Projet N°3 : PageRank - Pondération de Pages web[1]

1. Introduction

Écrivez un programme d'IA pour pondérer des pages web par ordre d'importance.

A rendre le : **15/12/2020**

2. Contexte

Lorsque les moteurs de recherche comme Google affichent des résultats de recherche, ils le font en plaçant plus haut les pages "importantes" et de meilleure qualité dans les résultats de recherche que les pages moins importantes. Mais comment le moteur de recherche sait-il quelles pages sont plus importantes que les autres ?

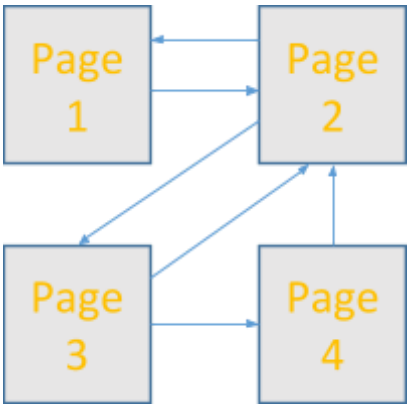
Une heuristique pourrait être qu'une page "importante" est une page vers laquelle de nombreuses autres pages renvoient, car il est raisonnable d'imaginer que plus de sites renvoient vers une page web de meilleure qualité qu'une page web de moindre qualité. Nous pourrions donc imaginer un système dans lequel chaque page se verrait attribuer un rang en fonction du nombre de liens entrants qu'elle a en provenance d'autres pages, et un rang plus élevé signalerait une plus grande importance.

Mais cette définition n'est pas parfaite : si quelqu'un veut faire paraître sa page plus importante, alors, dans le cadre de ce système, il pourrait simplement créer de nombreuses autres pages qui renvoient à la page souhaitée pour gonfler artificiellement son rang.

C'est pour cette raison que l'algorithme du **PageRank** a été créé par les co-fondateurs de Google (dont Larry Page, qui a donné son nom à l'algorithme). Dans l'algorithme du PageRank, un site web est plus important s'il est lié à d'autres sites web importants, et les liens de sites web moins importants ont un poids moindre. Cette définition semble un peu circulaire, mais il s'avère qu'il existe de multiples stratégies pour faire ces classements.

3. Modèle du surfeur aléatoire

Une façon de penser le PageRank est le modèle du surfeur aléatoire, qui considère le comportement d'un surfeur hypothétique sur Internet qui clique sur des liens au hasard. Considérons le corpus de pages web ci-dessous, où une flèche entre deux pages indique un lien d'une page à l'autre**.**



[1] https://cs50.harvard.edu/ai/2020

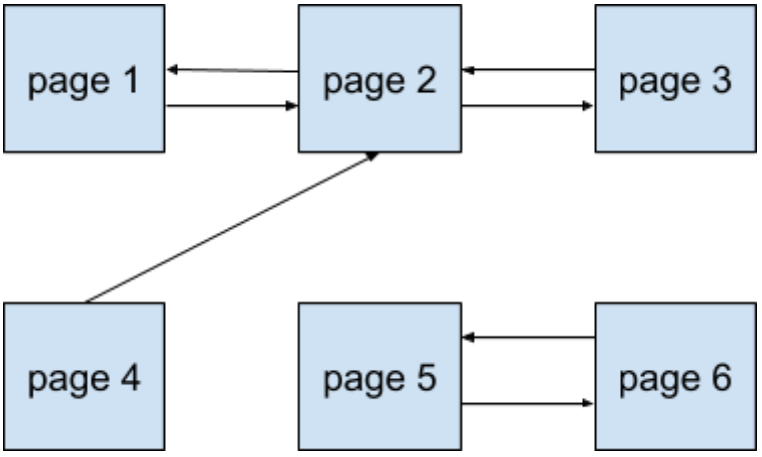
Le modèle du surfeur aléatoire imagine un surfeur qui commence par une page web au hasard, puis choisit aléatoirement les liens à suivre. Si l'internaute se trouve sur la page 2, par exemple, il choisira au hasard entre la page 1 et la page 3 à visiter ensuite (les liens en double sur la même page sont traités comme un lien unique, et les liens d'une page vers elle-même sont également ignorés). S'il choisit la page 3, l'internaute choisira alors au hasard entre la page 2 et la page 4 qu'il visitera ensuite.

