Pré-soutenance du projet Désambiguïsation lexicale semi-supervisée

Sofiya Kobylyanskaya
Beyza Tasdelen

Université Paris-VII Diderot
M1 Linguistique Informatique
2019-2020

Table des matières

1.Introduction

L'objectif du projet est d'élaborer un système permettant de désambiguïser le sens des verbes ambigus qui peut varier en fonction du contexte dans lequel ces verbes apparaissent.
Nous allons traiter 5 verbes : abattre, aborder, affecter, comprendre, compter pour lesquels un inventaire de sens nous a été fourni permettant d'identifier les différents sens de ces verbes ainsi que leur contexte.
La réalisation du projet comprendra les étapes suivantes :

- la création des vecteurs des verbes en nous basant sur l'information obtenue
- l'application de l'algorithme de clustering semi-supervisé (k-means avec des contraintes et clustering hiérarchique) pour la classification des exemples. Nous utiliserons la méthode semi-supervisée, car celle-ci est plus adaptée à la tâche de désambiguïsation en tenant compte de la difficulté d'avoir un nombre de données suffisamment grand pour pouvoir utiliser la méthode supervisée.
- l'évaluation du système de classification en utilisant les métriques de base : la précision, le rappel, le F-score.
- l'amélioration du système

2.Étapes du travail

2.1 Création des vecteurs

La difficulté principale de cette tâche est de créer des vecteurs les plus représentatifs de chaque sens permettant de catégoriser les exemples de manière plus efficace. Pour former la liste de traits nous nous basons sur l'inventaire de sens et sur les relations syntaxiques des verbes en question.

La liste des caractéristiques pouvant être utilisées pour la formation des vecteurs :

- sujet animé/inanimé
- sujet présent dans l'inventaire de sens/non
- objet animé/inanimé
- objet présent dans l'inventaire de sens/non
- forme active/passive
- distance entre le sujet et le verbe (combien de mots sont entre eux)
- distance entre le verbe et l'objet
- présence/absence d'un COI
- présence/absence d'une préposition après le verbe

- root/non
- présence/absence d'un adverbe avant/après
- traits de co-occurrence (la fréquence du mots se trouvant avant ou après)

La plupart des traits sélectionnés sont binaires, donc les vecteurs risquent d'être très creux. C'est pourquoi nous proposons des possibilités d'amélioration dans la partie "Améliorations possibles".

2.2. Clustering

Le clustering est une méthode d'apprentissage non-supervisée, permettant de regrouper les données en clusters (en groupe) par leur ressemblance. Dans notre cas, un cluster correspond à un groupe de sens de chaque verbe.

Nous allons tester la version semi-supervisée du clustering, en mettant des contraintes sur la formation et la fusion des clusters.

2.2.1 Contraintes

Pour pouvoir mettre des contraintes sur la formation des clusters, nous utiliserons la méthode des contraintes de Pairwise. On peut distinguer deux types contraintes de Pairwise : la contrainte must-link qui exprime l'obligation de mettre deux ou plusieurs exemples dans le même regroupement et la contrainte cannot-link qui exprime l'impossibilité de mettre deux ou plusieurs exemples dans le même regroupement.

Deux algorithmes de clustering pourront être testé : k-moyennes (K-means) avec des contraintes, clustering hiérarchique.

2.2.2. Représentation des clusters

Pour la représentation des clusters dans les deux cas, nous utiliserons des classes distinctes : classe Hierarchical et classe Kmeans.

Un objet de chaque classe représentera un groupe de sens d'un verbe et contiendra comme informations : le centre et les exemples. Les méthodes de la classe permettront de calculer la similarité entre les clusters, de les fusionner (pour le clustering hiérarchique), d'y ajouter de nouveaux éléments (pour le clustering k-moyennes).

