Rapport du projet - EatAWeek

Vincent Segonne, De Zhao (Team Li)
Université Paris Diderot

Table de Matières

1 - Introduction

2 - Rappel du projet

3 - Théorie et Méthodologie

4 - Technique et Implémentation

    k-means

    interface graphique

5 - Résultat et Évaluation

    métriques

6 - Discussion

7 - Conclusion

8 - Equipe

Introduction

Nous présentons dans ce rapport le travail que nous avons effectué dans le cadre du projet du cours de Fouille de Données (l'année 2016 - 2017). Le projet, nommé *EatAWeek*, est la réalisation d'un système de recommandation basé sur le contenu. Le coeur du système est réalisé avec l'algorithme d'apprentissage non supervisé (clustering) K-means.

Même si à première vue, l'implémentation de l'algorithme semble assez simple, notamment grâce au développement de module comme SkLearn, nous verrons qu'elle nous a permis d'approcher un certain nombre de problématiques très importantes dans le domaine du machine learning. Par exemple la gestion des données tant sur la quantité que la qualité demeure cruciale, le choix des attributs et plus généralement la représentation des données est également un point essentiel.

Par ailleurs, ce projet fut aussi l'occasion pour nous d'un nouvel entraînement au travail de groupe. Dans le domaine où nous travaillons, savoir collaborer avec d'autres personnes sur un même projet est essentiel et de ce point de vue, nous avons rencontré peu de difficulté. Nous appris grâce à ce projet à nous servir de Git ce qui pour nous en tant que futurs linguistes informaticiens est très important.

Nous allons dans la prochaine section commencer par vous présenter l'idée générale du projet, nous y rappellerons ses objectifs ainsi que l'intérêt qu'à celui-ci dans le cadre du cours.

La troisième partie intitulée ***Théorie et Méthodologie***, s'articulera autour de 2 axes principaux. Tout d'abord nous ferons un tour sur l'état de l'art en ce qui concerne l'apprentissage supervisé et les systèmes de recommandation. Nous verrons pourquoi nous avons décidé de nous tourner vers des méthodes non supervisées. Puis nous vous présenterons les ressources que nous avons utilisées pour implémenter l'algorithme.

Avec cette troisième partie, nous aurons ainsi rappelé les bases et fait un tour d'horizon du projet de manière globale. Ce qui nous permettra par la suite de nous concentrer sur les détails techniques et sur l'implémentation

mise en oeuvre. Pour cela, nous débuterons par donner un aperçu de l'architecture du programme. Le but n'est pas d'expliquer fonction par fonction comment le programme marche, mais plutôt d'expliquer quels sont les différents modules et comment ceux-ci interagissent entre eux. Une fois que l'architecture du programme aura été présentée, nous ferons par la suite l'état de nos choix d'***Implémentations*** expliquant la façon avec laquelle nous avons décidé de stocker les données et la façon dont nous avons réalisé l'interface graphique.

Dans la dernière partie, nous aborderons un champs essentiel à tout système travaillant avec du machine learning. Nous ferons le tours des méthodes utilisables pour mesurer les sytèmes de recommandations, et nous expliquerons pourquoi nous n'avons pour notre part pas réussi à évaluer le programme.

Enfin, nous clôturons le rapport par évoquer quelques idées qui pourraient améliorer le système.

Rappel du projet

Le projet EatAWeek est un projet qui vise à automatiser la tâche pénible de faire les commissions. Le but final étant de ne plus à se préoccuper de ce que nous allons manger dans la semaine au moment de faire les courses. Pour cela, l'idée principale du projet est de pouvoir générer automatiquement un certain nombre de repas pour la semaine avec la liste des ingrédients à acheter. Nous précisons que ce type de système existe déjà sur le web. L'intérêt qu'à ce projet dans le cadre de ce cours est qu'en plus de générer automatiquement des recettes , le système doit pouvoir apprendre à connaître les goûts de l'utilisateur afin de pouvoir au mieux choisir des plats qui lui plaisent. L'apprentissage automatique est donc au coeurs de ce projet. Il s'agit en fait simplement de monter un moteur de recommandation de recettes. Pour cela nous nous sommes largement inspiré de ce que nous avons vu en cours et avons implémenté les méthodes utiles pour les moteurs de recommandations.