Le PageRank d'une page peut donc être décrit comme la probabilité qu'un surfeur aléatoire se trouve sur cette page à un moment donné. Après tout, s'il y a plus de liens vers une page particulière, il est plus probable qu'un surfeur aléatoire se retrouve sur cette page. En outre, un lien provenant d'un site plus important a plus de chances d'être cliqué qu'un lien provenant d'un site moins important vers lequel moins de pages renvoient, de sorte que ce modèle gère également la pondération des liens en fonction de leur importance.

Ce modèle gère donc également la pondération des liens en fonction de leur importance. Une façon d'interpréter ce modèle est la chaîne de Markov, où chaque page représente un état, et chaque page dispose d'un modèle de transition qui choisit parmi ses liens au hasard. À chaque étape, l'état passe à l'une des pages liées par l'état actuel.

En échantillonnant les états au hasard de la chaîne de Markov, nous pouvons obtenir une estimation du PageRank de chaque page. Nous pouvons commencer par choisir une page au hasard, puis continuer à suivre les liens au hasard, en gardant une trace du nombre de fois où nous avons visité chaque page. Une fois que nous avons rassemblé tous nos échantillons (sur la base d'un nombre choisi à l'avance), la proportion du temps passé sur chaque page peut être une estimation du PageRank de cette page.

Cependant, cette définition du PageRank s'avère légèrement problématique, si l'on considère un réseau de pages comme celui présenté ci-dessous.



Imaginons que de manière aléatoire nous commençons à échantillonner la page 5. Nous n'aurions alors pas d'autre choix que d'aller à la page 6, puis à la page 5, puis à la page 6, et ainsi de suite. Nous aurions une estimation de 0,5 pour le PageRank des pages 5 et 6, et une estimation de 0 pour le PageRank de toutes les pages restantes, puisque nous avons passé tout notre temps sur les pages 5 et 6 et que nous n'avons jamais visité aucune des autres pages.

Pour que nous puissions toujours aller ailleurs dans le corpus des pages web, nous introduisons dans notre modèle un facteur d'amortissement d. Avec la probabilité d (où d est généralement fixé autour de 0,85), l'internaute choisira au hasard l'un des liens de la page en cours. Mais sinon (avec la probabilité 1 - d), l'internaute aléatoire choisit au hasard une des pages du corpus (y compris celle sur laquelle il se trouve actuellement).

Nous pouvons également définir le PageRank d'une page en utilisant une expression mathématique récursive. Soit PR(p) le PageRank d'une page p donnée : la probabilité qu'un surfer aléatoire se retrouve sur cette page. Comment définir le PR(p) ? Nous savons qu'il y a deux façons pour qu'un surfeur aléatoire se retrouve sur la page :

- Avec la probabilité 1 - d, l'internaute choisit une page au hasard et se retrouve à la page p.
- Avec la probabilité d, l'internaute a suivi un lien d'une page i à la page p.

La première condition est assez simple à exprimer mathématiquement : c'est 1 - d divisé par N, où N est le nombre total de pages de l'ensemble du corpus. Cela s'explique par le fait que la probabilité de 1 - d de choisir une page au hasard est répartie de manière équitable entre les N pages possibles.

Pour la deuxième condition, nous devons considérer chaque page i qui peut nous conduire à la page p. Pour chacune de ces pages entrantes, NumLinks(i) est le nombre de liens sur la page i. Chaque page i qui renvoie à p dispose de son propre PageRank, PR(i), représentant la probabilité que nous soyons sur la page i à un moment donné. Et comme à partir de la page i nous nous rendons à n'importe quel lien de cette page avec une même probabilité, nous divisons PR(i) par le nombre de liens NumLinks(i) pour obtenir la probabilité que nous soyons sur la page i et que nous choisissons le lien vers la page p.

Cela nous donne la définition suivante du PageRank pour une page p.

$$PR(p) = (1 - d)/N + d * \sum(PR(i)/NumLinks(i))$$

Dans cette formule,

- d est le facteur d'amortissement,
- N est le nombre total de pages dans le corpus,
- i s'étend sur toutes les pages qui renvoient à la page p, et
- NumLinks(i) est le nombre de liens présents sur la page i.

Nous pouvons calculer de différentes manières les valeurs de PageRank pour chaque page ? Nous pouvons le faire par itération : commençons par supposer que le PageRank de chaque page est 1 / N (c'est-à-dire qu'il y a autant de chances d'être sur n'importe quelle page). Ensuite, utilisez la formule ci-dessus pour calculer de nouvelles valeurs de PageRank pour chaque page, sur la base des valeurs de PageRank précédentes. Si nous continuons à répéter ce processus, en calculant un nouvel ensemble de valeurs de PageRank pour chaque page sur la base de l'ensemble précédent de valeurs de PageRank, les valeurs de PageRank finiront par converger (c'est-à-dire qu'elles ne changeront pas de plus d'un petit seuil à chaque itération).

