USTHB, Faculté d'Electronique et d'Informatique
Département d'Informatique
Module Traitement Automatique du Langage Naturel
Année Universitaire 2020 – 2021

Mini-Projet TALN
Implémentation d'un générateur automatique de grammaire pour la langue anglaise ou française

Date de Remise: Samedi 06 Mars 2021 à 14h00

Recherche de corpus faite par Mme Essma Selab, Doctorante

But du Mini-Projet :

A travers ce mini-projet vous pourrez apprécier de façon pratique la puissance des techniques de Traitement Automatique du Langage Naturel. Le problème qui vous est proposé ici vous permettra d'implémenter un générateur automatique de grammaire pour la langue anglaise ou française sous python en vous basant sur le travail fait pour la langue arabe dans (Selab & Guessoum, 2015-Arabic Grammar Induction).

Motivation :

L'analyse syntaxique (grammaticale) est une étape fondamentale du TALN. Bien que pour certaines applications du TALN un étiquetage morpho-syntaxique soit suffisant, l'analyse grammaticale complète est souvent utile. Une application potentielle serait dans un environnement d'enseignement/d'apprentissage intelligent où des textes/phrases pourraient être générées aléatoirement (selon le profil et l'évolution du niveau de l'apprenant) et des exercices d'analyses grammaticales suggérés et corrigés de façon automatique en exploitant une grammaire de la langue étudiée. Il s'avère que la programmation d'une grammaire d'une langue, i.e. un analyseur grammatical/syntaxique (parser) n'est pas une tâche aisée. En effet, couvrir toutes les structures syntaxiques d'une langue est une tâche laborieuse pour les experts en linguistique. Quand bien même ceux-ci seraient excellents, une programmation qui prendrait en compte toutes les structures, quel que soit le texte traité, peut montrer diverses lacunes quand les phrases deviennent plus longues et plus sophistiquées.

Les techniques de TALN basées sur les statistiques et l'apprentissage (machine) automatique ont permis des avancées spectaculaires du domaine. C'est pourquoi, une façon de programmer un analyseur syntaxique d'une langue donnée peut être envisagée par l'induction (apprentissage automatique) d'une grammaire à partir d'un corpus volumineux et suffisamment représentatif de cette langue, et, une fois ceci fait, l'utilisation de la grammaire induite comme ressource de l'analyseur syntaxique, comme le module de parsing de NLTK sur Python. C'est exactement ce qui est demandé dans ce mini-projet.

L'idée ici est de reprendre l'algorithme développé par Selab et Guessoum (Selab & Guessoum, 2015-Arabic Grammar Induction) dont le papier est ci-joint pour l'induction d'une grammaire propositionnelle de la langue arabe, l'implémenter en Python, et le réutiliser pour la langue anglaise ou française. Des corpus étiquetés de ces langues sont mis à votre disposition ainsi que le papier de Selab et Guessoum (Selab &Guessoum, 2015-Building TALAA Arabic Corpus), au cas où vous voudriez avoir une idée sur ce qui peut être fait pour la collecte et prétraitement de corpus.

Les corpus qui sont mis à votre disposition par Mme Selab sont un corpus de la langue Anglaise et un autre de la langue française, accessibles sur Google Drive :
Lien drive :
https://drive.google.com/drive/folders/17frEerXbX_BF0A70827RTW8o_VG9vJki?usp=sharing

Pour le Français, six collections de corpus, Free French Treebank master.zip, une collection de textes étiquetés. Les autres collections ne sont pas étiquetées mais en les analysant elles contiennent des phrases relativement longues qui pourraient être exploitées si besoin est.

Pour l'Anglais : deux collections du Brown corpus adaptées à la méthode propositionnelle ont été sélectionnées.

Il y a un « Read me » pour chaque collection.

