Réalisé par : **Rivaldo de SOUZA**

Glossaire

Dans la suite de ce document, nous rencontrerons un certain nombre de terminologies qu'il est important de définir.

Featurizer

Créer une représentation vectorielle du message et de la réponse de l'utilisateur (si spécifié) à l'aide de la fonctionnalité d'une fonctionnalité à préciser. Il existe divers algorithmes permettant de générer ce type de modèles. Plus concrètement, les algorithmes créent une représentation matricielle des données en entrée.

On distingue deux représentations :

- la représentation clairsemée :
  Elle consiste à stocker uniquement les mots - clés importants et leurs positions dans le vecteur (et par extension dans la matrice).
- la représentation dense :
  Elle consiste à stocker toutes les parties des phrases. Cela sous-entend une très grande consommation de mémoire.

Les **Featurizers** retournent deux types de matrices :

- la matrice à deux dimensions (“catégorie du mot (jeton)” X “le vecteur”)
  , le modèle est mieux détaillé.
- la matrice à une dimension (“liste des vecteurs”)

Tokenizer

Les **Tokenizers** divisent le texte en jetons. Si vous souhaitez diviser les intentions en plusieurs étiquettes, par exemple pour prédire plusieurs intentions ou pour modéliser la structure d'intention hiérarchique.

Réalisé par : **Rivaldo de SOUZA**

Les phrases sont décomposés en mots, ou en partie de mots. L'algorithme tient compte des séparateurs configurés. On peut donc tokenizer une phrase suivant les espaces, la ponctuation ou un mot clé.
Les connecteurs logiques peuvent par exemple permettre de détecter deux ou plusieurs sujets (intents) dans la même phrase.

Classifier

Les **Classifiers** associent au message entrant un "intent" en fonction du score de pertinence de celui-ci vis-à-vis du message. L' algorithme calcule le score de pertinence en se basant sur la casse, le ou les modèles déjà créé mais aussi les représentations vectorielles. Le résultat en sortie correspond à l'intent ayant le score le plus élevé.

N.E.R

N.E.R pour **Named Entity Recognition** :
également connu sous d'autres noms comme l'identification d'entité
ou l'extraction d'entités, est un processus de recherche et de classification des entités nommées existant dans le texte donné dans des catégories prédéfinies (les slots ou variables).
Cet algorithme permet de détecter et d'extraire efficacement des informations importantes susceptibles d'influencer la suite du processus.
On peut récupérer des noms, des valeurs, des listes d'éléments en analysant directement le texte de l'utilisateur.

Tagger

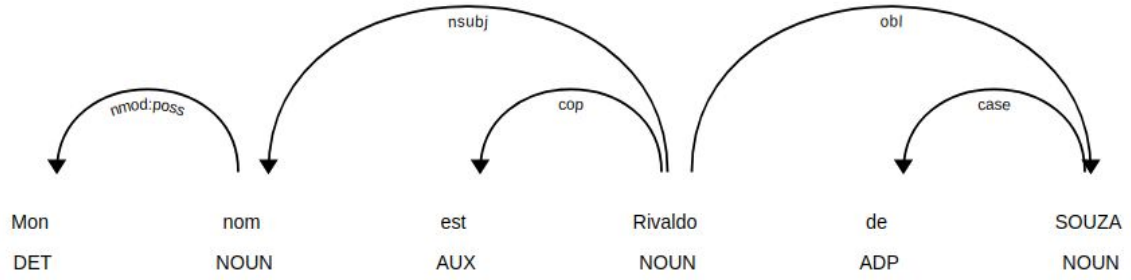
Le Tagging est un processus dans lequel vous lisez du texte et attribuez
à chaque mot ou jeton, une étiquette tel que nom, verbe, adjectif, etc.

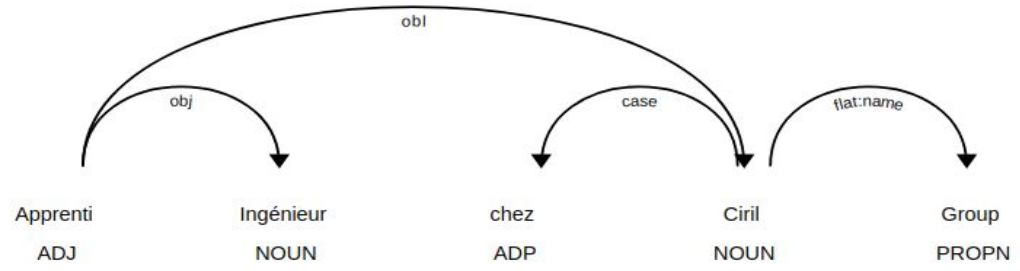
Il est donc en générale précédé d'une tokenisation. Une fois le tagging effectué on peut extraire les informations plus facilement.