Théorie et méthodologie

Comme ce que nous avons vu en cours, les méthodes d'apprentissage supervisé visent à apprendre et construire un modèle à partir des grosses volumes de données d'entraînement pour qu'avec ce modèle, la machine soit capable de classifier ou bien de prédire des nouvelles instances. Un système de recommandation s

Les méthodes d'apprentissage automatique visent à apprendre à partir de données un modèle qui permettent de faire des prédictions. On distingue deux principaux algorithmes, les algorithmes supervisés, comme le Perceptron ou le HMM et les algorithmes non-supervisées comme le clustering. La principale différence entre ces deux méthodes repose sur la disponibilité et le volume des données auxquelles nous avons accès. En effet, pour qu'un système entrainé avec des méthodes supervisée soit efficace, il est nécessaire de disposer d'un volume important de données. Lorsque ce n'est pas le cas, on se tourne alors en général vers des méthodes dites non-supervisées. La première étape consiste donc à se poser la question des données que nous allons utiliser. Dans notre cas, il s'agit de recettes de cuisine, comme il était bien entendu hors de question de rentrer des recettes à la main, nous avons opter pour un web crawler afin de récupérer les recettes présentes sur le site marmiton.org. Au total, c'est plus de 3000 recettes que nous avons récupérées. Il faut noter que cette tâche n'a pas été évidente pour nous, tout d'abord car c'était la première fois que nous implémentions une technique de webcrawling, nous avons dû apprendre à coder en php, et puis également parceque les données (comme la plupart des données du web) sont parfois de mauvais qualité ce qui rend le parsing difficile. Cela nous a freiné dans notre avancement de projet car sans les données il est assez difficile de faire des tests avec le système.

Une fois que nous avons récupéré les données, nous avons pu commencer à implémenter les méthodes vu en cours pour mettre en marche un moteur de

recommandation. La première tâche consiste à représenter les données de manière numérique. C'est à dire comment transformer objet réel comme les recettes en objets numériques sur lesquels il est possible de réaliser des calculs ? La puissance du machine learning repose sur la représentation de ces documents (les objets sur lesquels nous travaillons) en vecteur. Ainsi, chaque dimension du vecteur représente une caractéristique de l'objet. Dans ce cas comment représenter une recette ? Quelles sont ses caractéristiques ?

Nous avons implémenté une technique très utilisée pour représenter les documents textuels, la représentation par les termes du document, plus généralement appelée représentation TF-IDF. Chaque dimension du vecteur correspond alors au vocabulaire de l'ensemble du corpus (l'ensemble des recettes) et via des calcules de fréquence et de pondération, les mots saillants d'un document sont mis en valeur. L'avantage que présente les vecteurs est qu'ils permettent notamment de faire des calculs de similarité et donc par conséquent de comparer les documents entre eux. Nous avons donc appliqué un TF-IDF sur trois partie des recettes, le titre, les ingrédients et les instructions de la recette.

Nous avons également ajouté des features (traits) qui nous paraissaient important et qui pourrait également bien représenter une recette. En effet nous avons ajouté des traites pour la difficulté de la recette (Très facile, Facile, Moyennement Difficile et Difficile), le coût (Bon marché, Cher) et enfin les différents temps de préparation et cuisson que nous avons organisé en échelle (0-20, 20-45, +45 min) . On espère avec cela renforcer un peu plus la similitude entre documents et à fortiori pour voir mieux percevoir les préférences de l'utilisateur. Enfin, nous avons pondéré ces dimensions par un score, qui est en fait le nombre de fois où la recette a été choisie dans le menu de l'utilisateur divisé par le nombre de recette.

Maintenant que nous savons comment représenter les recettes que nous avons récupérées, nous pouvons commencer à parler du coeur du programme, l'algorithme qui permet au système de faire des recommandations de plats.

