Université de Paris - Romane GALLIENNE et Cécile GUITEL – M1 Linguistique Informatique
Apprentissage pour le TAL 1

Développement d'un classifieur PoS multiclasse

Sommaire

Présentation générale du perceptron

Un classifieur PoS (Part of Speech) multiclasse permet d'étiqueter un mot par sa nature grammaticale (nom, adverbe, adjectif, auxiliaire, verbe, etc.). Il existe plusieurs méthodes pour construire un classifieur PoS (régression logistique, perceptron, SVM), et nous avons décidé de programmer un perceptron moyenné.

Un perceptron est un algorithme d'apprentissage supervisé. Pour prédire l'étiquette d'un mot dans son contexte, il attribue un score à chaque étiquette suivant les caractéristiques du mot et un vecteur de poids. Les étiquettes sont distribuées indépendamment et identiquement, ce qui veut dire que les prédictions sont indépendantes les unes des autres.

L'apprentissage se fait sur un premier corpus d'entraînement annoté par un oracle. Pour chaque mot du corpus, représenté par ses caractéristiques one-hot dans un vecteur de mots, le perceptron prédit l'étiquette la plus probable. Le vecteur de poids est revu selon le résultat. S'il se trompe, l'étiquette prédite est rendue moins probable, et l'étiquette oracle est rendue plus probable.

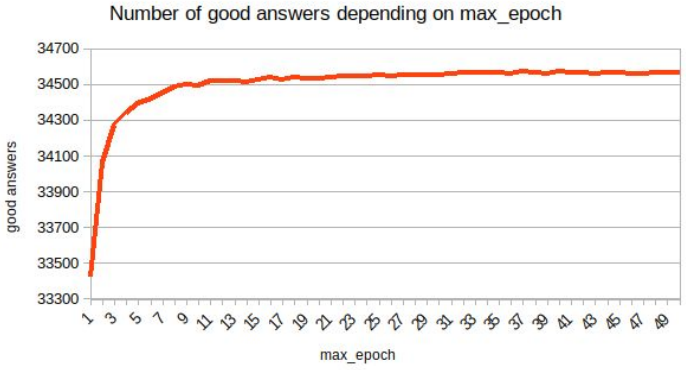
Spécificités de notre perceptron

Moyennage

Pour éviter que l'ordre des vecteurs dans le corpus n'influence le résultat, le perceptron est moyenné : on calcule plusieurs vecteurs de poids sur le même corpus, qui est mélangé à chaque fois, puis on fait la somme des vecteurs.

Il y a là une déviation au pseudo-code du cours. Nous effectuons le moyennage à chaque époque au lieu d'à chaque mot. Ce choix a été pris par souci de temps.

Le nombre de vecteurs de poids calculés correspond à MAX_EPOCHS, que l'on fixe manuellement. Pour cela, nous avons évalué la précision du système dans le domaine avec le corpus de développement, selon le nombre d'epoch. Cette précision commence vite à tendre vers une limite. Par souci de rentabilité du temps d'entraînement, nous avons donc choisi MAX_EPOCH = 10.



Sérialisation

Nous avons également décidé de sérialiser les vecteurs de poids afin de gagner en efficacité. Sérialiser permet d'éviter de recalculer à chaque fois les vecteurs de poids d'un corpus. Pour cela nous avons utilisé le module pickle[1]. Les vecteurs de poids sont sérialisés lorsque tous les epochs ont été faits.

Caractéristiques des vecteurs de mot

Pour extraire les caractéristiques de chaque mot, nous avons créé des vecteurs one-hot, stockés sous forme de dictionnaires. Pour rendre les vecteurs moins lourds, et donc améliorer l'efficacité du programme, nous n'avons ajouté que les caractéristiques qui étaient présentes pour le mot. Les caractéristiques extraites sont les suivantes :

- identité du mot,
- mot précédent ou mot en début de phrase,
- mot suivant ou mot en fin de phrase,
- première lettre en majuscule,
- toutes les lettres en majuscule,
- longueur 1, 2, 3 ou supérieure à 3,
- préfixes et suffixes : nous extrayons jusqu'aux 4 premières et dernières lettres du mot, selon sa longueur.

Exemple d'un vecteur pour le mot *est* :

