

# ANALYSE ET CLASSIFICATION D'URLS À DES FINS DE SEGMENTATION AUTOMATIQUE PAR GROUPES DE CIBLAGE

Matthieu Brito Antunes  
Data Scientist  
Tradelab - Paris  
septembre 2020

## Table des matières

|  |          |
|--|----------|
| <b>1. Introduction</b>   | <b>1</b> |
| <b>2. Segmentation d'URL</b>   | <b>1</b> |
| <b>3. Analyse de la structure des URL et premiers essais de segmentation automatique via <i>pattern trees</i></b>  | <b>1</b> |
| 3.1. Structure classique d'une URL . . . . .   | 1        |
| 3.2. Analyse de la structure des URLs via <i>pattern trees</i> . . . . .   | 2        |
| 3.3. Résultats de transformations d'URL et orientation vers une autre méthode                                      | 2        |
| <b>4. Approche statistique de la classification automatique d'URL</b>  | <b>3</b> |
| 4.1. Distance et similarité . . . . .  | 3        |
| 4.2. DBSCAN et segmentation . . . . .  | 4        |
| 4.3. Résultats de <i>clustering</i> . . . . .  | 4        |
| 4.4. Limites de la méthode . . . . .   | 6        |
| <b>5. Approche sémantique de l'analyse des URL et segmentation automatique par la méthode des <i>n</i>-grammes</b> | <b>6</b> |
| 5.1. Le modèle <i>n</i> -grammes . . . . .   | 7        |
| 5.2. Construction du modèle <i>n</i> -grammes à partir des données d'URL disponibles                               | 7        |
| 5.3. Résultats de segmentation d'URL . . . . .   | 7        |
| <b>References</b>  | <b>8</b> |

## Liste des tableaux

## Table des figures

|   |  |   |
|---|--|---|
| 1 | Exemple de <i>pattern tree</i> construit à partir du site <a href="http://www.wretch.cc">www.wretch.cc</a> . . . . . | 3 |
| 2 | Résultats de <i>clustering</i> par DBSCAN selon différentes valeurs de <i>r</i> . . . . .                            | 5 |

## 1. Introduction

L'ensemble des adresses web (URL) des pages visitées sur Internet peut constituer une source pertinente lorsque l'on souhaite réfléchir à une méthode de ciblage publicitaire. Les informations extraites à partir du traitement des URL permettent de comprendre les mécanismes qui régissent le regroupement d'utilisateurs, ou *segmentation*.

La segmentation d'utilisateurs est une étape cruciale dans le processus de ciblage publicitaire en ligne et peut s'avérer très chronophage. Il apparaît donc essentiel de chercher à utiliser toute information qui permettrait de comprendre comment tendre vers son automatisation totale.

Les informations contenues dans la structure des URL nous renseignent principalement sur l'organisation globale d'un site web, et la manière dont sont créées les URL associées aux nouvelles pages. Ceci permet de construire des regroupements de pages web sans avoir à analyser leur contenu. Ces regroupements pourraient idéalement être réalisés de manière automatique, à partir d'un modèle de *clustering* qui recevrait une URL en entrée et fournirait en sortie le groupe auquel elle appartient après avoir analysé sa structure.

Ce rapport présente le travail de construction d'un modèle d'analyse et de classification automatique d'URL. La première partie du rapport est centrée sur l'étude de la problématique et des enjeux associés. Diverses pistes d'analyse de structure d'URL sont proposées. La seconde partie du rapport offre une vision théorique de la solution proposée pour automatiser la création de segments à partir des données d'URL collectées chaque jour chez Jellyfish France. La dernière partie du rapport présente la mise en place de la solution et les résultats obtenus sur des données récoltées sur une période fixe.

## 2. Segmentation d'URL

Le nombre croissant de visites de pages web et de liens échangés constitue un défi auquel doivent faire face les ingénieurs et chercheurs appliqués dans le vaste domaine du *web mining*. L'analyse de l'information contenue dans les pages web et les renseignements que des séquences d'URL peuvent apporter sont cruciaux lorsqu'il s'agit de mettre en place des stratégies visant à optimiser l'utilisation d'Internet, que ce soit à des fins publicitaires (ciblage, analyse de parcours de navigation), de sécurité (détection d'URL de pages malveillantes), de construction de moteurs de recherche, pour l'archivage du web, etc.