Pour mettre au point le moteur de recommandations, nous avons fidèlement utilisé les méthodes qui nous avons vu en cours. Tout d'abord il nous faut reconnaître les goûts de l'utilisateur afin de pouvoir au mieux lui présenter des recettes. Pour cela il existe deux principales méthodes : la modélisation par utilisateurs et la modélisation par contenu. Dans le cas de la modélisation par utilisateurs, le système va tenter de comparer les utilisateurs entre eux. En effet, cette méthode repose sur l'idée que deux utilisateurs qui se ressemblent vont avoir tendance à aimer les mêmes choses. Dans le cadre de ce projet il est difficilement imaginable d'avoir recourt à cette technique car nous n'avons pas comme Facebook, Google ou Amazon les moyens d'avoir accès à des milliers , millions voire milliards d'utilisateurs. Il nous a fallu donc nécessaire envisager une modélisation par contenu. Cette méthode ne nécessite pas de connaître plusisueurs utilisateurs car elle ne se base que ce par quoi l'utilisateur du système est intéressé. En effet, on imagine assez bien qu'une personne ayant une préférence pour un certain type d'objet aime également les objets qui lui ressemblent. La modélisation du contenu consiste donc à modéliser un profil utilisateur et puis à comparer les nouveaux objets (dans notre cas les recettes) avec profil pour trouver ceux qui sont les plus susceptibles de lui plaire.

Alors arrivent deux principales difficultés. Tout d'abord comment représenter le profil utilisateur ? Une première technique consiste à calculer le vecteur moyen de ses goûts. Cette technique bien que très simple à implémenter a un inconvénient important, est-ce que ce vecteur reste pertinent si l'utilisateur aime des choses très différentes ? Avec les recettes c'est particulièrement le cas, il est naturel d'aimer des plats tout à fait différent. Pour parer à ce problème et pour éviter d'avoir un profil trop "ambiguë" nous avons décidé d'implémenter une autre méthode, celle qui consiste non pas à modéliser le profil par un vecteur moyen mais par un clustering, c'est à dire plusieurs regroupements de recettes qui sont similaires. Ainsi on s'assure de comparer les nouveaux objets à des recettes comparable. La méthode de clustering ue nous avonNous avons de plus implémenté un petit grid search qui permet de trouver le meilleur k (hyper paramètre de clustering) afin d'optimiser au mieux le profil utilisateur. Pour cela,

nous avons itéré sur au plus 20 clusters et avons calculé la somme des meilleures similarité cosinus de chaque recette. L'itération dont le score est le plus haut et le k à utiliser pour le cluster.

La deuxième difficulté est communément appelée le problème du cold-start. C'est à dire qu'à la création du profile utilisateur, le système n'as pas accès à ses données. Nous avons donc implémenté une phase d'initialisation de profil. Pendant cette phase, des repas aléatoires sont présentés à l'utilisateurs et c'est à lui d'aimer via un bouton si une recette lui plait (ou non!).
Une fois que son profil est crée, la phase de recommandation se lance pendant laquelle le grid search trouve le meilleur nombre de cluster possible. Une fois que les centroides ont été calculés, le système calcule la similarité cosinus avec toutes les recettes de la base de données et les trie dans l'ordre décroissant. Seules les 10 meilleurs recettes sont retenues et proposée à l'utilisateur.

Techniques et Implémentation

Nous allons maintenant quitter la partie théorique pour entrer plus en détails dans le fonctionnement du programme. Dans cette partie nous présenterons d'abord l'architecture globale du projet, nous verrons les différents modules PHP et Python que nous avons utilisé et comment ils permettent au programme de fonctionner.

Cette description recouvre l'intégralité du projet, de la récupération des données à la proposition des recettes.

Ensuite nous passerons à la question des choix d'implémentation. Ce sera notamment l'occasion de parler des choix que nous avons fait concernant le stockage et la gestion des données, une des principales difficultés à laquelle nous avons confrontés.

Avec ce passage en revue des différents éléments qui constituent l'ensemble du programme nous espérons pourvoir donner une vision globale de celui-ci.

Le projet a été divisé en trois tâches principales : la récupération des données, l'implémentation de l'algorithme de classification et la réalisation de l'interface graphique.

Nous allons commencer par expliquer comment nous avons récupérer les données. Pour avoir une bonne couverture des recettes marmiton, il nous a fallu crawler en fonction de lettre latine passer en argument, cela réduit d'un coté le temps utilisé et de l'autre coté le mémoire consommé. Après avoir crawlé les données et les avoir écrites dans un fichier .txt. Nous avons utilisé le module sqlite3 en Python pour constituer une base de données. Cette étape de transformation réduit beaucoup le temps de chargement des données et les rend plus facile à traiter.