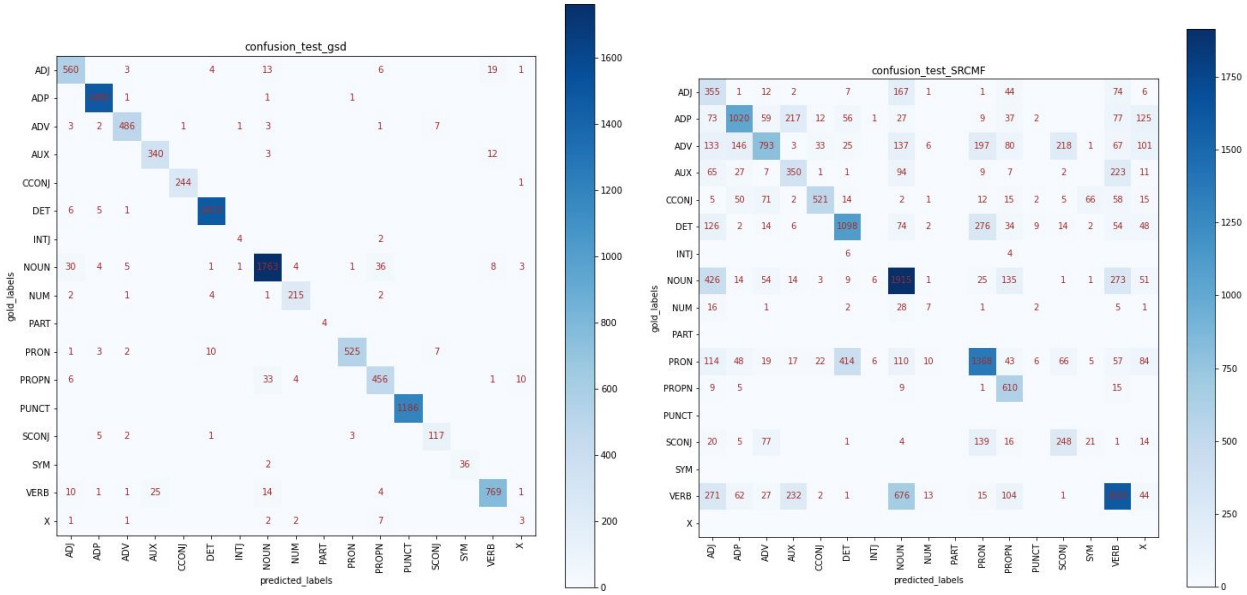
{'word=est': 1, 'word-1=,': 1, 'word+1=mise': 1, 'len=3': 1, 'prefix1=e': 1, 'prefix2=es': 1, 'prefix3=est': 1, 'suffix1=t': 1, 'suffix2=st': 1, 'suffix3=est': 1}

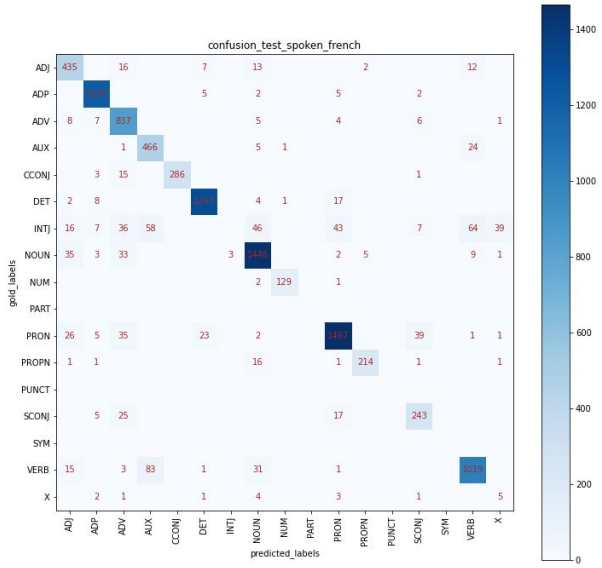
Résultats

Le perceptron est entraîné sur le corpus GSD French de *Universal Dependencies*. Nous utilisons les corpus French-Spoken et French-SRCMF de *Universal Dependencies* pour l'évaluation hors-domaine.

[1] Vous pouvez trouver la documentation sur le module pickle ici : https://docs.python.org/3/library/pickle.html

Confusions pour les différents fichiers test du corpus





Précision

	dans le domaine		hors domaine	
	train gsd	eval gsd	eval spoken	eval SRCMF
Bonnes réponses	347 369	9 649	9 062	10 233
Nombre de mots	354 662	10 019	9 991	17 358
Précision (%)	97,9	96,3	90,7	59,0

Problèmes soulevés par les corpus hors-domaine

On peut voir qu'il y a beaucoup plus d'erreurs sur le corpus de test SRCMF. Ceci s'explique par la différence de langue utilisée puisque le corpus SRCMF est tiré de la chanson de Roland en ancien français, qui est très éloigné du français contemporain. Nous pouvons voir là les limites de notre modèle : pour des données trop éloignées du domaine d'apprentissage, il est difficile d'améliorer la performance.

Aucune des interjections du français parlé (par exemple, "euh") n'ont été correctement classifiées. Il s'agit d'un nouveau vocabulaire qui ne peut être inféré par son orthographe. Il faudrait alors se baser sur sa place dans la phrase, mais les interjections sont très peu courantes dans le corpus d'entraînement (6 en tout). Outre le fait que les interjections pourraient jouer un rôle sémantique différent d'un domaine à l'autre, et donc ne pas présenter les mêmes caractéristiques syntaxiques, et donc ne pas être utilisées aux mêmes endroits dans la phrase, elles ne sont tout simplement pas assez présentes pour être évaluées de manière satisfaisante. Une autre piste serait de voir si on pourrait les apparenter à la ponctuation du corpus gsd. Le corpus de français parlé ne contient naturellement aucune ponctuation.

