Rapport du projet Python

Auto-complétion de saisie

Mengwei YANG, M2 R&D, Université Paris 3

Xi RONG, M2 IM, INALCO

Yunbei ZHANG, M2 IM, INALCO

I. Présentation

Ce projet a été réalisé dans le cadre du cours Python de M2 Traitement Automatique des Langues, il consiste à réaliser une auto-complétion d'un champ de formulaire. L'objectif est de proposer une complétion de saisie en multi-mots en tant qu'une interaction dynamique sur une page web. À la sortie, nous aurons un site web étant une interface de moteur de recherche, il s'agit de 2 perspectives de l'auto-complétion :

Quand l'utilisateur entre un ou plusieurs caractères, l'interface du moteur de recherche va lui proposer une liste de mots qui commencent par les caractères entrées. Par exemple ; quand on entre une chaîne de caractères 'il', il propose des mots comme "ils", "illustre", etc. Ces candidats sont ordonnés selon leurs fréquences dans notre corpus de base, c'est-à-dire qu'il affiche à priorité les mots les plus fréquentés dans le corpus ; D'ailleurs, une fois l'utilisateur choisit un mot proposé dans la liste (ou entre un mot pertinent qui apparu dans notre corpus), et entrée un espace blanc, il va par la suite lui proposer une liste de mots candidats concernant le mot entrée précédemment (le dernier mot). De plus, quand l'utilisateur entre un mot assez complet et il n'y a pas assez de candidates mots proposés, il proposera des mots dépendant du mot entrée, par exemple quand l'utilisateur entre ' conventionnel ', pour compléter ce mot, il n'y a que des candidates comme ' conventionnelle ', ' conventionnelles ', il propre donc des candidates comme ' conventionnel de ', ' conventionnel par ', etc.

Du à un temp limité, nous n'avons fait qu'un dictionnaire de cooccurrences combinées de deux mots, en conséquence, le mot proposé dépend juste du dernier mot entré, par exemple quand on entre "est ", il va proposer une liste de mots comme 'est pas', 'est un', etc. Mais quand on entre 'il est', il propose une liste comme 'il est pas', 'il est un', le mot "il" n'influence pas la proposition.

Au départ nous nous limitons à une complétion de mot à partir des caractères saisis par l'utilisateur, par la suite, nous allons surtout mettre en place les techniques permettant la complétion sur une interface web, pour y achever, nous avons employé une bibliothèque JavaScript côté client : Jquery UI et un serveur Python en mode REST pour les données : Flask.

II. Ressources utilisées

Notre corpus utilisé fait partie du site "Statistical Machine Translation" qui consiste à rechercher en traduction automatique statistique. Il propose une grande quantité de corpus parallèle. Nous nous intéressons au corpus parallèle Europarl, qui est un extrait des travaux du Parlement européen. Il comprend des versions en 21 langues européennes. Nous avons enfin choisit celui de la version 7, qui est une version élargie et améliorée du corpus. Étant donné que la taille de corpus est très volumineuse, nous avons extrait des données uniquement en français, et les stocké dans un fichier nommé corpus_fr.txt avec une phrase par ligne, il y en total 5000 000 lignes dans ce fichier.

Tous les mots de ce corpus composent nos lexiques.

Pour la partie de notre script, il s'agit de 5 modules utilisés : *flask, re, Collections, pickle et nltk*. *Flask* est un framework open-source de développement web en python, son but principal est d'être léger, afin de garder la souplesse de la programmation Python, associé à un système de templates. Ensuite pour le module *re*, l'expression régulière, ce qui nous semble très familière, nous l'avons utilisé pour le nettoyage du corpus et obtenir des lexiques consiste uniquement des mots en français. Par ailleurs le module *Collections* nous permet d'utiliser des "specialized container datype" comme *Counter* qui sert à calculer des fréquences des mots. Le module *pickle* est super pratique, il permet de sauvegarder dans un fichier, au format binaire, n'importe quel objet Python. Il permet de stocker et de restaurer un objet Python tel quel sans aucune manipulation supplémentaire, il fonctionne comme le module json mais n'est pas limité à un seul format d'objet. Plus précisément, on l'a utilisé pour manipuler le dictionnaire sans répéter le processus de traitement du corpus pour générer le dictionnaire. En fin le module *nltk* (Natural Language Toolkit) qui est une bibliothèque logicielle en Python permettant un traitement automatique des langues, dans notre projet, on l'a utilisé pour tokeniser les mots.

III. Fonctionnalité réalisé et le techniques utilisés

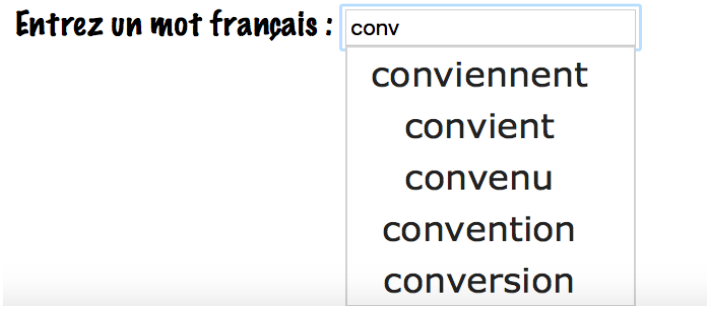
Nous avons implémenté un programme qui réalise 2 fonctionnalités de l'auto-complétion, dont l'un est de compléter le mot en fonction des caractères que l'utilisateur fait entrée si le motif est présenté dans notre corpus ; l'autre fonctionnalité concerne la complétion des n-mots suivants selon le dernier mot entré par l'utilisateur.

