Rapport du projet Search Engine Machine

CABRERA Cyril - LOK Tshanon

Polytech Paris-Saclay - ET5 Info 2021-2022

Table des matières

... A faire ...

Introduction

Google est une entreprise américaine créée en 1998 par Larry Page et Sergey Brin. Sa fondation repose sur le fameux "Google Search". Ce moteur de recherche est le site le plus visité au monde et le plus utilisé, gérant 3.5 milliards de requêtes par jour. Cela représente 92% de part de marché en 2021.

Cette domination se justifie par la fiabilité de ses réponses. Cela est dû en partie à son algorithme "PageRank". Cette algorithme trie les pages Internet en leur attribuant un rang en fonction de plusieurs critères particuliers. Plus les pages seront hautes de ce système de classification, plus elles seront recommandées aux utilisateurs qui cherchent des termes qui sont en liens avec celles-ci.

Dans ce projet de Data Science, nous avons décidé de recréer les caractéristiques principales de "Google Search", à savoir, l'algorithme "PageRank" de Google ainsi que la fonctionnalité de recherche par mot-clé.

Méthodes et technologies utilisées

Matrice Google et "PageRank"

Matrice Google

L'algorithme "PageRank" a été développé par Google afin de classer un certains nombres de sites web en fonction de leur popularité. La popularité d'une page web est définie par le nombre de références qui y sont faites sur tous les autres sites repertoriés.

Considérons par exemple un ensemble de sites web A, B, C, D, E et F qui pointent les uns vers les autres. Si A est référencé sur tous les autres sites, alors

il aura une plus grande popularité que les autres. Lors d'une recherche, A sera classé plus haut que B, C, D, E et F dans les résultats.

Le web peut être assimilé à une chaîne de Markov où chaque point représente une page web et chaque transition représente le lien de référence entre ces pages. Donc si A pointe vers B, alors A fait référence à B sur sa page web. À ces transitions sont associées des probabilités qui sont la répartition équilibrée de la probabilité d'aller vers une page référencée.

Mise en situation : Supposons que l'on ait 6 pages web qui pointent les unes vers les autres de la manière suivante.

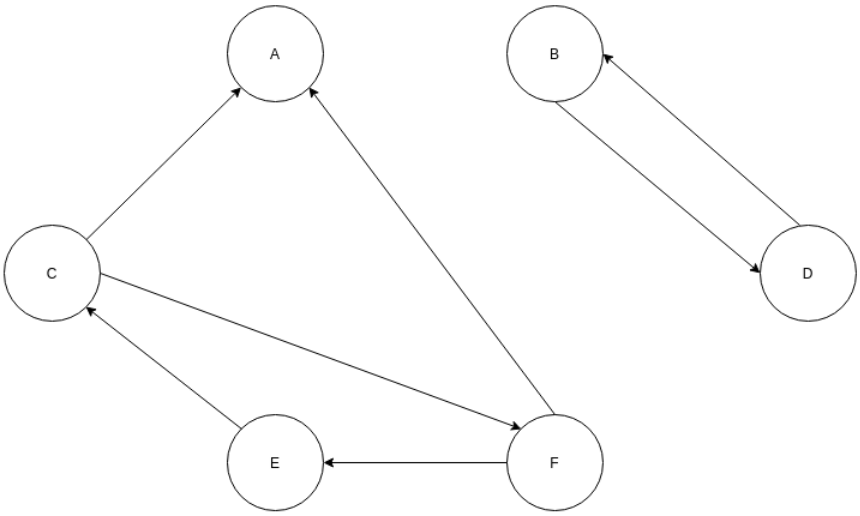


Figure 1: title

Dans notre cas, on retrouve la matrice de transition suivante :

$$\begin{matrix} A & B & C & D & E & F \end{matrix} \; L = \begin{pmatrix} 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \end{pmatrix} \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \end{matrix}$$

Comme on peut le constater, la somme des éléments de la première colonne vaut 0 car l'élément A ne fait référence à aucun autre état. Le problème est que l'algorithme ne pourra pas converger. Pour pallier ce problème, on suppose ainsi que tous les noeuds qui ne font référence à aucun état font maintenant référence

à tous les états. On obtient ainsi la matrice S suivante :

$$\begin{matrix} A & B & C & D & E & F \end{matrix} S = \begin{pmatrix} \frac{1}{6} & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{6} & 0 & 0 & 1 & 0 & 0 \\ \frac{1}{6} & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{6} & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{6} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{6} & 0 & \frac{1}{2} & 0 & 0 & 0 \end{pmatrix} \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \end{matrix}$$

