POLYTECH PARIS-SUD

Et4-Info

Introduction au TAL

Projets

Consignes générales

Le but de ce projet est de montrer que vous pouvez mener un projet jusqu'au bout. Faites donc du travail incrémental, en commençant par les choses les plus simples. La moitié de la note sera consacrée au rapport, la raison étant que, dans le monde de la recherche comme dans celui de l'entreprise, la façon de présenter et de vendre le travail que l'on fait peut parfois être aussi (voire plus) importante que le travail effectué.

Le projet est à réaliser en groupes de deux ou trois. Vous pouvez choisir un des projets décrits ci-dessous ou un projet de votre choix, à condition d'en avoir discuté au prélable avec moi et que je l'aie approuvé. La date de rendu de votre projet est **samedi le 6 mai 2017 à 23h59** (-5pts par jour de retard [1]).

Vous utiliserez un dépôt `git` (`https://github.com`) que vous partagerez entre les membres de votre groupe pour assurer la compatibilité entre les modifications que vous faites. Vous devez m'envoyer un lien vers votre projet (`rachel.bawden@limsi.fr`) dès que vous avez créé le répertoire. La rendu consistera donc en un simple mail qui m'indiquera que le répertoire est prêt pour être noté. La date d'arrivée du mail dans ma boîte mail fera foi. Toute modification du projet après l'envoi du mail sera ignorée (`git` permet cela très facilement). Tout mail de rendu arrivé en retard induira une retenue sur la note finale comme indiqué ci-dessus.

À rendre (dans le répertoire `git`)

1. Un fichier texte nommé `README` contenant les indications nécessaires pour exécuter votre programme. Vous indiquerez notamment la version de Python que vous avez utilisée, les bibliothèques à installer, les options en ligne de commande (si nécessaire), ainsi que des exemples d'utilisation.
2. Le code, écrit en Python (2 ou 3), séparé en fichiers et en fonctions de façon logique et commenté de façon appropriée. La lisibilité du code sera un critère dans la notation (y compris la présence de commentaires précis, concis, clairs et bien orthographiés).
3. Un rapport, écrit en LaTeX et compilé en un PDF de 5 à 7 pages qui décrit votre travail (police et marges de taille raisonnable). Vous devez y décrire l'objectif de votre projet, la motivation derrière ce que vous avec fait, la méthode que vous avez utilisée, à quel point cela fonctionne, et les limitations de votre programme (c'est-à-dire les cas qui restent à traiter ou qui fonctionnent mal). Vous devez également inclure une petite section qui décrit de façon brève la contribution de chaque membre du groupe. Votre rapport peut être écrit en anglais ou en français. Le rapport ne doit pas inclure de code source.

Notation

Le barème ci-dessous n'est donné qu'à titre indicatif et pourra être modifié ultérieurement. Vous pouvez néanmois l'utiliser comme guide général.

1. **Code et README : 10 points**
   — 3 pts pour la lisibilité et clarté de votre code

1. Un jour de retard est compté à partir d'une minute après minuit.

— 7 pts pour le bon fonctionnement du programme et ses fonctionnalités, telles que décrites dans le rapport.

2. **Rapport : 10 points**
   — 4 pts pour l'orthographe, la grammaire et la bonne formulation (style professionnel) de votre rapport. Donc relisez-vous bien et prenez du plaisir à « vendre » votre travail !
   — 6 pts pour le contenu du rapport : est-ce que vous avez bien présenté la motivation du projet, ce que votre programme fait et ce qui resterait à faire dans l'avenir ?

Conseils

— Committez souvent votre travail vers le répertoire `git` et communiquez bien entre vous. Ceci évitera les conflits entre versions différentes du code. Votre contribution au projet pourra être vérifiée sur `github`, en regardant les commits que vous avez faits. Les notes peuvent être différentes pour des membres d'une même équipe si le volume travail de tel ou tel membre de l'équipe est clairement différent de celui des autres. Une bonne manière d'assurer la participation de tous est de bien diviser le travail en tâches et de bien répartir ces tâches.

— Mettez-vous d'accord sur les espaces et les tabulations pour l'indentation de votre code. Par exemple, décidez que toute tabulation sera remplacée automatiquement (en modifiant les paramètres de votre éditeur de texte) par 4 espaces, ou soyez sûrs que tout le monde utilise les tabulations.

— Le rapport est une opportunité pour vendre votre travail. Soyez-en fiers ! Si quelque chose n'est pas mentionné dans le rapport, il ne sera pas vu et donc pas noté. Mettez en avant ce que votre programme gère bien. Les limitations de votre programme sont importantes à mentionner, mais il est aussi important de les présenter comme des points d'amélioration plutôt que comme des failles. Si vous savez comment vous auriez souhaité résoudre ces limitations si vous aviez eu plus de temps, mentionnez-le comme un des choses prévues pour l'avenir.