2.2.3 Clustering k-moyennes

K-moyennes est un algorithme non supervisé de clustering non hiérarchique. L'algorithme prend la valeur k en hyperparamètre comme entrée et produit k clusters différents qui sont formés autour de k vecteurs centraux choisis au hasard à la première étape. Le centre nous permettra de faire un calcul de similarité avec les autres vecteurs, d'attribuer lee vecteurs les plus proches aux clusters correspondants. Voici le fonctionnement de l'algorithme K-Moyennes étape par étape [2]:

1. Choisir l'hyperparamètre K : le choix du nombre de clusters dépend du nombre de sens possibles de chaque verbe dans l'inventaire.
2. On initialise k points au hasard qui seront les centres des k clusters.
3. On assigne chaque exemple des données au centre de cluster le plus proche. Ainsi, on fera une première répartition des données.
   Si un exemple se trouve à la même distance de plusieurs clusters, il sera assigné au cluster représentant le sens le plus fréquent.
4. On recalcule les centres de tous les k clusters et on redéfinit les centres.
5. On reprend les étapes 3 et 4.

La condition d'arrêt de l'algorithme peut être définie par trois critères :

- Le centre des nouveaux clusters ne change plus
- Les exemples ne changent plus de cluster
- Le nombre prédéfini d'itérations n est atteint

Le problème de l'algorithme de clustering k-moyennes consiste en l'initialisation aléatoire des centres des clusters qui peut entraîner une fausse classification des données. Afin d'éviter ce problème nous proposons d'utiliser l'algorithme K-moyennes avec des contraintes Pairwise [6].

Nous nous servirons de la contrainte cannot-link pour définir les centres des premiers clusters représentant un sens du verbe. Autrement dit, les premiers centres assignés ne pourront jamais être regroupés dans le même cluster. Nous viserons à respecter cette contrainte à chaque mise à jour des clusters. Les données qui nous ont été fournies ne nous permettent pas de créer la contrainte must-link, car nous ne savons pas d'avance le degré de similarité entre les exemples.

2.2.4 Clustering hiérarchique

Le principe du clustering hiérarchique consiste à initialiser le nombre de clusters au nombre d'exemples d'apprentissage (1 cluster = 1 exemple) et ensuite à fusionner ces clusters en fonction de la similarité des exemples, c'est-à-dire de la distance entre les vecteurs.
Afin de faire un apprentissage semi-supervisé, au moment de l'initialisation des clusters, nous allons choisir de manière aléatoire un exemple par groupe de sens et allons imposer la contrainte de telle sorte que les clusters correspondant à ces exemples ne puissent pas être

fusionnés entre eux. Ainsi, cela nous permettra de garder le nombre de clusters voulu qui correspond au nombre de sens du verbe dans l'inventaire.

On peut distinguer deux méthodes de clustering hiérarchique [3] :

- méthode agglomérative ou ascendante qui consiste à initialiser le nombre de clusters au nombre d'exemples d'apprentissage (1 cluster = 1 exemple) et ensuite à fusionner ces clusters en fonction de la similarité des exemples, c'est-à-dire de la distance entre les vecteurs.
- méthode descendante qui consiste à initialiser un seul cluster contant tous les exemples et ensuite à le diviser à un nombre de clusters plus petits.

Nous testerons la méthode ascendante et ensuite, comparerons le résultat avec celui obtenu avec la méthode K-moyennes avec des contraintes.
Comme pour la méthode K-moyennes, avant l'initialisation des clusters, nous calculerons la dissimilarité entre les exemples et imposerons la contrainte de telle sorte que les exemples les plus dissimilaires ne pourront pas se trouver dans le même cluster.
Nous ferons ensuite la fusion cluster par cluster en calculant leur similitude paire par paire. Plusieurs moyens de calcul de la similitude peuvent être testés (idem), par exemple :

- le calcul de la distance minimale entre deux exemples de clusters distincts
- le calcul de la distance maximale entre deux exemples de clusters distincts
- le calcul de la distance moyenne entre deux exemples de clusters distincts

L'algorithme se terminera au moment où le nombre de cluster prédéfini sera atteint.
La sortie de cet algorithme peut être représentée par un dendrogramme, une structure arborescente, illustrant les étapes de fusion des clusters.