Enfin, on remarque qu'il y a deux sous-graphes qui se dressent. Le graphe [A,C,E,F] et [B,D]. On ne peut pas passer du graphe [A,C,E,F] à [B,D] et inversement. Google a donc mis en place une formule :

$$M = \alpha S + \frac{1-\alpha}{N} S$$

avec alpha qui est un "damping factor" :

$$\alpha = 0.85$$

On obtient ainsi la matrice Google M suivante :

$$\begin{matrix} A & B & C & D & E & F \end{matrix} M = \begin{pmatrix} 0.116667 & -0.025 & 0.4 & -0.025 & -0.025 & 0.4 \\ 0.116667 & -0.025 & -0.025 & 0.825 & -0.025 & -0.025 \\ 0.116667 & -0.025 & -0.025 & -0.025 & 0.825 & -0.025 \\ 0.116667 & 0.825 & -0.025 & -0.025 & -0.025 & -0.025 \\ 0.116667 & -0.025 & -0.025 & -0.025 & -0.025 & 0.4 \\ 0.116667 & -0.025 & 0.4 & -0.025 & -0.025 & -0.025 \end{pmatrix} \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \end{matrix}$$

On se retrouve enfin avec une unique chaine de Markov régulière : pas d'état absorbant et pas de chaine secondaire. Tous les états sont donc réguliers, c'est-à-dire que ils peuvent tous être quittés ou rejoints.

Page Rank

Nous avons maintenant en place la matrice M correspondant à la matrice Google. Il est possible, à partir de cette matrice de déterminer le rang de chaque page web.

Qu'est ce qu'un rang ? Le rang correspond à la popularité d'un site web en fonction des autres.

Explications :

On considère notre matrice Google M et un vecteur P(t), correspondant à la répartition de la population dans les différents sites web.

À un temps t donné, si on souhaite connaitre la répartition de la population dans les différents sites web après un temps $\Delta t = 1$, il faudrait effectuer cette opération :

$$P(t + \Delta t) = M \cdot P(t)$$

On obtient donc à la période suivante :

$$P(t + 2\Delta t) = M\check{s} \cdot P(t)$$

Comme la matrice M est indépendante du temps, c'est-à-dire homogène, on en déduit la formule suivante :

$$P(x) = M^x P(0)$$

avec x le pas de temps

Donc, après plusieurs pas de temps, on a une probabilité de $(M^x)_{ij}$, pour passer de l'état j à i en x pas de temps.

On remarque qu'à partir d'un certain temps t, la répartition de la population ne varie plus beaucoup : c'est notre **état d'équilibre**.

Pour déterminer cet état d'équilibre, il faudrait calculer :

$$P(\infty) = M^\infty P(0)$$

Si on attend suffisament longtemps, l'état final i ne dépend plus de l'état initial j. Donc $(M^\infty)_{ij}$ ne dépend plus de j. On peut noter $(M^\infty)_{ij}$ par $(\vec{\pi}, \vec{\pi}, ...)$ avec $\vec{\pi} = (\pi_1, \pi_2, ...)$.

$$\begin{aligned} P(\infty) &= M^\infty \cdot P(0) \\ &= \vec{\pi} \cdot (\sum_i \vec{P_i}(0)) \\ &= \vec{\pi} \cdot 1 \\ &= \vec{\pi} \\ \iff P(\infty) &= M \cdot P(\infty) \\ &= \vec{\pi} \end{aligned}$$

Donc, on a : $M \cdot \vec{\pi} = \vec{\pi}$

Donc on sait que $\vec{\pi}$ est vecteur propre de M de valeur propre 1.

Ainsi, pour déterminer le rang des différentes pages web, il faut obligatoirement déterminer le vecteur propre associé à la valeur propre 1, c'est-à-dire trouver v tel que :

$$\begin{aligned} M \cdot v &= \lambda v \\ &= 1v \end{aligned}$$

Les vecteurs propres et les valeurs propres peuvent être déterminés par $E(M - \lambda I)$. Le noyau d'une matrice A est défini par :

$$Ker(A) = \{v \in R^n | \forall u \in A, u \cdot v = 0\}$$

On sait également que $Ker(A) = E(A - \lambda I)$ et que :

$$E(A - \lambda I) \iff (A - \lambda I) \cdot v = 0$$

On calcule donc les vecteurs propres de M :

$$\begin{pmatrix} 0.75528512 & 0.70000128 & -0.09852262 & -0.25754839 + 0.47535414i & -0.25754839 - 0.47535414i & -0.85 \end{pmatrix}$$

