# Scoring des URL et l'initialisation de nouveau brin – Synthèse des échanges

Voici une synthèse de nos réflexion sur la gestion des scores de pertinence des brins et urls *(@EagleRed, @PangoCâlin*).

## Fonction d'évaluation et MVP

- Pour rappel, une fonction d'évaluation a été proposée par *@EagleRed* ici pour évaluer la pertiviralité d'un brin (ici : https://discord.com/channels/902607249632034828/904454302465130566/912824362585976902 ). Nous n'avons pour le moment pas le volume de données suffisant pour utiliser une telle fonction. De plus pour se donner les moyens d'utiliser cette fonction dans le futur, nous avons besoin qu'une URL soit associée à 1 ou plusieurs mots-clés (le nombre de mots clés restant à définir).
- Dans le cadre du MVP, on envisage donc plutôt un fonctionnement basé sur un calcul de scoring pondéré pour chacune des URL, ce qui est relativement simple à mettre en œuvre, voir ci-dessous
- Nous faisons ici un focus sur les scores de pertinence, des modifications seront certainement nécessaire dans une future version pour mieux prendre en compte la viralité d'une URL

## Evolution du scoring des URL

Recherche basée sur le score de pertinence

*Tableau 1 : Base de données au temps T*

| **Nom url** | **Popularité** | **Mot clé 1** | **Score 1** | **Mot clé 2** | **Score 2** | **Mot clé 3** | **Score 3** | **…** |
|---|---|---|---|---|---|---|---|---|
| Url 1 | 130 | Python | 50 | C++ | 40 | Javascript | 30 | … |
| Url 2 | 250 | Chat | 70 | Python | 30 | Chien | 20 | … |
| … | … | | | | | | | … |
| Url N | 10 | Virus | 10 | Bug | 8 | Problème | 5 | … |

Une personne recherche les mots clés suivant : Bug, Python, Virus

On regarde dans la base de données les url qui ont les mots clés

| Nom url | Popularité | Mot clé 1 | Score 1 | Mot clé 2 | Score 2 | Mot clé 3 | Score 3 | … |
|---|---|---|---|---|---|---|---|---|
| Url 1 | 130 | Python | 50 | C++ | 40 | Javascript | 30 | … |
| Url 2 | 250 | Chat | 70 | Python | 30 | Chien | 20 | … |
| … | … | | | | | | | … |
| Url N | 10 | Virus | 10 | Bug | 8 | Problème | 5 | … |

On calcule le score de chaque url en sommant les scores de chaque mot-clé et on retourne le brin trié par ordre de pertinence :

- Url 1 : 50
- Url 2 : 30
- Url N : 10 + 8 = 18

(On peut aussi imaginer que plus il y a de mots clés pertinents en même temps, plus l'url ressort, par exemple ici l'url N a deux mot clés très pertinent chez lui qui sont en commun avec notre problème, on pourrait donc multiplier son résultat de pertinence par 2 par rapports aux deux autres).

- Url N : (10 + 8)*2 = 18*2= 32

On obtient donc, dans l'ordre, les url suivant proposé à l'utilisateur : (1,N,2)

Naturellement il supprime l'onglet 2 qui enfaite traite des animaux, il reste donc (1,N)

On aura donc pour ces 3 mots clés Bug, python, virus dont les scores vont être abaissés pour l'Url 2 s'ils existent, mais augmentés pour l'Url 1 et N.

*Tableau 2: Base de données au temps T+1*

| Nom url | Popularité | Mot clé 1 | Score 1 | Mot clé 2 | Score 2 | Mot clé 3 | Score 3 | … |
|---|---|---|---|---|---|---|---|---|
| Url 1 | 131 | Python | 50+1 | C++ | 40 | Javascript | 30 | … |
| Url 2 | 250-0.5 | Chat | 70 | Python | 30-1 | Chien | 20 | … |
| … | … | | | | | | | … |
| Url N | 11 | Virus | 10+1 | Bug | 8+1 | Problème | 5 | … |

## Réponse au besoin exprimé :

Si on prend l'exemple  du besoin exprimé par *@caillef* et envisagé pour le MVP :

https://discord.com/channels/902607249632034828/904454302465130566/913377625551011891

« 

1. GET "/strands?keywords=best smartphone" *qui me retourne les brins qui ont des keywords en commun une array en JSON où chaque objet a : l'id du brin, le nom du brin (les keywords), la liste des liens et le nombre de fois qu'il a été forké*
2. POST "/strands/:strand_id/forks" *pour donner l'info que le strand a été forké, ça return l'id du nouveau brin (ça permettra de savoir combien de fois il l'a été)*
3. GET "/links?keywords=best smartphone" *pour placer les liens si la personne démarre d'un brin vide*
4. *peut-être qu'il faudra un autre endpoint pour dire "Je mets un j'aime sur ce lien" pour que le lien remonte plus facilement pour le même keyword, ou alors vous pourriez juste parcourir de votre côté tous les brins similaires et remonter les liens les plus présents. Si 3 brins ont déjà "Comparatif meilleur smartphone 2021 du site frandroid", alors je le place en premier si je génère un nouveau strand avec des mots clés similaires*

» 

Alors la structure de scoring précédente répond assez bien à ce besoin :