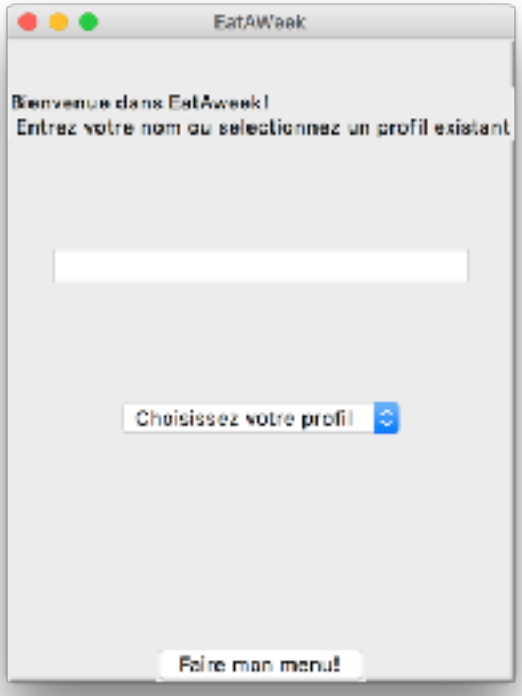
La vectorisation des données s'est faire de manière hybride, la partie TFIDF a été réalisée via le module Sklearn.feature_extraction.text et les autres dimensions ont été codées à la main. Cette partie se trouve dans le fichier utils.py.

En ce qui concerne le coeur du programme c'est à dire l'implémentation du moteur de recommandation, nous avons utilisé les modules fournis par Sklearn. L'implementation de l'algorithme de clustering se trouve dans le fichier RecipeAdviser.py, nous avons utilisé deux modules principaux : Sklear.cluster et plus particulièrement le Kmeans et puis Sklearn.metrics pour obtenir la similarite cosinus nécessaire au clustering. Le grid search pour estimer le meilleur k a été implémenté à la main et se trouve dans le fichier utils.py, il s'agit de la fonction get_best_k().

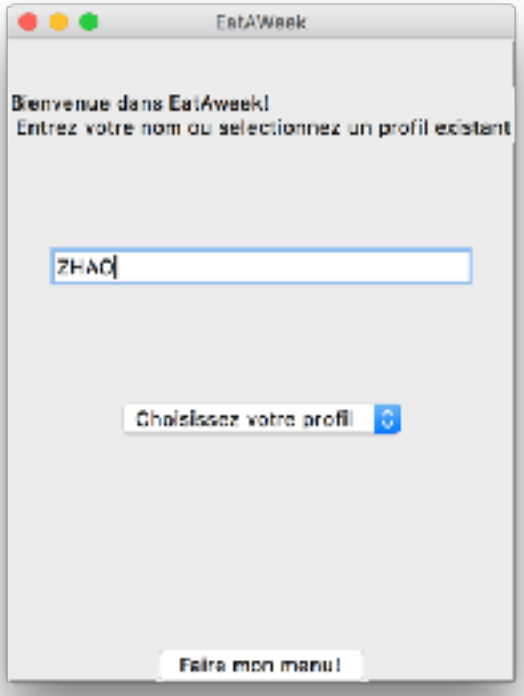
Enfin, pour ce qui est de l'interface graphique, nous avons utilisé le module Tkinter.

Le programme se lance avec Python3 de la manière suivante:

