# Travaux Dirigés d'informatique linguistique n°10

**Cours d'Introduction à l'informatique linguistique**

*—Licence Informatique 3ème année—*

---

## Réseaux sémantiques, levée d'ambiguités et clusters de mots

Dans ce TP, nous allons manipuler Wordnet un réseau sémantique à l'aide de NLTK pour faire de la levée d'ambiguité. Puis, nous implémenterons un outil pour extraire les différents sens d'un mot.

---

## 1 Ressources

Pour cette séance, nous vous fournissons un certain nombre de ressources se trouvant dans le répertoire suivant :
`http://igm.univ-mlv.fr/~mconstan/enseignement/l3/infolingu/data/`

- `wsd.py`, un module pour extraire les meilleurs voisins d'une occurrence d'un mot
- `td10.py`, un exemple d'utilisation du module précédent
- `textSpaceVector.py`, un module de manipulation des vecteurs de textes
- `wsd-data1.txt` et `wsd-data2.txt`, deux textes contenant sur chaque ligne des occurrences de *resistance*

Vous pouvez également vous aider des documentations sur NLTK, situées à l'url suivante :

`http://nltk.sourceforge.net/lite/doc/en/words.html`

L'auto-complétion dans `ipython` vous permet de découvrir des méthodes et leur documentation.

## 2 Wordnet et NLTK

1. Ecrire un script python pour obtenir l'ensemble des synsets du nom *resistance*.
2. Ecrire une fonction prenant un synset en paramètre et renvoie la liste des mots contenus dans le synset.
3. Ecrire une fonction qui prend un synset en paramètre et renvoie la liste de ses hyperonymes (les mots contenus dans les synsets hyperonymes)
4. Ecrire une fonction qui prend un synset en paramètre et renvoie la liste des hyponymes de ses hyperonymes

5. Ecrire une fonction qui prend un synset en paramètre et renvoie la liste des hyponymes des hyponymes de ses hyperonymes
6. Ecrire une fonction `voisins` qui prend un synset en paramètre et renvoie l'union de l'ensemble de ses hyperonymes, de l'ensemble des hyponymes de ses hyperonymes et l'ensemble des hyponymes des hyponymes de ses hyponymes.

## 3 Levée d'ambiguité sémantique avec Wordnet

Le nom *resistance* a 10 sens dans Wordnet. Le but de l'exercice est d'assigner automatiquement le bon sens de *resistance* à toutes les occurrences dans le texte `wsd-data1.txt`. Chaque ligne du texte contient un paragraphe comprenant au moins une occurrence de *resistance*. Chacune de ses lignes est suivie d'un séparateur de paragraphes sur une autre ligne.

Pour chacune des occurrences de *resistance*, il s'agira d'extraire la liste des noms voisins dans le texte et d'évaluer le recouvrement avec la liste des mots "voisins" dans Wordnet pour chacun des sens (synset) possibles. Le sens sélectionné sera celui qui a le meilleur recouvrement. Pour cela, vous vous aiderez de la classe `wsdData` dans `wsd.py` (cf. exemple d'utilisation `td10.py`). La classe `wsdData` permet pour chacun des paragraphes d'un texte d'extraire la liste des noms (méthode `getInstances`) et le `textVector` associé (méthode `getInstanceVectors`).

Nous supposons d'abord que les "voisins" d'un sens d'un mot dans Wordnet est la liste renvoyée par la fonction `voisins` implémentée ci-dessus.

1. Ecrire un script python qui permet d'assigner à chaque occurrence de *resistance* dans le texte un sens (un synset).
2. Evaluer les résultats.
3. N'y a-t-il pas moyen d'améliorer ces résultats ? Implémenter vos solutions.

## 4 Extraction automatique des sens d'un mot

Le but de l'exercice est d'extraire automatiquement les différents sens du mot `resistance` à partir du texte `wsd-data2.txt`. Implémenter l'algorithme du cours en Python. Que constatez-vous ? Comment améliorer les résultats ? Implémenter vos solutions.