Dans ce projet, vous mettrez en œuvre ces deux approches pour le calcul du PageRank - en calculant à la fois par échantillonnage des pages d'un internaute aléatoire avec la chaîne de Markov et en appliquant de manière itérative la formule du PageRank.

4. Présentation du code source pagerank.zip

Regardons le fichier *pagerank.py*. Remarquez d'abord la définition de deux constantes en haut du fichier : *DAMPING* représente le facteur d'amortissement et est initialement fixé à **0,85**. *SAMPLES* représente le nombre d'échantillons que nous utiliserons pour estimer le PageRank en utilisant la méthode d'échantillonnage, initialement fixée à **10.000** échantillons.

Maintenant, observons la fonction principale *main*. Elle attend un argument de ligne de commande, qui sera le nom d'un répertoire d'un corpus de pages web pour lequel nous aimerions calculer le PageRank. La fonction *crawl* prend ce répertoire, analyse tous les fichiers HTML qui s'y trouvent et renvoie un dictionnaire représentant le corpus. Les clés de ce dictionnaire représentent des pages (par exemple, *"2.html"*), et les valeurs du dictionnaire sont un ensemble de toutes les pages liées par la clé (par exemple {*"1.html", "3.html"*}).

La fonction principale *main* appelle ensuite la fonction *sample_pagerank*, dont le but est d'estimer le PageRank de chaque page par échantillonnage. La fonction prend comme arguments le corpus de pages généré par le *crawl*, ainsi que le facteur d'amortissement et le nombre d'échantillons à utiliser. Au final, *sample_pagerank* devrait retourner un dictionnaire dont les clés sont le nom de chaque page et les valeurs du PageRank estimé de chaque page (un nombre compris entre 0 et 1).

La fonction principale *main* appelle également la fonction *iterate_pagerank*, qui calcule également le PageRank pour chaque page, mais en utilisant la méthode de la formule itérative au lieu de l'échantillonnage. La valeur de retour devrait être au même format, et nous espérons que la sortie de ces deux fonctions sera similaire lorsqu'elles seront données au même corpus !

5. Travail à faire

Compléter l'implémentation des fonctions *transition_model*, *sample_pagerank*, et *iterate_pagerank*.

La fonction *transition_model* doit retourner un dictionnaire représentant la distribution de probabilité sur la page qu'un internaute aléatoire visiterait ensuite, compte tenu d'un corpus de pages, d'une page actuelle et d'un facteur d'amortissement.

- La fonction accepte 3 arguments: *corpus*, *page*, et *damping_factor*.
    - *corpus* est un dictionnaire en Python qui associe un nom de page à un ensemble de toutes les pages accessibles à partir de la page clé.
    - *page* est une chaîne de caractères représentant la page sur laquelle se trouve le surfer aléatoire.
    - *damping_factor* est un nombre flottant représentant le facteur d'amortissement à utiliser lors de la génération des probabilités.
- La valeur de retour de la fonction doit être un dictionnaire Python avec une clé pour chaque page du corpus. Chaque clé doit être associée à une valeur représentant la

probabilité qu'un internaute choisisse la page suivante au hasard. La somme des valeurs de cette distribution de probabilité retournée doit être égale à 1.
    - Avec la probabilité *damping_factor*, le surfer aléatoire doit choisir au hasard un des liens de la page avec une probabilité égale.
    - Avec la probabilité 1 - damping_factor, le surfer aléatoire devrait choisir au hasard une des pages du corpus avec une probabilité égale.
- Par exemple, si le corpus était *{"1.html" : {"2.html", "3.html"}, "2.html" : {"3.html"}, "3.html" : {"2.html"}}*, la page était *"1.html"*, et le facteur d'amortissement était de **0.85**, alors la sortie de *transition_model* devrait être *{"1.html" : 0,05, "2.html" : 0,475, "3.html" : 0.475}*. En effet, avec une probabilité de **0.85**, nous choisissons au hasard de passer de la page **1** à la page **2** ou à la page **3** (la probabilité que chacune des pages 2 ou 3 commence est donc de **0.425**), mais chaque page reçoit un **0.05** supplémentaire car avec une probabilité de **0.15**, nous choisissons au hasard parmi les trois pages.
- Si la page n'a pas de liens sortants, alors *transition_model* devrait retourner une distribution de probabilité qui choisit au hasard parmi toutes les pages avec une probabilité égale. (En d'autres termes, si une page n'a pas de liens, on peut prétendre qu'elle a des liens vers toutes les pages du corpus, y compris elle-même).