Au niveau de techniques utilisées, nous avons implémenté 2 programmes. Le scipt *pair.py* consiste à nettoyer le corpus initial et à préparer les pairs de mots ainsi que leurs fréquences. Dans ce script, on a utilisé les modules re, collection, pickle et nltk. D'abord, nous avons traité le corpus, les tâches à ce niveau consistent à enlever les chiffres et les ponctuations. Après avoir nettoyé le corpus, pour chaque mot dans une phrase, on a tiré le mot et le mot suivant comme une paire, plus précisément, il s'agit de calculer la fréquence de chaque mot, et la fréquence d'une paire de chaque mot et le mot qui le suit avec Counter. À la sortie : deux dictionnaires *pairDictionary.txt wordFrequenceDictionary.txt* sauvegardés sous format binaire de *pickle*.

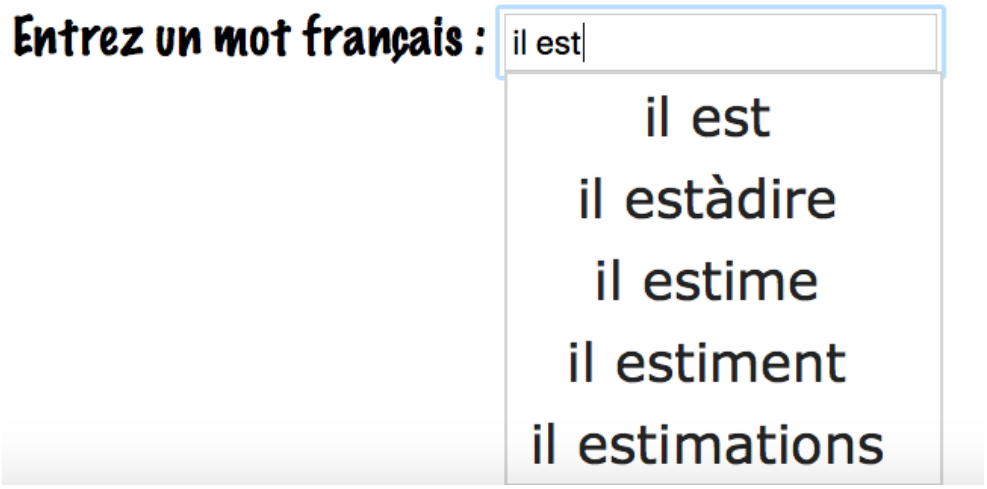
Le script nommé *hello_copy.py* est notre script principal qui a pour réaliser les fonctionnalités conçus à partir de notre dictionnaire à l'aide du module *flask*, le script prend une chaine de caractères en entrée et propose une liste de l'auto-complétion selon les différents de cas :

Le premier fonctionnalité concerne la complétion d'un seul mot selon les caractères entrés par l'utilisateur. Si les propositions ne sont pas suffisantes, on vérifie si le dernier mot est un mot assez complet et on peut trouver ses mots dépendants, si oui on retourne

une liste de proposition y compris le mot proposé et ces dépendants, par exemple si on entre 'conventionnel', il retourne 'conventionnelles', 'conventionnelle' et 'conventionnel de', voici un exemple de l'entrée :



Le deuxième cas est l'entrée combinée par plusieurs mots et l'usage ne type pas espace après dernier mot dans ce cas, la sortie doit être inclue les chaine de caractères entrés et le mot proposé, ou des mots dépendants proposés par exemple si l'entrée est "il est", il retourne des propositions comme 'il estime', voici le résultat de proportion :



IV. La conclusion et les difficultés :

A la fin du travail, nous avons réalisé une interface de moteur de recherche qui propose soit un mot en fonction des caractères entrés par l'utilisateur, soit les mots suivants. Le résultat est plutôt satisfaisant, cependant, nous avons rencontré des difficultés au cours de l'implémentation.

La première difficulté concerne le problème de l'apostrophe. Nous nous demandons comment segmenter le mot comme "j'ai", "l'un", "aujourd'hui", car l'apostrophe peut être au sein d'un mot ou étant un séparateur de deux mots. Nous avons enfin choisit de garder l'apostrophe, et le considérer comme un mot. Pour obtenir un résultat pertinent, il faut ajouter un espace blanc avant et après l'apostrophe. Voici un exemple du résultat :

La deuxième difficulté est que la taille du corpus est très volumineuse, donc c'est difficile d'apercevoir des caractères incorrects dans le corpus, par exemple : "aujourdâ€™hui". Nous n'apercevons ce problème jusqu'a quand on fait le test, comme le corpus est gros c'est difficile de trouver ces mots anormaux. Avec cette expérience, nous saurons la prochaine fois vérifier d'abord des mots bizarres avant traiter le corpus.

La dernière difficulté est que quand l'entrée est une chaine de caractères combinés de plusieurs mots, s'il renvoie une liste de mots dépendants ou de mots complets, il y aura des problèmes : quand on entre "il est ", ce qu'on veut obtenir est "il est pas", "il est un" au lieu de un seul mot "un" ou "pas". Donc quant à une chaine de caractères de plusieurs mots, il faut renvoyer une liste de proposition de chaine entrée suivi chaque mot proposé.