1. Pour le premier endpoint, on rajoute en complément du nombre de fork le score de pertinence (on pourra utiliser la fonction discutée en début de ce document post-MVP pour mixer pertinence / viralité)
2. Cette action pourrait déclencher un événement qui influe sur le score de viralité de toutes les URL du brin. On peut insérer ça dans la table d'événements ajoutée au modèle
3. Ici on peut imaginer un mix entre les liens connus par le réseau stigmee, et les liens non connus renvoyés par l'API de recherche. L'algorithme génétique proposé précédemment (https://discord.com/channels/902607249632034828/904454302465130566/911760147032993843 ) peut servir à juger rapidement si de nouvelles URL sont pertinentes par rapport à celles connues du réseau.
4. Une idée pour la fonctionnalité « j'aime » pourrait être de la transformer en un j'aime « keywordisé », qui permettrait de faire remonter un lien tout en étant plus fin sur la pondération donnée à chaque mot clé pour une URL. En effet, comme montré précédemment, un mot-clé peut être pertinent pour une URL et pas du tout pour une autre, et les mauvaises associations ne seront certainement pas rares lors des débuts du réseau.

## Fork d'un nouveau brin

Sur le principe de l'algo évolutionniste déjà proposé, on peut imaginer un équilibrage entre les différentes sources de données pour la proposition des liens d'un nouveau brin (Tinder-like)

- Pool initialization : Chaque source de données propose une URL de sa collection en lien avec les mots clés concernés, présentées par ordre de pertinence. Chaque source possède une probabilité de sélection (initialement équivalente pour toutes les sources)
- Des URL peuvent apparaitre en doublon sur plusieurs moteurs, ce qui augmente leur chance de sélection

*Tableau 3 : Exemple de population initiale*

| Stigmee-B1 : 0.25 | | | | | | | | Google : 0.25 | | | | | | | | DuckDuckGo : 0.25 | | | | | | | | Stigmee-B2 : 0.25 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | . | . | . | 8 | 1 | 2 | 3 | 4 | . | . | . | 8 | 1 | 2 | 3 | 4 | . | . | . | 8 | 1 | 2 | 3 | 4 | . | . | . | 8 |

- Mating : La sélection utilisateur impacte positivement ou négativement la probabilité que les prochaines URL proposées soient issues de la même source de données.

*Tableau 4 : Exemple d'Impact de la sélection (Tour d'initialisation)*

| | | | |
|---|---|---|---|
| Stigmee-B1::1 | YES | 0.25 | 0.25 + ((1/n) * (1/3 yes) / rank) = 0.25+0.010 = 0.26 |
| Google::1 | YES | 0.25 | 0.25 + ((1/n) * (1/3 yes) / rank) = 0.25+0.010 = 0.26 |
| DuckDuckGo::1 | YES | 0.25 | 0.25 + ((1/n) * (1/3 yes) / rank) = 0.25+0.010 = 0.26 |
| Steegmee-B2::1 | NO | 0.25 | 0.25 - ((1/n) * (1/1 no) / rank) = 0.25+0.031 = 0.219 |

*Tableau 5 : Exemple de contenu du brin à l'issu du tour 1*

| |
|---|
| Stigmee::1 |
| Google::1 |
| DuckDuckGo::1 |

- Les URL ayant été sélectionnées par l'utilisateur intègrent le pool de référencement de Stigmee, en complément de celles déjà existantes. Ceci augmente leur score, ce qui augmente également la pertinence des autres brins utilisant éventuellement cette URL.
- L'impact d'une sélection / non sélection est atténuée par le rang de l'URL dans le sous-ensemble concerné.
- Une URL sélectionnée quitte le pool au tour suivant, réduisant donc le sous-ensemble de la source de données correspondante

*Tableau 6 : Exemple de population au tour 2*

| Stigmee B1 : 0.26 | | | | | | | Google : 0.26 | | | | | | | DuckDuckGo : 0.26 | | | | | | | Stigmee B2 : 0.219 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | . | . | 7 | 1 | 2 | 3 | 4 | . | . | 7 | 1 | 2 | 3 | 4 | . | . | 7 | 1 | 2 | 3 | 4 | . | . | 7 |

Probabilité de sélection pour évaluation, pour un total de n URL dans le pool : P(source) * 1/n

- 0.29 * 1/28 pour les 3 premières sources.
- 0.13 * 1/28 pour la source 4

Et ainsi de suite jusqu'à arrêt de sélection par l'utilisateur ou épuisement des URL disponibles.

Evolution de $P^{[t]}(Source)$ selon le choix de l'utilisateur:

- Pour n URL dans le pool
- Avec rank = numéro d'ordre de l'url dans son sous-groupe

$$P^{[t-1]}(Source) \pm ((1/n) \ / \ rank)$$

## Remarque

Avec ces modifications, il ne s'agit plus d'un algorithme génétique à proprement parler, car il n'y a pas de crossover, ni de création d'enfants, la génération suivante étant simplement composée des survivants de la précédente. En contrepartie la sélection est pondérée de manière récursive. C'est plutôt une adaptation rapide qui permet de constituer facilement un nouveau brin à partir de sources multiples aux pertinences variables.