La fonction *sample_pagerank* doit accepter un corpus de pages web, un facteur d'amortissement et un certain nombre d'échantillons, et retourner un PageRank estimé pour chaque page.

- La fonction prend en entrée trois arguments : le *corpus*, un *facteur_d'amortissement* et *n*.
    - Le *corpus* est un dictionnaire Python qui associe un nom de page à un ensemble de toutes les pages accessibles à partir de cette page.
    - Le *damping_factor* est un nombre réel flottant représentant le facteur d'amortissement à utiliser par le modèle de transition.
    - Le nombre **n** est un nombre entier représentant le nombre d'échantillons qui doivent être générés pour estimer les valeurs de PageRank.
- La fonction retourne un dictionnaire Python avec une clé pour chaque page du corpus. Chaque clé doit être associée à une valeur représentant le PageRank estimé de cette page (c'est-à-dire la proportion de tous les échantillons qui correspondent à cette page). La somme des valeurs de ce dictionnaire doit être égale à 1.
- Le premier échantillon doit être généré en choisissant une page au hasard.
- Pour chacun des échantillons restants, l'échantillon suivant doit être généré à partir de l'échantillon précédent sur la base du modèle de transition de l'échantillon précédent.
    - Vous voudrez probablement passer l'échantillon précédent dans votre fonction *transition_model*, avec le *corpus* et le *damping_factor*, pour obtenir les probabilités pour l'échantillon suivant.
    - Par exemple, si les probabilités de transition sont *{"1.html" : 0.05, "2.html" : 0,475, "3.html" : 0,475}*, alors pour **5%** du temps la prochaine page sélectionnée devrait être *"1.html"*, **47,5%** du temps la prochaine page sélectionnée devrait être *"2.html"*, et **47,5%** du temps la prochaine page sélectionnée devrait être *"3.html"*.
- *n* est supposé être au moins égale à **1.**

La fonction *iterate_pagerank* prend en argument un corpus de pages web et un facteur d'amortissement, calcule les PageRanks sur la base de la formule d'itération décrite ci-dessus, et renvoie le PageRank de chaque page avec une précision de 0,001.

- La fonction accepte deux arguments : corpus et damping_factor.
    - Le corpus est un dictionnaire Python qui associe le nom d'une page à un ensemble de toutes les pages liées à cette page.
    - Le damping_factor est un nombre à virgule flottante représentant le facteur d'amortissement à utiliser dans la formule de PageRank.
- La valeur de retour de la fonction doit être un dictionnaire Python avec une clé pour chaque page du corpus. Chaque touche doit être associée à une valeur représentant le PageRank de la page. La somme des valeurs de ce dictionnaire doit être égale à 1.
- La fonction doit commencer par attribuer à chaque page un rang de 1 / N, où N est le nombre total de pages dans le corpus.
- La fonction doit ensuite calculer de façon répétée de nouvelles valeurs de classement en fonction de toutes les valeurs de classement actuelles, selon la formule de PageRank de la section "Contexte". (c'est-à-dire calculer le PageRank d'une page sur la base des PageRanks de toutes les pages qui y sont liées).
    - Une page qui n'a pas de liens du tout doit être interprétée comme ayant un lien pour chaque page du corpus (y compris elle-même).
- Ce processus doit être répété jusqu'à ce qu'aucune valeur de PageRank ne change de plus de 0,001 entre les valeurs de classement actuelles et les nouvelles valeurs de classement.

6. Indications

Vous ne devez rien modifier d'autre dans *pagerank.py* que les trois fonctions dont la spécification vous est donnée ci-dessus, bien que vous puissiez écrire des fonctions supplémentaires et/ou importer d'autres modules de la bibliothèque standard Python. Vous pouvez également importer numpy ou pandas, si vous les connaissez bien, mais vous ne devez pas utiliser d'autres modules Python tiers.

7. Soumission

Le dépôt se fera en ligne sur la plateforme https://fad.univ-thies.sn/. Vous êtes enrôlé dans l’espace du cours d’IA. Vous pouvez aussi partager avec moi votre espace Git contenant votre projet.