Parser ou Dependency Parser

L'analyse des dépendances est l'une des fonctionnalités les plus belles et les plus puissantes. Rapide et précis, l'analyseur peut également être utilisé pour la détection des limites de phrases . Il détecte les relations entre les mots d'une phrase.

Exemple :
" Mon nom est Rivaldo de SOUZA "
" Apprenti Ingénieur chez Ciril Group"





Les modèles de Natural Language Processing

SpacyNLP

Points Positifs:

Spacy dispose de nombreux modèles pré-entraînés et disponible dans près de 86 langues dont le “Français”. En plus de ces modèles, Spacy permet de charger des vecteurs de mots et de les traiter avec l’algorithme Open-Source développé par Facebook, FastText.

Spacy fonctionne étonnamment bien dans les situations où, supposons que vous avez un énoncé comme: "Quel temps fait-il à Boston?". Quand nous formons notre modèle sur le même exemple d'énoncé, puis nous lui demandons de prédire l'intention de "Quel temps fait-il à Londres?" notre modèle est maintenant suffisamment intelligent pour savoir que les mots «Boston» et «Londres» sont similaires et appartiennent à la même intention.

Points Négatifs:

Cependant, les algorithmes de SpacyNLP ont de temps en temps des ratés liés à la qualité des données d’entraînement.

Réalisé par : **Rivaldo de SOUZA**

Précision en %	Précision NER	Précision de l'analyse syntaxique
5000 vecteurs	83.23	94.20
20 000 vecteurs	84.43	95.72
500 000 vecteurs	85.48	96.23

La précision augmente avec la quantité de données d'entraînement fournit.

Tensorflow_Embedding

Le modèle Tensorflow ne se base sur aucun modèle pré-entraîné mais s'adapte au jeu de données qu'on lui fournit.

Points Positifs:

Mode de Fonctionnement de Tensorflow :

En Français, le mot «jouer» peut être étroitement lié à «un sport» ou «une activité de plaisir ou de récréation», et cela peut sembler très différent du mot «acteur» au sens théâtral.
Dans un domaine théâtral, «jouer» et «acter» sont étroitement liés, où «jouer» signifie «participer à une forme de littérature écrit par un dramaturge », et il est très nécessaire de dire à notre modèle d'apprendre spécifiquement à notre domaine et ne pas être confus à cause de certains modèle pré-formé.

Lorsqu'on se trouve dans un domaine où le modèle a été entraîné, son efficacité moyenne est proche des 97%.

Points Négatifs:

Lorsque s'éloigne du domaine de prédilection du modèle, son efficacité se dégrade très vite. On est obligé de fournir des données en quantité et en qualité suffisante pour espérer un résultat acceptable.

- **HFTransformersNLP**
- **MitieNLP**
- **Sklearn**

Les trois modèles ci-dessus citées présentent les mêmes caractéristiques que Spacy à la seule différence qu'ils ne fournissent pas de modèles pré entraînés. Ils en attendent un venant de l'utilisateur.
Les algorithmes de traitement sont différents et la plupart du temps moins efficaces que ceux de Spacy.

Spacy vs Tensorflow

Spacy:

{

'intent':

{'name': 'get_horoscope', 'confidence': **0.8839278054416725**},

'entities': [],

'intent tracking': [

{'name': 'get_horoscope', 'confidence': 0.8839278054416725},

{'name': 'greeting', 'confidence': 0.05677650871638704},

{'name': 'subscription', 'confidence': 0.039641487657824596},

{'name': 'dob_intent', 'confidence': 0.019654198184116085}],

'text': 'I am looking for my horoscope for today. I am wondering if you can tell me that'

}

Cas du “Out of scope”:

{

'intent':

{'name': 'get_horoscope', 'confidence': **0.6745608070236074**},

'entities': [],

'intent ranking': [

{'name': 'get_horoscope', 'confidence': 0.6745608070236074},

{'name': 'subscription', 'confidence': 0.20373415140373155},

{'name': 'greeting', 'confidence': 0.08435193280944701},

{'name': 'dob_intent', 'confidence': 0.037353108763214043}],

'text': 'I am a robot'

}

Tensorflow:

{

'intent':

{'name': 'get_horoscope', 'confidence': **0.9664300680160522**},

'entities': [],

'intent ranking': [

{'name': 'get_horoscope', 'confidence': 0.9664300680160522},

{'name': 'dob_intent', 'confidence': 0.07643520087003708},

{'name': 'greeting', 'confidence': 0.0}, {'name': 'subscription', 'confidence': 0.0}],

'text': 'I am looking for my horoscope for today. I am wondering if you can tell me that'

}

Cas du “Out of scope”:

{

'intent':

{'name': None, 'confidence': **0.0**},

'entities': [],

'intent ranking': [],

'text': 'I am a robot'

}