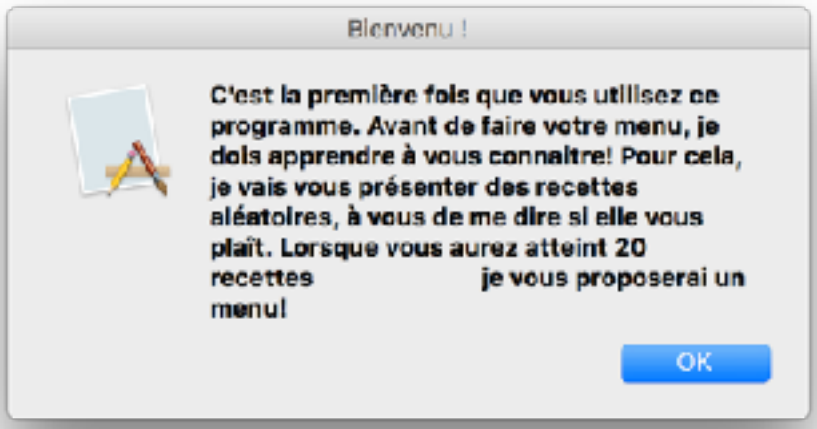
```
> python3 main.py
```



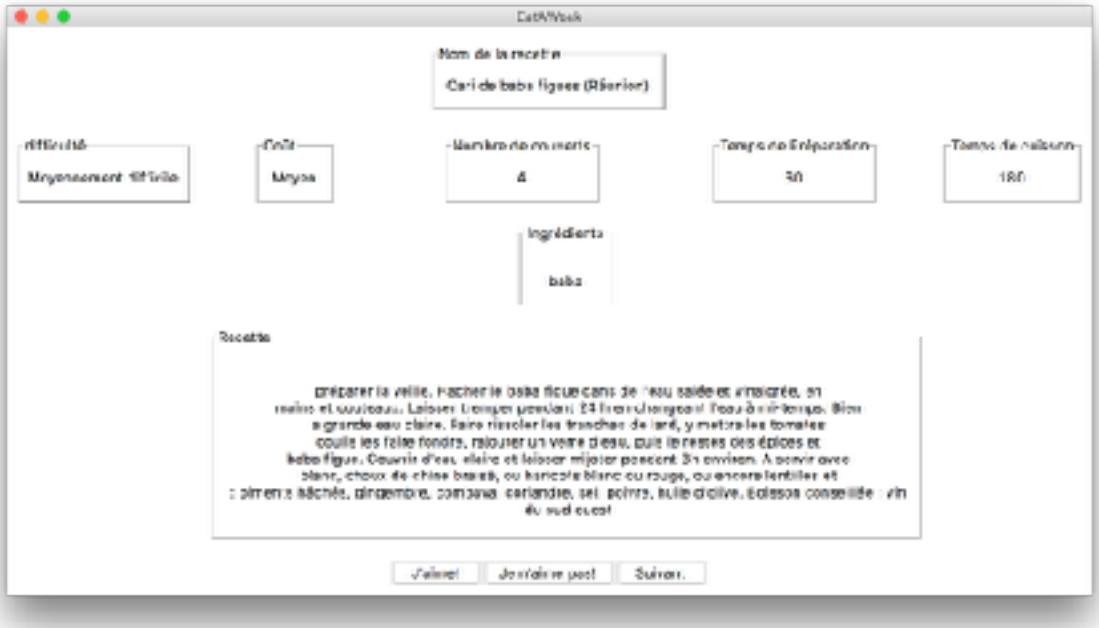
Voici la fenêtre d'accueil de notre programme, il vous demande d'entrer un nom ou choisir un profile déjà enregistré.



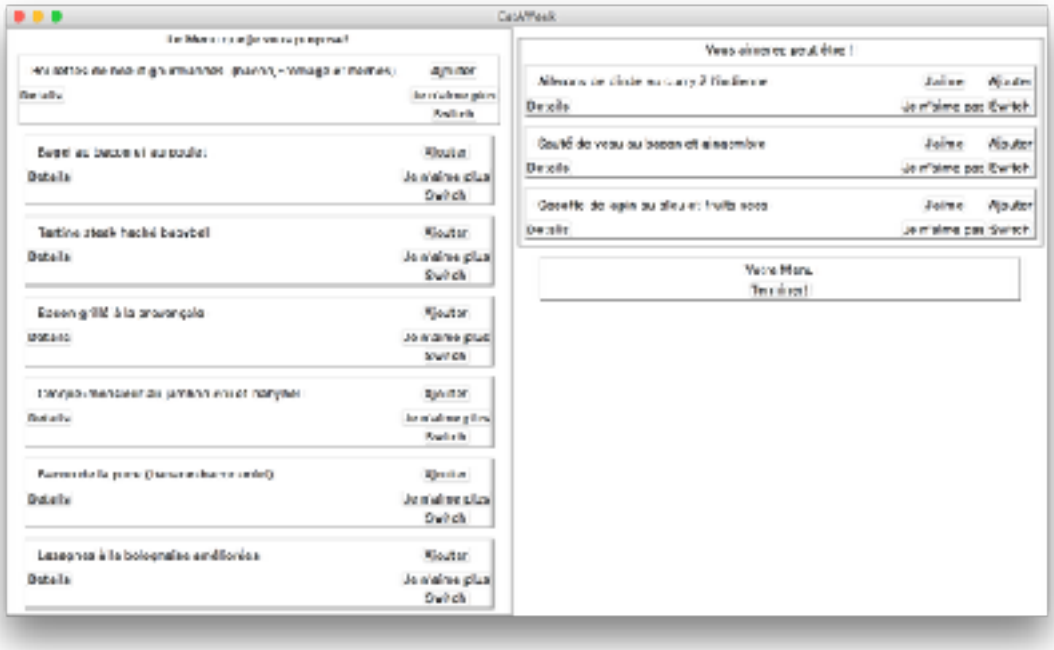
Nous entrons un nom et clique sur 'Faire mon menu!'.