$$\begin{pmatrix} -3.82640508 \times 10^{-1} & 5.57969940 \times 10^{-1} & -9.04030650 \times 10^{-1} & -3.97035708 \times 10^{-1} - 0.07732039i & -3 \\ 5.72319646 \times 10^{-1} & -3.81754853 \times 10^{-1} & 1.35021942 \times 10^{-1} & 3.93007272 \times 10^{-2} + 0.02675777i & 3.9 \\ -3.07601626 \times 10^{-1} & 4.37093602 \times 10^{-1} & 2.38101306 \times 10^{-1} & 6.83723701 \times 10^{-1} & \\ 5.72319646 \times 10^{-1} & -3.81754853 \times 10^{-1} & 1.35021942 \times 10^{-1} & 3.93007272 \times 10^{-2} + 0.02675777i & 3.9 \\ -2.09552640 \times 10^{-1} & 2.92592167 \times 10^{-1} & 1.23074120 \times 10^{-1} & -1.40994194 \times 10^{-1} + 0.39525253i & -1 \\ -2.44858362 \times 10^{-1} & 3.47181536 \times 10^{-1} & 2.72813604 \times 10^{-1} & -2.24294652 \times 10^{-1} - 0.37144722i & -2 \end{pmatrix}$$

On prend la valeur propre la plus proche de 1 en valeur absolue. Dans notre exemple, il s'agit du vecteur propre associé à -0.85 à savoir :

$$\begin{pmatrix} -9.15181426 \times 10^{-18} \\ -7.07106781 \times 10^{-1} \\ -9.81970491 \times 10^{-17} \\ 7.07106781 \times 10^{-1} \\ 1.37578635 \times 10^{-16} \\ 4.21772479 \times 10^{-17} \end{pmatrix}$$

Après un certain temps, la distribution se stabilise et les valeurs sont similaires à celles du vecteur propre associé à la valeur 1.

Choix de la modélisation

Les pages web

Afin d'avoir un support pour notre modélisation, nous avons créé des pages web sous forme de fichiers txt. Les pages web sont actuellement : stackoverflow, reddit, youtube, marmiton, amazon, wikipedia. Tous ces fichiers txt sont regroupés dans le dossier "pages".

La mise en forme des pages web est la suivante : Le titre doit être sous la forme :

\[nom_de_la_page_web_sans_espace] + ".txt". exemple : reddit.txt

Dans chaque fichier, on trouve un titre pour la page, du texte et des pointeurs vers d'autres fichiers qui sont sous la forme : "pointeurvers : [nomPageWeb].txt".

Dans notre modélisation, les pages web pointent entre elles de la manière suivante :

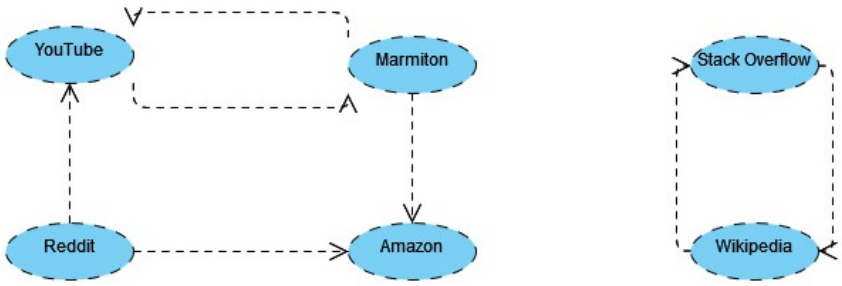


Figure 2: title

Par rapport à notre choix de modélisation, on peut voir que les sites les plus populaires sont :

1. stackoverflow
2. wikipedia (ex aequo avec 1)
3. marmiton
4. amazon
5. youtube
6. reddit

Il possible d'ajouter d'autres site web en respectant la mise en forme de la modélisation des pages web.

Fonctionnalités

Pour créer notre moteur de recherche, nous avons créé plusieurs fonctions dont les fonctionnalités seront présentées dans ce document.

Function count_Nb_Pages La fonction count_Nb_Pages prend en paramètre :

- un string correspondant au chemin du dossier contenant nos pages web.

Cette fonction permet de compter le nombre de pages web contenu dans ce dossier. La fonction retourne un nombre que l'on nomme n par la suite.

Function init_markov_chain La fonction init_markov_chain prend en paramètre :

- n, le nombre de pages web
- path, le chemin des pages web
- alpha, le damping factor que l'on a fixé à 0.85.