On remarque aussi une confusion fréquente des verbes et des auxiliaires. Il s'agit souvent de mots connus comme "est" qui présentent une ambiguïté syntaxique. La seule solution est d'utiliser le contexte pour dissocier les deux cas. Dans le corpus de vieux français, on remarque aussi beaucoup de noms catégorisés comme des verbes. Il s'agit souvent de mots nouveaux avec des suffixes ambiguës liés à la conjugaison des verbes du vieux français.

Améliorations possibles

Pour rendre le modèle plus robuste

Les caractéristiques que nous avions choisies au début du développement de notre modèle étaient basées sur des exemples pré-existants de perceptron de POS-tagging (cf bibliographie). Cette recherche préliminaire nous a permis d'avoir de relativement bons résultats assez rapidement.

Pour améliorer ces résultats, nous avons ajouté de nouvelles caractéristiques aux vecteurs de mots (on prend en compte mot-2 et mot+2, les préfixes et suffixes vont jusqu'à une longueur 4). Cependant, certains ajouts n'étaient pas pertinents : par exemple, la prise en compte de l'avant-avant mot et de l'après-après mot évitait une erreur sur le corpus d'évaluation du français parlé.

Si l'on avait des tâches à faire sur l'un des deux autres domaines, et que la précision obtenue n'était pas suffisante, on pourrait ré-entraîner le perceptron spécifiquement pour ce nouveau domaine. Plusieurs situations envisageables :

- on a accès à un corpus d'entraînement annoté dans le nouveau domaine, on peut alors calculer de nouveaux poids en l'utilisant,
- on a un corpus non annoté, on peut alors utiliser un modèle de langue pour sélectionner les phrases du corpus in-domaine les plus proches et ainsi créer un corpus d'entraînement chevauchant les deux domaines,

- on n’a pas de corpus suffisant pour créer un modèle de langue, ou le domaine est trop éloigné du domaine d’apprentissage pour trouver un chevauchement, mais il est possible de “traduire” l’un dans l’autre, auquel cas on peut faire une analyse manuelle de l’orthographe et de la syntaxe pour déterminer des règles systématiques de traitement des données.

Pour la suite

Il faudrait davantage généraliser le code afin de pouvoir lancer les différentes parties du programme en entrant directement dans la console le chemin vers les données à utiliser ainsi que les actions à accomplir.

Il faudrait restructurer les données pour standardiser les corpus durant leur extraction. Par exemple, seul le corpus GSD prend en compte la ponctuation. Le corpus SRCMF ne sépare pas les contractions en deux (par exemple, “du” ne devient pas “de le”).

Il faudrait restructurer les données et l’analyse pour permettre de différencier entre le vocabulaire connu et les nouveaux mots.

Il faudrait évaluer le taux d’erreurs des confusions en plus de leur nombre. En n’utilisant que le nombre total, on fait l’hypothèse que notre objectif est de réduire le nombre d'erreurs total, et que le corpus est représentatif du domaine cible en termes de fréquence des catégories grammaticales.

Organisation du programme

Afin d’avoir un code plus modulaire, nous avons organisé notre programme en plusieurs fichiers :

- *perceptron_basics.py* contient les fonctions liées au traitement des données ainsi que la fonction de prédiction. Il est utilisé par les autres fichiers et n‘a donc pas besoin d‘être lancé.

- *perceptron_seralisation.py* contient les fonctions de sérialisation et de désérialisation des vecteurs de poids utilisées par le reste du programme. Il est utilisé par les autres fichiers et n’a donc pas besoin d’être lancé.

- *perceptron_train.py* calcule et sauvegarde dans un fichier les poids du perceptron. Il doit avoir été lancé avant de pouvoir lancer les programmes d’évaluation. C’est aussi là que l’on peut évaluer MAX_EPOCH.

- *perceptron_evaluate_accuracy.py* évalue la précision du perceptron sur différents corpus et, potentiellement, imprime les erreurs pour pouvoir les analyser manuellement.

- *perceptron_evaluate_confusion.py* génère un heatmap de la matrice de confusion et extrait les trois types de confusion les plus fréquents.

Il y a donc trois phases à l’utilisation du perceptron :

- *train*, avec en paramètre modulable dans le code le nombre d’époques et l’option d’évaluer la précision selon le nombre d’époques,
- *evaluate_accuracy*,
- *evaluate_confusion*, où il faut choisir le corpus à analyser et le titre du graphique à créer.

Sitographie

BELLECK A., *Part-of-Speech tagging tutorial with the Keras Deep Learning library* [en ligne]. https://becominghuman.ai/part-of-speech-tagging-tutorial-with-the-keras-deep-learning-library-d7f93fa05537 (consulté le 11 décembre 2020)

HONNIBAL M. , *A Good Part-of-Speech Tagger in about 200 Lines of Python* [en ligne]. https://explosion.ai/blog/part-of-speech-pos-tagger-in-python (consultée le 11 décembre 2020)

Python Software Foundation, *Pickle* [en ligne] https://docs.python.org/3/library/pickle.html (consulté le 15 janvier 2021)

WISNIEWSKI G., *Perceptron moyenné* [en ligne]. https://moodle.u-paris.fr/pluginfile.php/649525/mod_resource/content/1/perceptron.pdf (consulté le 11 décembre 2020)