1 Sujets

1.1 Chatbot/assistant personnel

Un chatbot/assistant personnel est un agent programmé pour entretenir un dialogue avec un utilisateur. À l'origine, les chatbots étaient destinés à une tâche spécifique (p. ex. réservation de billets de train, assistance pour les achats dans un magasin, psychologue, etc.), mais de nos jours, il existe aussi des agents destinés à la conversation en général, une tâche bien plus difficile.

Enjeux

- Le but d'un chatbot est de créer une situation de dialogue entre un utilisateur et l'agent. Son langage doit donc essayer de refléter le mieux possible les énoncés attendus par un humain.
- Veillez à ce que les énoncés que vous produisez soient grammaticalement bien formés. Ceci peut être parfois difficile, mais fera une grande différence pour la qualité de votre chatbot !
- Des réponses plus précises (qui prennent en compte les énoncés de l'utilisateur par exemple) augmenteront la crédibilité de votre chatbot. Un système simple qui répère quelques mots clés peut être assez efficace.

À faire

- Vous choisirez un domaine de compétence pour votre chatbot. Quel est est son but ? Vous pouvez par exemple créer un chatbot pour interroger un site de réservation de billets de train, pour donner des directions, ou pour entretenir une conversation. Si vous choisissez de développer un agent conversationnel général, vous devrez probablement mettre en avant un ou deux aspects de la conversation que votre chatbot maîtrise particulièrement bien.
- Vous pouvez utiliser la méthode que vous souhaitez pour construire votre chatbot : heuristiques, motifs, mots clés, grammaires (CFGs avec NLTK), réponses génériques, etc.
- les énoncés produits par votre chatbot devront être grammaticalement bien formés. Faites attention à l'accord, aux temps des verbes, etc.
- Un dialogue est formé d'un locuteur (celui qui parle à un instant donné) mais aussi d'un interlocuteur (celui qui écoute). Pour créer un dialogue plus réaliste, pensez à inclure des énoncés qui montrent que l'agent prend en compte ce que dit l'utilisateur. Les « backchannels » sont des signaux verbaux ou non verbaux qui sont les réponses de l'interlocuteur (hochements de têtes pour les agents qui ont en ont une, des marqueurs de compréhension tels que « uh huh », « oui », des reprises de l'énoncé précédent en cas de manque de compréhension etc.)
- Mettez en avant ce que votre agent arrive à faire, mais incluez dans votre rapport une section qui discute des limitations de votre chatbot : y a-t-il des choses qu'il ne peut pas gérer ? Quelle sorte de solutions proposeriez-vous pour l'avenir ?

1.2 E-opinion : classification de reviews

L'e-opinion (analyse de sentiments) est un domaine très utile pour les entreprises, surtout pour celles impliquées dans la vente de produits ou de services. Les retours des clients sont utiles pour pouvoir s'adapter aux marchés ou d'identifier des points d'amélioration.

Les reviews clients se trouvent en grand nombre sur internet (cf. Amazon, RottenTomatoes, etc.). Elles comprennent souvent une note (par exemple sur 5), ainsi qu'un avis en language naturel, dans lequel on peut trouver des informations exploitables si on connaît le TAL !

Un moyen simple de s'attaquer au problème est de classifier les reviews comme étant positives ou négatives (voire sur une échelle plus fine). Vous pouvez également essayer d'identifier automatiquement certains aspects des reviews qui peuvent être pertinents pour une entreprise, par exemple en identifiant toutes les phrases négatives/positives.

Enjeux

- Vous aurez probablement un certain nombre de mots qui relèvent du domaine spécifique que vous aurez choisi. Vous pourrez donc constituer un petit lexique contenant des mots spécialisés du domaine, par exemple des aspects du produit qui est en vente.
- Détecter les phrases positives et négatives n'est pas trivial. Par exemple, l'existence d'un mot négatif dans une phrase ne signifie pas forcément que la phrase est négative. Par exemple : « Ce n'est pas du tout un **mauvais** portable ». Vous pouvez également rencontrer des comparaisons avec d'autres produits : « Le portable de la marque concurrente est bien meilleur que ce portable ». Donc une approche « sac de mots » ne sera pas forcément la méthode la plus adaptée. Mais vous pouvez toujours tester cette méthode pour l'utiliser comme point de comparaison avec d'autres méthodes plus complexes que vous essayerez.