Cette fonction permet d'initialiser et de retourner :

- M, une matrice correspondant à notre matrice google,
- order, un vecteur contenant la liste des sites.

Le vecteur order permet de garder en mémoire l'ordre de lecture des différents sites web dans notre matrice M.

Explications: La matrice M est une matrice de transition. On passe d'un état A à un état B via les références de chaque page. Pour être sûr de l'ordre des états dans la matrice M, on met en place un vecteur order qui permet de garder en mémoire l'ordre.

Dans notre modèle, le vecteur order a été fixé comme cela:

$$order = \begin{pmatrix} amazon \\ marmiton \\ reddit \\ stackoverflow \\ wikipedia \\ youtube \end{pmatrix}$$

Le calcul de M a été réalisé de la manière que celle décrite dans la partie "Matrice Google" de ce document. Voici les étapes de calculs :

On commence par créer une matrice de transition L.

$$L = \begin{pmatrix} 0 & 0.5000 & 0.5000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0.5000 & 0.5000 & 0 & 0 & 0 \end{pmatrix}$$

La somme des colonnes ne valent pas toutes 1. On fait donc pointer les états associés à tous les états. On obtient la matrice S suivante :

$$S = \begin{pmatrix} 0.1667 & 0.5000 & 0.5000 & 0 & 0 & 0 \\ 0.1667 & 0 & 0 & 0 & 0 & 1.0000 \\ 0.1667 & 0 & 0 & 0 & 0 & 0 \\ 0.1667 & 0 & 0 & 0 & 1.0000 & 0 \\ 0.1667 & 0 & 0 & 1.0000 & 0 & 0 \\ 0.1667 & 0.5000 & 0.5000 & 0 & 0 & 0 \end{pmatrix}$$

Enfin, pour éviter tout problème de sous-graphe, on applique la formule $M = \alpha S + \frac{1-\alpha}{N} S$ ce qui nous donne M.

$amazon$ $marmiton$ $reddit$ $stackoverflow$ $wikipedia$ $youtube$

$$M = \begin{pmatrix} 0.1667 & 0.4500 & 0.4500 & 0.0250 & 0.0250 & 0.0250 \\ 0.1667 & 0.0250 & 0.0250 & 0.0250 & 0.0250 & 0.8750 \\ 0.1667 & 0.0250 & 0.0250 & 0.0250 & 0.0250 & 0.0250 \\ 0.1667 & 0.0250 & 0.0250 & 0.0250 & 0.8750 & 0.0250 \\ 0.1667 & 0.0250 & 0.0250 & 0.8750 & 0.0250 & 0.0250 \\ 0.1667 & 0.4500 & 0.4500 & 0.0250 & 0.0250 & 0.0250 \end{pmatrix} \begin{matrix} amazon \\ marmiton \\ reddit \\ stackoverflow \\ wikipedia \\ youtube \end{matrix}$$

Function find_rank La fonction find_rank prend en paramètre :

- n, le nombre de pages web
- path, le chemin des pages
- M, la matrice Google.

Cette fonction permet d'initialiser et de retourner :

- StablePR, un vecteur contenant le score de chaque page web

Pour ce faire, il nous suffit de trouver le vecteur propre de la matrice Google M avec une valeur propre associée de 1.

En MatLab, la fonction eig permet de retourner les valeurs propres et vecteurs propres d'une matrice.

$$X = \begin{pmatrix} 0.2659 & -0.3205 & -0.0000 & -0.0000 & 0.4676 & -0.0000 \\ 0.3178 & -0.4136 & 0.7071 & -0.7071 & -0.7399 & 0.0000 \\ 0.0918 & -0.0591 & -0.7071 & 0.7071 & -0.1057 & 0.0000 \\ 0.6120 & 0.5568 & 0.0000 & 0.0000 & -0.0449 & 0.7071 \\ 0.6120 & 0.5568 & 0.0000 & 0.0000 & -0.0449 & -0.7071 \\ 0.2659 & -0.3205 & 0.0000 & 0.0000 & 0.4676 & -0.0000 \end{pmatrix}$$

$$\begin{pmatrix} 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.7685 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.6268 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.8500 \end{pmatrix}$$

Comme expliqué dans la partie "Page Rank" de ce document, on en déduit StablePR en prenant le vecteur propre dont la valeur propre vaut 1, c'est à dire la première colonne de X.

$$StablePR = \begin{pmatrix} 0.2659 \\ 0.3178 \\ 0.0918 \\ 0.6120 \\ 0.6120 \\ 0.2659 \end{pmatrix} \begin{matrix} amazon \\ marmiton \\ reddit \\ stackoverflow \\ wikipedia \\ youtube \end{matrix}$$

Function sort_page_search La fonction sort_page_search permet de chercher un mot parmi les différentes pages web et de retourner la liste des pages web contenant ce mot.