## 3. Analyse de la structure des URL et premiers essais de segmentation automatique via *pattern trees*

### 3.1. Structure classique d'une URL

Sur Internet, les pages web regroupent divers formats de données (texte, image, son...) et sont repérées par une chaîne de caractères renseignant deux choses :

- leur emplacement
- le protocole internet permettant d’y accéder <sup>1</sup>

Cette chaîne de caractères, appelée *adresse web* ou encore URL (pour *Uniform Resource Locator*) est générée pour chaque nouvelle page ajoutée à un site, et permet donc à l’utilisateur de savoir où il se trouve. Encore faut-il qu’il sache la lire ! On retrouve en effet plusieurs éléments constitutifs d’une URL, qui peuvent être présents ou absents selon le type de site sur lequel l’utilisateur se trouve. Globalement, une URL peut être décrite comme suit :

`http://www.bing.com/images/search?q=hp&form=bifd`

avec

- `http` : *scheme*
- `www.bing.com` : *authority*
- `images/search` : *path*
- `q=hp&form=bifd` : *query*

### 3.2. Analyse de la structure des URLs via *pattern trees*

La première approche vise à mettre en évidence des structures caractéristiques au sein des URL en faisant appel à des outils analogues aux arbres de décision [3]. L’idée est ici d’apprendre à transformer des URL doublons en une forme canonique à l’aide de règles de réécriture. À partir des règles d’écriture dégagées après analyse des URL disponibles, la segmentation automatique est possible sur de nouvelles URL, en considérant que les URL soumises aux mêmes règles de réécriture appartiennent au même segment.

L’ensemble des règles de réécriture est regroupé dans un *pattern tree*, i.e. un groupe de structures d’URL organisé selon une certaine hiérarchie. Au sein de cet arbre, chaque nœud représente un groupe d’URL ayant la même structure syntaxique. Cet arbre permet de rassembler l’ensemble de l’information statistique relative aux URL à disposition, augmentant ainsi la robustesse et la fiabilité du processus de segmentation. C’est sur cette idée que la première approche de ce travail est basée pour essayer de créer automatiquement des segments d’URL présentant des similitudes sans avoir à analyser le contenu des pages auxquelles elles renvoient (fig. 1).

### 3.3. Résultats de transformations d’URL et orientation vers une autre méthode

L’utilisation d’un *pattern tree* pour regrouper des URL en se basant sur leurs similarités syntaxiques présente l’inconvénient d’être principalement adaptée aux URL d’un seul et même site web. Lorsque l’algorithme de construction du *pattern tree* est fourni en URL trop différentes les unes des autres, des résultats peu significatifs peuvent être observés, le principal (et le plus fâcheux) étant l’obtention d’un unique segment d’URL

---

1. Les protocoles les plus connus sont ceux que l’on rencontre tous les jours dans la barre d’adresse de notre navigateur : `http` et `https` pour *HyperText Transfer Protocol (Secure)*.

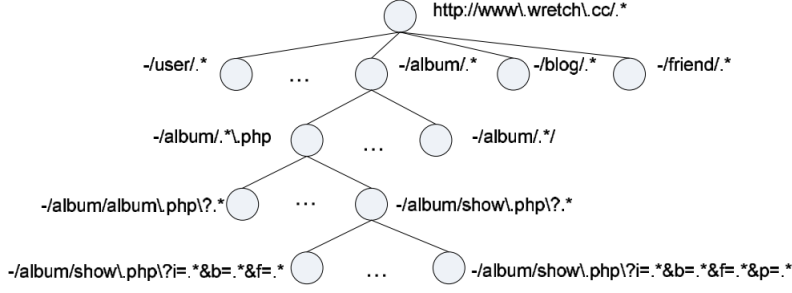


FIGURE 1 – Exemple de *pattern tree* construit à partir du site [www.wretch.cc](http://www.wretch.cc)

transformées en la forme `http://*/*/*`. On s'aperçoit qu'au sein de ce segment toutes les URL sont regroupées et qu'il ne présente par conséquent aucune spécificité exploitable dans l'optique d'un ciblage publicitaire d'utilisateurs.

## 4. Approche statistique de la classification automatique d'URL

La seconde approche adoptée s'est basée sur des outils largement répandus dans le domaine des systèmes de recommandation et d'analyse des préférences d'utilisateurs [2]. Le principe de cette approche statistique est basé sur le calcul de distance entre les objets à classer. D'une façon générale, retenons que tout objet que l'on cherche à faire passer par un processus de classification automatique (ou segmentation) peut être représenté par un vecteur. En d'autres termes, il est possible de projeter les caractéristiques des objets que l'on cherche à classer dans un espace au sein duquel leur classification est possible à l'aide de techniques de calcul efficaces.

