Object search

Gautier DI FOLCO

Janvier 2014

Table des matières

Résumé

Le but est de voir s'il est possible, à partir de n'importe quel site marchand, via *machine learning* d'en extraire les informations principales.

1 Méthodologie

Nous somme partie sur une approche CRF et nous avons pour cela pris chaque noeud d'un document HTML (en ayant filtré un grand nombre de noeuds non-suceptible de nous intéresser pour les informations que nous cherchons, afin de limiter le nombre de noeuds inintéressant, qui rendraient l'apprentissage plus long) et nous avons établi une liste de *feature*s.

Nous avons manuellement marqué (via l'ajout d'un attribut *data-tess-label*) une partie (10 pages) de notre jeu de données (il comporte 50 pages par site, ces sites étant amazon.fr carrefour, ldlc, fnac et rueducommerce) afin de nous en servir comme base d'apprentissage pour notre logiciel de CRF (*CRFSuite*).

1.1 Features extraites

— Le nombre de parents ayant une classe contenant le mot "prod"
— Le nombre d'ancêtres ayant une classe contenant le mot "prod"
— Le nombre d'ancêtres ayant une classe contenant le mot "desc"
— Le nombre de classes ayant "prod" dans son nom pour le noued courant
— Le nombre de classes ayant "desc" dans son nom pour le noued courant
— Le nombre d'occurences de "prod" dans le texte du noued courant

— Le nombre d'occurences dun symbole de monnaie dans le texte du noued courant
— Le nombre d'occurences de "desc" dans le texte du noued courant
— Le nom du noeud courant
— Le nom du noeud parent
— Les propriétés du noeud
— La classe courante
— La classe du noeud parent
— Le nombre d'ancêtre
— Le nombre de déscendants
— Le noeud courant ou un de ses ancêtre est-il un noeud de type h1, h2 ou h3

1.2 Données recherchées

Nous avons tenté de chercher la description, le titre et le prix de chaque page.

Dans cet objectif, notre convertisseur de HTML en *dataset CRFSuite* va, pour chaque noeuds non-filtré de la page, chercher si ce noeud a l'attribut *data-tess-label* si c'est le cas, le type de l'enregistrement aura cette valeur, si non il aura comme type "other".

1.3 Méthodologie

Dans un premier temps nous récupérons un jeu de 50 pages sur 5 sites, nous en marquons manuellement 10.

Nous convertissons ensuite ces deux jeux (l'original de 50 pages et le marqué de 10 pages) en format CRFSuite.

Puis nous lançons la procédure d'apprentissage, une fois les modèles générés nous les appliquons sur les pages non-marquées au format CRFSuite.

Ensuite nous convertissons le résultat en page HTML.

Nous récupérons de plus des statistiques sur les phases d'apprentissage et de marquage.

1.3.1 Première tentative : inter-site

Lors de l'apprentissage nous avons générer les 120 combinaisons possibles de sites (afin de voir si l'ordre des jeux de données influence l'apprentissage) puis nous avons retirer un jeu (le dernier site de la liste) et nous avons fait un apprentissage sur les jeux restants.

Puis nous appliquons chaque modèles aux pages du jeu restant.

Nos observations sont les suivantes :

— L'ordre des jeux de données à une influence sur le temps d'apprentissage
— L'apprentissage est très long (entre 20h et 6 jours)
— Les modèles générés sont inefficaces (aucun marquage n'a remonté une information recherchée)

Nous en concluons qu'il faut, avant de poursuivre les investigations plus loin, valider que CRFSuite est fonctionnel.

1.3.2 Deuxième tentative : site par site

Les pages de chaque site étant générés dynamiquement à partir d'un gabarit fixe CRFSuite devrait pouvoir s'y retrouver plus facilement si chaque site à un apprentissage et un marquage isolé.

Malheureusement ce n'est pas le cas, les temps d'apprentissages sont encore long (de quelques heures à quelques jours) et les marquages sont encore inefficaces.

Nous en déduisons donc que soit le nombre de noeuds "other" est trop important par rapport aux noeuds intéressants, soit les features sont mal choisies/insuffisantes.