La fonction prend en paramètre :

- hdata : le mot que l'on cherhce
- StablePR : le page rank des pages web
- path : le chemin de chaque page web
- order : un vecteur de string contenant l'ordre de lecture des pages
- n : le nombre de pages web

La fonction renvoie :

- result, un vecteur contenant la liste des pages web contenant le mot cherché et est triée de manière décroissante en fonction du page rank des pages web.

L'interface graphique

Création du vecteur P Nous avons donc créé un vecteur P qui correspond à une distribution aléatoire de population qui se trouve déjà sur les differents sites web. Chaque nombre se réfère au vecteur order qui permet de connaitre le nombre dans chaque site précisément.

$$P = \begin{pmatrix} 0.2330 \\ 0.2585 \\ 0.0369 \\ 0.2614 \\ 0.1818 \\ 0.0284 \end{pmatrix} \begin{matrix} amazon \\ marmiton \\ reddit \\ stackoverflow \\ wikipedia \\ youtube \end{matrix}$$

Par exemple, on 23,3% de la population se trouve initialement sur amazon.

Note : la somme des éléments du vecteur P vaut 1.

Moteur de recherche en html Afin de modéliser le moteur de recherche Google, une page html a été créée pour ressembler le plus possible à l'original. En effet, une barre de recherche permet d'obtenir la liste des sites qui contiennent le mot recherché.

Lorsqu'on lance notre moteur de recherche, une modélisation de l'évolution de la distribution de la population sur les différents sites web au cours du temps est faite. On peut ainsi voir sur quel site vont les personnes.

Pour notre modélisation, et pour la suite, le nombre total de personnes ne change pas et est fixé par le paramètre pop_tot.

Les deux représentations graphiques sont corrélées et les couleurs sont cohérentes entre les deux fenêtres.

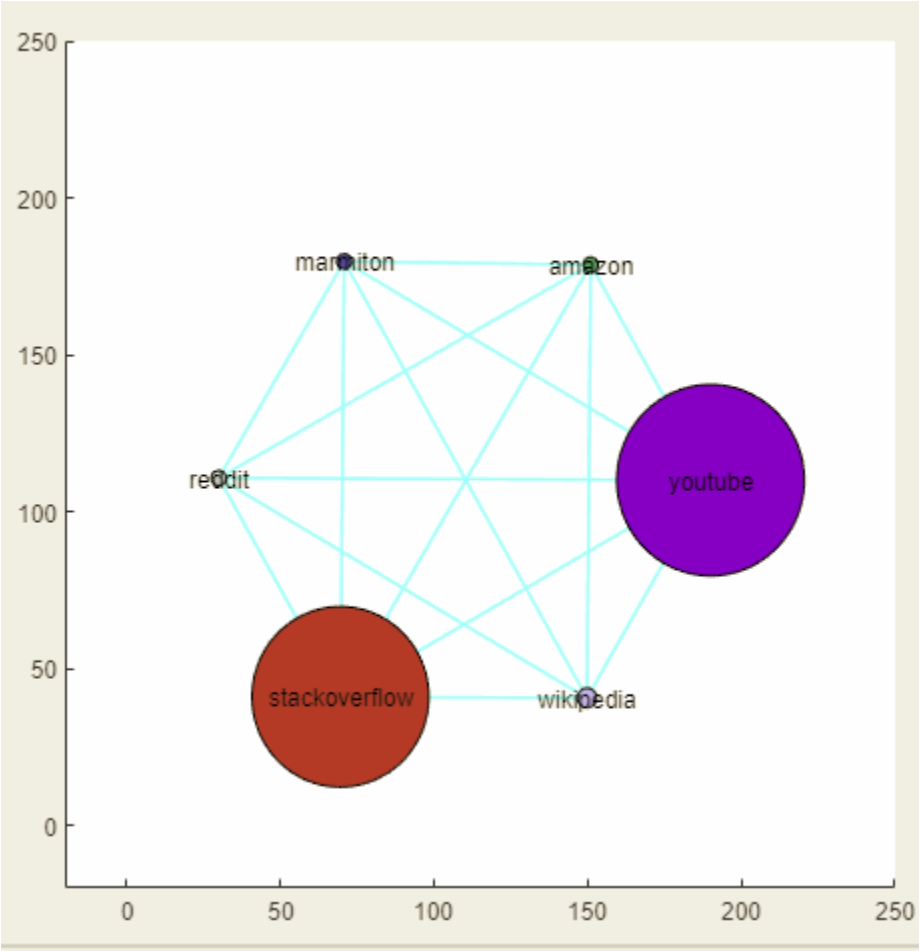


Figure 3: title

Dans la première fenêtre, la taille des cercles correspond à la quantité de personnes qui visitent le site en question.