Ce qui vous est demandé:

Il s'agit donc de faire ce qui suit :

1. M'envoyer un email pas plus tard que le 9/2 à 23h59 pour choisir si vous allez travailler sur l'Anglais ou le Français. Votre email aura pour sujet « Choix de la langue à traiter pour le mini-projet TALN » et pour contenu :
   Choix 1 : La langue (Anglais ou Français) sur laquelle vous préférez travailler
   Choix 2 : L'autre langue (au cas où le premier choix ne vous est pas accordé, donc (Français ou Anglais)
   Vous indiquerez dans ce même email les noms et prénoms du binôme. (Ce travail doit être fait seulement en binôme. Toute autre forme d'association sera pénalisée.)
   Je vous informerai de l'affectation des langues aux différents binômes pas plus tard que le 12/2.
2. Etudier les 2 papiers susmentionnés, surtout celui de génération automatique de grammaire arabe.
3. Etudier les Corpus de la langue qui vous a été affectée et comprendre le tag set (ensemble d'étiquetage) utilisé.
4. Implémenter l'algorithme suggéré par Selab & Guessoum pour l'induction d'une grammaire propositionnelle.
5. Injecter la grammaire induite dans le module d'analyse syntaxique de Python-NLTK et évaluer l'analyseur ainsi obtenu sur un corpus de test différent de celui à partir duquel vous avez induit la grammaire. Produisez des résultats selon les mesures d'évaluation suggérées dans le travail de Selab & Guessoum.
6. Développement d'une plateforme qui permet ce qui suit:
   - Introduire le nom d'un répertoire (contenant un corpus) ou d'un texte pour l'analyse syntaxique.
   - Afficher le texte à analyser syntaxiquement ou donner à l'utilisateur de sélectionner un texte à afficher et à analyser;
   - Afficher le texte analysé syntaxiquement (ou n'importe quel texte du corpus analysé) chaque phrase sur une ligne (ou plusieurs lignes) suivie par son analyse syntaxique sur une autre ligne (ou plusieurs lignes);
   - Permettre à l'utilisateur d'introduire un nouveau corpus d'apprentissage pour la regénération de la grammaire.
   - Etc. (selon votre imagination/ingéniosité !)
   - Sauvegarder un fichier de toutes les phrases qui n'ont pu être analysées en utilisant le parseur que vous avez produit.

Implémentation et tests de tout l'environnement (incluant le corpus). Votre interface devra être aussi user-friendly que possible (belle et facile à utiliser).

Votre note sera basée sur tout ce que vous pourrez inclure dans votre application et interface sur la base des fonctionnalités requises ainsi que tout ce que vous aurez inclus (en innovant par rapport à la description ci-dessus).

N.B. :

1. Concevez votre solution et implémentez-la exclusivement en Python.
2. Toutes les impressions sur l'interface (et/ou sauvegardes dans des fichiers) doivent être faites dans des états de sorties aussi lisibles et présentables que possible. Corrigez l'orthographe et la grammaire de ces états de sortie !
3. Remettez une application qui installera votre logiciel de façon automatique et facilitera à l'utilisateur de lancer votre application à moins que ce ne soit l'exécution simple d'un programme Python sur Anaconda.

Fin de la description du sujet

Ce que vous devez remettre :

Chaque binôme doit remettre un rapport et un CD.

a. Le rapport imprimé, d'une 10aine de pages au plus (police Times New Roman 11pts) doit présenter tout ce qui a été fait comme étapes. Le rapport doit être bien présenté, très lisible, très clair, présentant l'analyse, la conception, les améliorations apportées, les résultats, les évaluations, etc.
b. Un « manuel d'utilisation » en 2 pages maximum, ajouté en Annexe du rapport, expliquant pas par pas comment faire fonctionner les différentes parties de votre système, à moins qu'elles ne soient évidentes.
c. Une mention claire et précise dans le rapport de qui a fait quoi (ces tâches doivent être nettement distribuées et chacun(e) sera questionné(e) et noté(e) sur sa partie). Vous perdrez des points si ceci n'est pas mentionné. La démo sera une occasion pour questionner chaque membre d'une équipe sur TOUS les aspects du mini-projet.
d. Une copie électronique du rapport en format PDF.
e. Tout le code Python dans un dossier à part nommé « Code Python ». Les noms des fichiers/dossiers doivent être clairs selon votre solution. Il doit être possible d'exécuter votre programme (par le biais de l'interface) et recommencer tout le traitement à zéro (e.g. apprentissage à partir d'un nouveau corpus).
f. La « déclaration de non-plagiat », dûment remplie et signée par chaque membre de l'équipe, attestant que votre rapport ainsi que votre code est entièrement votre travail sauf là où des références explicites et exhaustives sont données. (Le formulaire de cette déclaration vous est envoyé avec cet énoncé). La signature doit être votre signature officielle.

