance d'un modèle seulement sur sa fonction de cout. On utilise les mesures suivantes :

LAS (Labeled Attached Score)
Ratio de dépendances avec le **bon gouverneur** et la **bonne étiquette**

UAS (Unlabeled Attached Score)
Ratio de dépendances avec le **bon gouverneur**

LUS (Labeled Attached Score)
Ratio de dépendances avec la **bonne étiquette**

Sources et liens utiles

[0] Lien vers l'exo : https://github.com/kirianguiller/M1_2023_initiation_au_parsing

[1] matrice d'adjacence vers graphes : https://graphonline.ru/fr/create_graph_by_matrix

[2] stanford CS224 : https://web.stanford.edu/class/cs224n/

[3] Analyse syntaxique automatique du pidgin-créole du Nigeria à l'aide d'un transformer, Guiller, 2020

[4] BertForDeprel : https://github.com/kirianguiller/BertForDeprel/