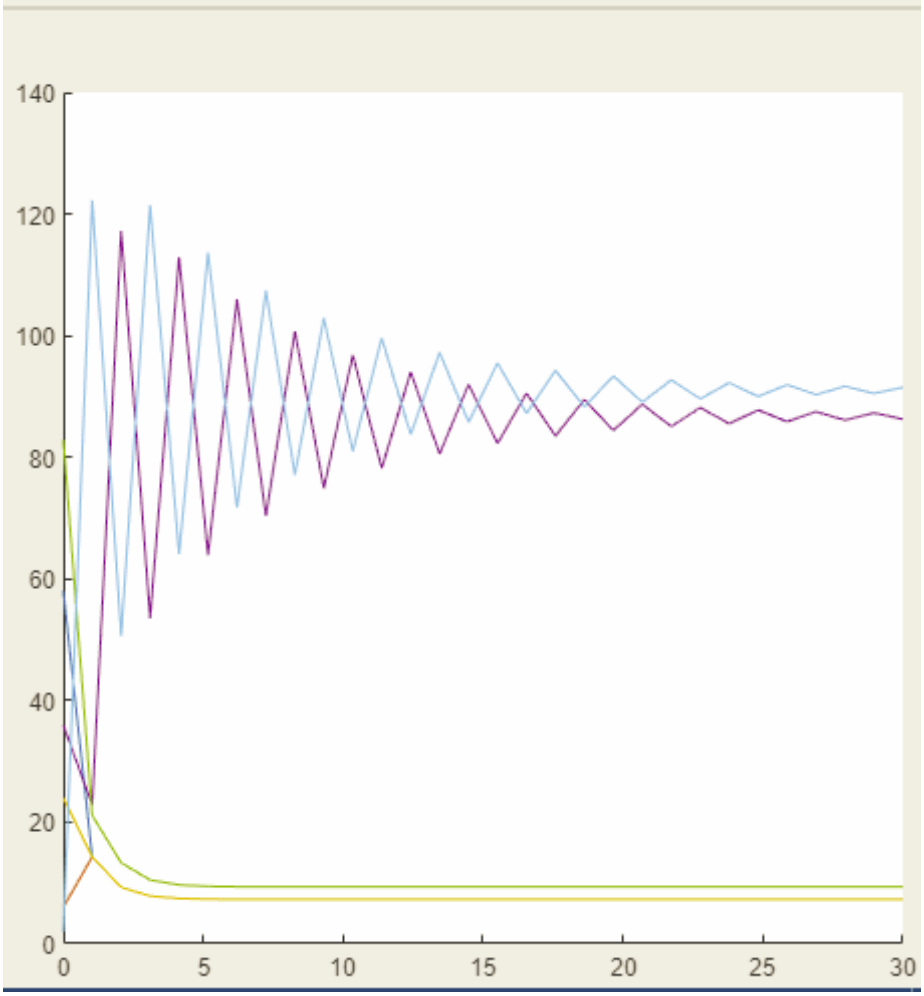


Figure 4: title

Dans la seconde fenêtre, on voit l'évolution de la population sur les sites en fonction du temps.

Obtention du mot cherché et retour des pages pertinentes Cependant, deux graphes ont été ajoutés sur la droite de la fenêtre.

- Retour depuis la page html
- Calcul fait

- Envoie des données vers la page html
- Vecteur P changé

$$P = \begin{pmatrix} 0.3333 \\ 0 \\ 0.3333 \\ 0 \\ 0 \\ 0.3333 \end{pmatrix} \begin{matrix} amazon \\ marmiton \\ reddit \\ stackoverflow \\ wikipedia \\ youtube \end{matrix}$$

- Modélisation dans l'interface graphique modifié pour correspondre à la situation nouvelle.

ATTENTION : PARLER DE LA DOCUmentATION

Après plusieurs recherches

Combien de temps on met pour revenir à l'état de stabilité ?