Date de remise du mini-projet (rapport + CD) : **Samedi 06 Mars 2021 à 14h00.**

N.B. :

1. Je n'accepterai aucune remise du projet ou parties du projet par email; vous devrez tout remettre dans ma boite postale au niveau du département (A moins que les mesures sanitaires ne le permettent pas, auquel cas je vous indiquerai la façon de le remettre.).
2. Pour chaque jour de retard, vous perdrez 25% de votre note.
3. **Aucun projet ne sera accepté après le Mardi 09/03/2021 à 15h00.**
4. J'utiliserai l'outil anti-plagiat MOSS (http://theory.stanford.edu/~aiken/moss/) pour vérifier que le code que vous remettrez est bien le vôtre !!! Aucune copie de code de l'Internet, de vos camarades de section, ou autres ne sera tolérée. (Bien sûr, vous pouvez utiliser toute la boite à outils NLTK. Si vous avez des doutes sur tout module trouvé sur Internet, vous pourrez me demander avant la date d'échéance s'il est permis de l'utiliser ou pas.)
5. La note de votre travail prendra en considération la qualité du travail (analyse, programmation, modularité, commentaires, choix des noms de variables et fonctions, etc.), l'interface de l'outil, les fonctionnalités et l'évaluation de la solution, etc.

6. Chaque cas de non-respect des directives énoncées plus haut sera pénalisée.

La note du mini-projet sera divisée comme suit :

a. Votre rapport comptera pour 30% de la note finale.
b. Chaque binôme sera appelé à faire une démo de son système ; cette démo comptera pour 70% de la note. Vous aurez le droit de continuer à améliorer votre système jusqu'au jour de la démo auquel cas vous remettrez un nouveau CD si vous avez modifié votre implémentation. (La note du rapport ne sera pas changée avec ces modifications ; seulement la note de la démo en serait affectée.)
La date de la démo est fixée au Jeudi 11/03/2021. (Ainsi vous pourrez passer les vacances à préparer pour vos EMD et ne plus vous soucier du mini-projet!)

7. La note de chaque membre d'un binôme ne sera pas nécessairement la même, chaque étudiant(e) selon sa contribution et sa compréhension du problème, de la solution, etc. De plus, votre note finale sera relative à celles des autres étudiants de la section ; ceci veut dire que vous devrez considérer les autres binômes comme concurrents. ☹

Bonus :

- Toute innovation intéressante sera généreusement récompensée. Par exemple, la solution que vous proposerez pourrait être juste une amélioration ou une solution encore plus intéressante que celle de Selab & Guessoum. (Vous avez appris beaucoup de choses en SII !!)
  Notez que si vous produisez un travail meilleur que celui de votre enseignant, ce sera une preuve de la réussite de … votre enseignant !! 😉

Remarque finale :
Vous pourriez considérer l'utilisation de Google Colab si vous voulez utiliser des moyens de calcul conséquents.

Bibliographie

1. Selab, E., and Guessoum, A. (2015). A statistical approach for the induction of a grammar of Arabic, In: 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, Morocco, pp. 1-8. doi:10.1109/AICCSA.2015.7507250.
2. Selab, E., and Guessoum, A. (2015). Building TALAA, a Free General and Categorized Arabic Corpus. In Proceedings of the 7th International Conference on Agents and Artificial Intelligence, Lisbon, Portugal, 10–12 January, pages 284-291. DOI: 10.5220/0005352102840291

Bon courage et, surtout, … Enjoy it ! ☺