2.2.5 Mesures de similarité

Différents moyens peuvent être utilisés pour mesurer la distance entre les clusters, par exemple la distance euclidienne, la distance Manhattan, la distance maximale et la distance Mahalanobis. En ce moment, il est difficile de savoir quelle métrique serait la plus efficace. On pourrait utiliser plusieurs moyens et ensuite comparer la performance du modèle.

2.3 Evaluation

Pour l'évaluation de nos deux classifications, nous utiliserons les métriques de base : la précision, le rappel et F-score. Nous utiliserons ces métriques pour comparer les résultats d'attribution des labels (le bon sens pour chaque exemple) qu'on obtient et les classes gold qui nous ont été fournies.

La précision consiste à calculer la proportion des réponses prédites correctement. Cela veut dire qu'on calcule la proportion d'objet correctement attribué à la classe i par rapport au nombre d'objets attribués à la classe i. Le rappel consiste à calculer la proportion des objets

correctement classées à la classe i par rapport au nombre d'objets classées gold dans la classe i . La précision a l'avantage de pénaliser les mauvaises prédictions par rapport au rappel. Le F-score nous permet de combiner ces deux mesures.

Pour la classe i quelconque:

$$Précision : \frac{Nbr\ d'objet\ classés\ correctement\ classés\ dans\ i}{Nbr\ d'objets\ totales\ attribués\ à\ la\ classe\ i}$$

$$Rappel : \frac{Nbr\ d'objets\ classés\ correctement\ dans\ i}{Nbr\ d'objets\ classées\ gold\ dans\ la\ classe\ i}$$

$$F-score : \frac{2PR}{P+R}$$

3. Conclusion et améliorations

La difficulté principale de ce projet consiste à choisir les traits les plus informatifs pour les vecteurs de telle sorte que l'algorithme de clustering détecte les similarités et les dissimilarités entre les exemples et fasse le moins d'erreurs possible pour la classification. La plupart des caractéristiques que nous avons sélectionnées sont binaires (-1,1), cela veut dire que nos vecteurs risquent d'être creux.
Une des améliorations possibles pourrait être de pré-entraîner les vecteurs avec le modèle Word2Vec afin d'obtenir les embeddings plus informatifs.
Une autre amélioration possible serait d'utiliser "Averages word embeddings" en moyennant les embeddings des mots contextuels.[5] Les mots contextuels peuvent être les mots qui apparaissent dans la fenêtre du mot en question (du verbe ambigu), par exemple 3 mots précédents et suivants, ou cela peut être les relations de dépendance/gouvernance, par exemple le sujet, le COD, le COI, etc.

Bibliographie

1. McInnes, Bridget T. and Mark Stevenson. (2014). "Determining the difficulty of Word Sense Disambiguation." Journal of biomedical informatics 47, 83-90 .
2. Fahim A. M., Salem A. M., Torkey F. A. and Ramadan M. A. (2006) "An efficient enhanced k-means clustering algorithm," Journal of Zhejiang University Science A., 1626–1633.
3. Bair E. (2013). Semi-supervised clustering methods. Wiley interdisciplinary reviews. Computational statistics, 5(5), 349–361.
4. Arthur, David & Vassilvitskii, Sergei. (2007). K-Means++: The Advantages of Careful Seeding. Proc. of the Annu. ACM-SIAM Symp. on Discrete Algorithms 8, 1027-1035.
5. Segonne V., Candito M., Crabbé B. (2019). Using Wiktionary as a Resource for WSD : The Case of French Verbs.

6. Basu, Sugato & Banerjee, Arindam & Mooney, Raymond. (2004). Active Semi-Supervision for Pairwise Constrained Clustering. Proceedings of the SIAM International Conference on Data Mining.