On peut le calculer de la manière suivante : On prend la deuxième plus grande (en valeur absolu) valeur propre. Calcul

Chez nous : ...

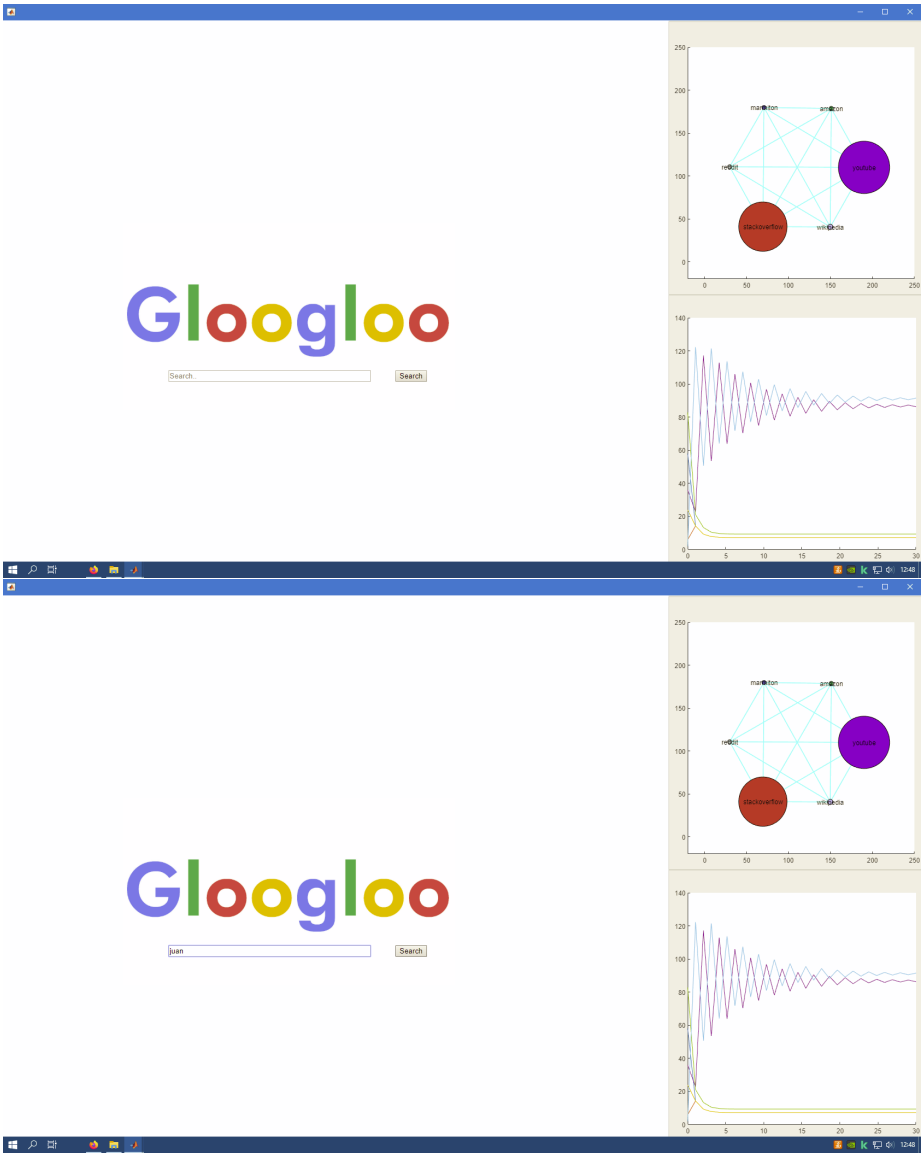
Et si on modifiait α ?