À faire

- Choisissez un domaine sur lequel vous voulez appliquer votre analyseur de sentiments (p. ex. des marques de téléphones portables, machines à laver, films, livres, etc.) et définissez l'objectif de votre étude. Imaginez-vous dans la position d'une entreprise qui fournit une étude de marché pour un client. Votre tâche est de vendre votre analyseur de sentiments et de leur montrer comment cela marche et à quel point ça marche bien !
- Trouvez le corpus sur lequel vous allez travailler. Vous pouvez regarder sur Amazon ou sur les sites de reviews de films ou de livres. Votre corpus sera constitué d'un ensemble de documents (chaque document étant une critique de produit) contenant du texte en langage naturel. Parfois les reviews sont aussi associées à des notes (par exemple un nombre d'étoiles sur 5). Ces notes pourraient être utilisées comme moyen de vérifier que votre analyseur fonctionne bien.
- Vous utiliserez la méthode (ou les méthodes) que vous voulez pour effectuer cette tâche (probabilités, expressions régulières, grammaires CFG (CFGs avec NLTK), règles écrites à la main, etc.)

1.3 Détection d'entités nommées

Une des tâches de pré-traitement les plus utiles pour de nombreuses applications de TAL est la détection d'entités nommées. Une entité nommée est une séquence de mots référentielle qui, dans la plupart des cas, représente une entité référentielle dans le « vrai » monde, tel que le nom d'une personne, d'un lieu, d'une organisation, une date, une mesure, etc.

Enjeux

- L'ambiguïté : une expression peut souvent faire référence à plusieurs types d'entités en fonction de son contexte d'utilisation, ou simplement à quelque chose qui n'est pas une entité nommée. Par exemple « Pierre » est un nom propre ou un nom commun, « Paris » peut être une ville en France, une ville aux États-Unis, un prénom ou le pluriel de « pari ». Pour désambiguïser, il faut regarder le contexte des mots, leur mise en majuscules, etc.
- Des entités nommées peuvent bien sûr comporter plusieurs tokens (« Ministre de la Défense », « Homer Simpson », « Le Monde »). Il faudra pouvoir donc identifier ces entités aussi. Il faudra faire des choix : Est-ce que dans « le Maire de Paris » vous voulez identifier « Paris » comme étant un endroit, ou est-ce que vous identifez seulement la personne « Maire de Paris » ?
- Certaines entités peuvent faire partie d'amalgames : par exemple dans « Un journaliste du Monde est venu... », le journal « Le Monde » n'a pas exactement la même forme, donc on pourrait décider de segmenter les amalgames en plusieurs tokens (« de Le »).

Entités à reconnaître

Vous pouvez reconnaître des entités au niveau des caractères (chiffres et mesures (poids, distances, prix), adresses-mails, smileys, dates en chiffres (ex : 04/02/2016)) et des entités lexicales (dates en toutes lettres (ex : 4 février 2016), nombres en toutes lettres (ex : mille, quatre-vingt, ...), personnes, lieux, organisations, marques, etc.).

Annotations à fournir

Vous choisirez d'abord un corpus sur lequel vous allez travailler. Ce corpus doit être grand (au moins plusieurs milliers de phrases), et plus d'entités nommées y apparaissant, mieux c'est. Vous pouvez par exemple récupérer un nombre important d'articles de journaux, ou du texte librement disponible sur internet (ex : `https://www.gutenberg.org`). Il existe également des corpus annotés en entités nommées que vous pouvez trouver sur internet. Dans ce cas, le but serait de pouvoir reproduire les annotations « gold » fournies dans le corpus. Vous pouvez choisir de travailler sur des textes en anglais ou en français.

La détection d'entités nommées est une tâche d'annotation. Cela veut dire que votre programme devrait être capable de détecter les entités qui s'y trouvent. Vous devez donc diviser votre corpus en plusieurs sous-corpus, par exemple un sous-corpus d'entraînement et un sous-corpus de test, ce dernier étant utilisé pour vérifier à quel point votre programme est généralisable à de nouvelles données.

Vous déciderez du format de vos annotations. Vous pouvez décider d'annoter directement votre texte, ou de faire un document d'annotation à part, dans lequel se trouvent la liste des entités trouvées avec leur type et leur endroit dans le texte (par exemple l'indice du caractère de départ et l'indice du caractère de fin de l'entité).

Conseils

- Vous pouvez utiliser la méthode que vous voulez pour faire cette tâche. Une approche par heuristiques est totalement légitime. Vous pouvez vous servir de listes de noms trouvées sur internet, d'expressions régulières, de grammaires CFG (voir la documentation avec NLTK), de règles écrites à la main, etc.
- Commencez toujours par les cas les plus simples. Il y a beaucoup d'exceptions à gérer, mais aussi beaucoup de cas assez communs. Il est très important de pouvoir gérer les cas simples avant de passer aux cas plus complexes.