Là, nous voyons un signal d'invitation.



Nous tombons sur une représentation de la recette.



Dans cette fenêtre, nous pouvons liker une recette, l'ajouter dans le menu ou bien la rejeter.

Résultats et Évaluation

Par manque de temps, nous n'avons malheureusement pas pu évaluer notre système comme nous le voulions, cependant nous tenions tout de même à évoquer les recherches que nous avons faites et que nous avions à l'origine prévue de réaliser.

Au premier abord, la méthode d'évaluation d'un système de recommandation basé sur le contenu n'était pas évidente pour nous. Parce que la tâche de recommandation n'est pas comme la plupart des tâches en Traitement Automatique des Langues où nous disposons dès le début des corpus d'entraînement et parallèlement des corpus de test. Nous pouvons ainsi comparer les résultat de prédiction/classification avec le GOLD pour avoir une vision globale du fonctionnement de notre parseur/classifieur. Par exemple, un simple triplet <précision,rappel,f-score>.

Pourtant, après quelques recherches, nous avons remarqué qu'il existe au moins trois métriques d'évaluation pour un système de recommandation basé sur le contenu. La première est la métrique MAE (Mean Absolute Error). Elle vise à calculer la déviation moyenne entre les scores prédits et les scores fournis par humain. Dans notre cas, ce sera de calculer la somme de différences entre le score prédit par machine et le score fourni par humain pour une même recette. Tant que la valeur MAE baisse, l'accuracy du système augmente.

$$MAE = \frac{1}{\#\,recette} \sum_{u,i} \left| score\_p_{u,i} - score\_h_{u,i} \right|$$

soit i une recette, u un utilisateur, *score_p* le score prédit pour la recette i, *score_h* le score fourni par u

La deuxième métrique s'appelle RMSE (Root Mean Square Error) qui mets plus l'accent sur des erreurs absolues.

$$RMSE = \sqrt{\frac{1}{n}\sum_{u,i}\left(score\_p_{u,i} - score\_h_{u,i}\right)^2}$$

Conclusion

Pour conclure ce rapport, nous voulions simplement rappeler que ce projet a été très enrichissant pour nous. Nous avons pu nous améliorer sur le travail de groupe, nous avons implémenter de manière efficace via les modules sklearn des techniques de machine learning et avons expérimenté de nouveaux domaine comme le webcrawling. Ce projet était de plus assez ludique puisqu'il donne une réelle fin en soit même si le programme présenté n'est encore qu'une ébauche il pourra sans doute à terme être utilisé dans la vie de tous les jours. À ce sujet nous tenions à évoquer quelques pistes pour l'amélioration du système que nous n'avons malheureusement pas eu le temps de tester et d'implémenter. Tout d'abord, il semble évident que des filtres seraient très utiles autant sur le plan pratique que sur le plan théorique en effet. On pense par exemple aux personnes végétariennes, ce serait bien plus pratique pour eux d'écarter d'office les plats avec de la viande et cela rendrait probablement le moteur de recommandation plus efficace. Un autre point qu'il serait très intéressant de tester serait l'utilisation des "dislike", c'est à dire plats qui n'ont pas été aimés. Car si le fait d'aimer une recette constitue une richesse en terme d'informations, le fait de ne pas aimer peut également un indicateur sur le profil de l'utilisateur. Pourquoi pas envisager un double trie, ou faire la différence entre la meilleurs similarité entre les objets avec le profil "like" et la pire similarité avec le profil "dislike".

Modules utilisés dans ce projet

Marmiton Crawler
https://github.com/sylouuu/marmiton-crawler.git

PHP HTML Parser
https://github.com/paquettg/php-html-parser.git

PHP Simple HTML DOM Parser
https://sourceforge.net/projects/simplehtmldom/
http://simplehtmldom.sourceforge.net/

SK-learn