Répartition du travail

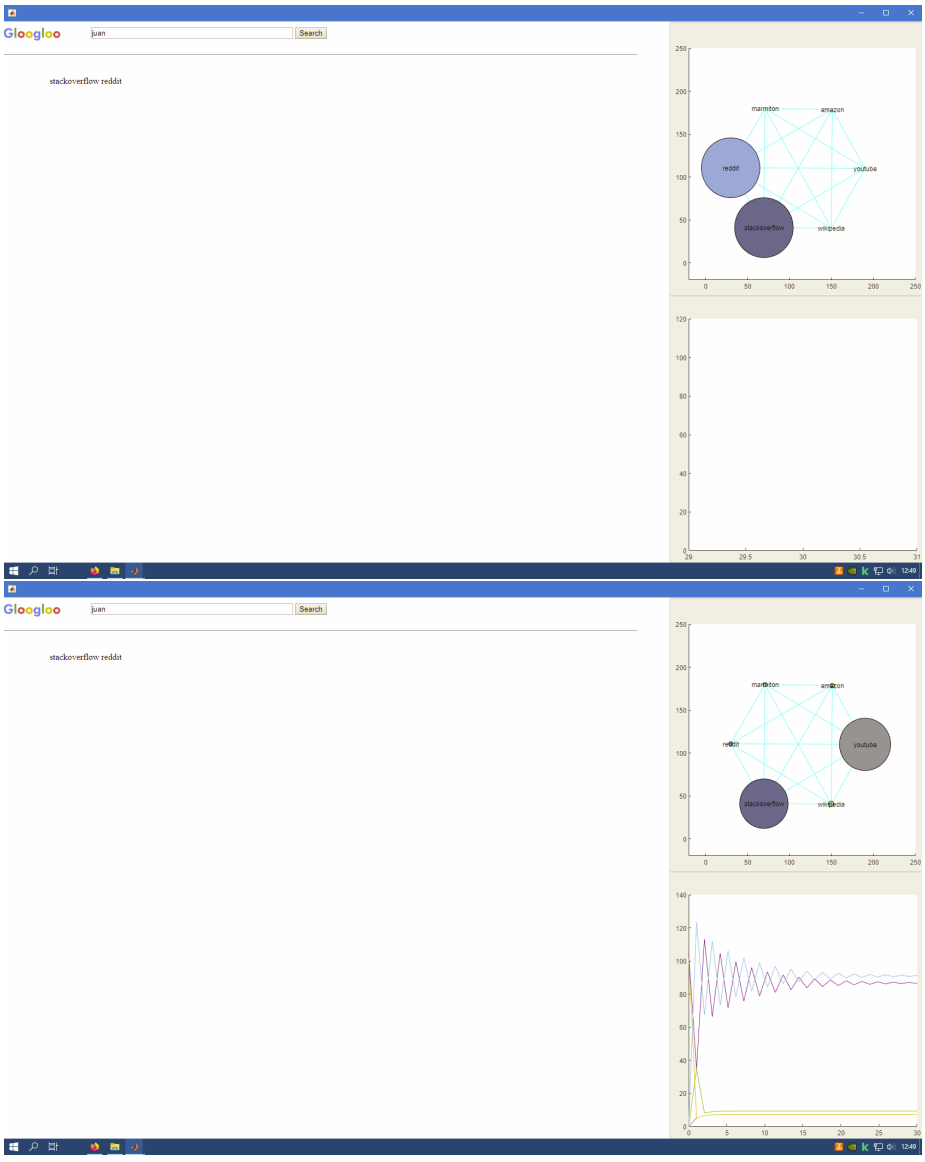
Une grande partie du projet a été effectuée en même temps par les deux membres du projet. En effet, la mise en place de la modélisation, la création de la matrice google M, les différents vecteurs ont été fait lors des 3 séances de TP de DataScience. Par la suite, l'interface graphique ainsi que les phases de recherche-réponse ont été ajoutés après ces trois séances. En outre, le projet a été réparti de manière équitable.

On s'est bien réparti le travail **chacal**.

Résultats



Commentaires sur les photos. On visualise bien la stabilité de la population dans notre modélisation : tout semble cohérent.



Conclusion

Ce que l'on aurait pu ajouter, améliorer. Résumé de la partie résultat. On aurait pu ajouter la pertinence du site web selon le nombre d'apparition du mot cherché. Si on ajoute des sites web, la modélisation de l'évolution de la population devient de moins en moins lisible. On aurait pu améliorer cela. On aurait aussi pu améliorer la façon dont les résultats sont affichés : c'est-à-dire le faire comme Google le fait. On aurait pu, pour aller plus loin, essayer de faire

une application : dans le sens où on aurait pu créer un exécuteur pour ne pas aller sur matlab puis cliquer sur RUN. On pense que ce projet aurait pu être utile pour une entreprise notament dans le cas où cette dernière souhaiterait développer un moteur de recherche interne à l'entreprise tout en essayant de faire des liens entre plusieurs sites, documents. Et ainsi voir les documents/sites les plus utiles.

Bibliographie

source : https://www.google.com/search/howsearchworks/algorithms/

source : "PageRank algorithm, fully explained", toward data science, Amrani Amine, 20 décembre 2020. Disponible sur : https://towardsdatascience.com/pagerank-algorithm-fully-explained-dc794184b4af

source : Cours de Data Science par Monsieur Desesquelles, délivré septembre 2021.

source : matlab