### 4.1. Distance et similarité

En mathématiques, l'idée de distance renvoie à une application bien particulière visant à quantifier la *similarité* entre deux points  $x_1$  et  $x_2$  du plan ou de l'espace. Une mesure de similarité est appelée *distance* si et seulement si elle respecte les propriétés suivantes :

- $d(x_1, x_2) \geq 0, d(x_1, x_2) = 0 \iff x_1 = x_2$  (positivité)
- $d(x_1, x_2) = d(x_2, x_1)$  (symmétrie)
- $d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3) \forall (x_1, x_2, x_3)$  (inégalité triangulaire)

Il existe plusieurs manières de calculer la distance entre deux objets. On utilise l'une d'elles dans la vie de tous les jours sans même s'en apercevoir dans les espaces à deux ou trois dimensions : la distance Euclidienne, définie par la racine de la somme des différences entre les coordonnées des deux objets élevées au carré<sup>2</sup>.

<sup>2</sup>. Pour deux objets  $x = (x_1, x_2, \dots, x_n)$  et  $y = (y_1, y_2, \dots, y_n)$ , la distance qui les sépare vaut  $\sqrt{\sum_{i=1}^n (x_i - y_i)^2}$

Ici, la distance utilisée tient compte du fait que les objets à classer ne sont plus de simples points dans l'espace, mais des URL, qui sont en somme des chaînes de caractères. Trois mesures de distance entre deux URL  $s_1$  et  $s_2$  ont été étudiées : la distance de Levenshtein<sup>3</sup>, la distance de Jaro<sup>4</sup>, et une version modifiée de la distance de Levenshtein<sup>5</sup>.

Pour ce travail, la dernière mesure de distance  $d_{URL}$  a été retenue.

## 4.2. DBSCAN et segmentation

Une fois la distance entre chaque objet mesurée, il faut constituer automatiquement des groupes au sein desquels celle-ci n'excède pas une valeur seuil choisie avec parcimonie. Si cette valeur seuil est trop élevée, les groupes risquent de se révéler trop grands, englobant ainsi trop d'hétérogénéité. Si elle est au contraire trop petite, les groupes risquent de ne pas capturer assez d'information, et de trop nombreux groupes risquent d'être créés.

Tout ce processus de construction de groupes porte en apprentissage statistique le nom de *clustering*, et peut être réalisé au moyen de méthodes diverses et variées. Ici, c'est la méthode DBSCAN<sup>6</sup> qui a été choisie, par souci d'efficacité et de rapidité d'exécution.

Rappelons que l'idée derrière le *clustering* est de créer des groupes d'objets homogènes suffisamment denses pour contenir assez d'information, et suffisamment distincts pour que cette information soit correctement distribuée. DBSCAN fonctionne sur le principe suivant :  $r$  est le rayon dans lequel doivent se trouver au moins *MinPts* points ou plus pour que soit constituée un *cluster*. Pour un nouveau point donné, l'algorithme vérifie par la suite que son  $r$ -voisinage contient au moins *MinPts* points. Ce nouveau point appartient donc alors à un nouveau *cluster*, et l' $r$ -voisinage est parcouru de proche en proche afin de définir les limites de ce *cluster*.

## 4.3. Résultats de *clustering*

Une étape essentielle dans ce genre de travaux consiste à observer l'influence de la variation des paramètres des algorithmes que l'on utilise sur les résultats obtenus, c'est la *sensitivity analysis*. Ici, les deux paramètres que l'on peut faire varier touchent directement à la taille des *clusters* créés par DBSCAN, il s'agit du rayon  $r$  et du nombre de points minimum *MinPoints* les constituant.

$$3. \ d_{LVS} = \begin{cases} \max(i, j) & \text{if } \min(i, j) = 0, \\ \min \begin{cases} d_{LVS}(i-1, j) + 1 \\ d_{LVS}(i, j-1) + 1 \\ d_{LVS}(i-1, j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

4.  $d_{Jaro} = \frac{1}{3} \left( \frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right)$  où  $|s_i|$  est la longueur de la chaîne de caractères  $s_i$ ,  $m$  est le nombre de "caractères correspondants" et  $t$  est le nombre de "transpositions"

5.  $d_{URL(s_1, s_2)} = 1 - \frac{|s_1| + |s_2| - d_{LVS2}(s_1, s_2)}{|s_1| + |s_2|}$  où  $d_{LVS2}$  est une modification de  $d_{LVS}$  pondérant les remplacements de caractères par 2 plutôt que par 1

6. *Density-Based Spatial Clustering of Applications with Noise* est un algorithme utilisant la distance entre deux points et le nombre minimum de points choisis pour constituer un *cluster* comme paramètres de segmentation automatique

Dans les faits, nous avons obtenu des résultats significatifs et exploitables. Même si l'analyse aurait pu être poussée plus loin, nous nous satisfaisons pour l'instant des résultats que nous avons. On observe une claire variation du nombre de *clusters* créés et de la proportion de points considérés comme du bruit<sup>7</sup> en fonction du rayon  $r$  de *clustering* utilisé (fig. 2)

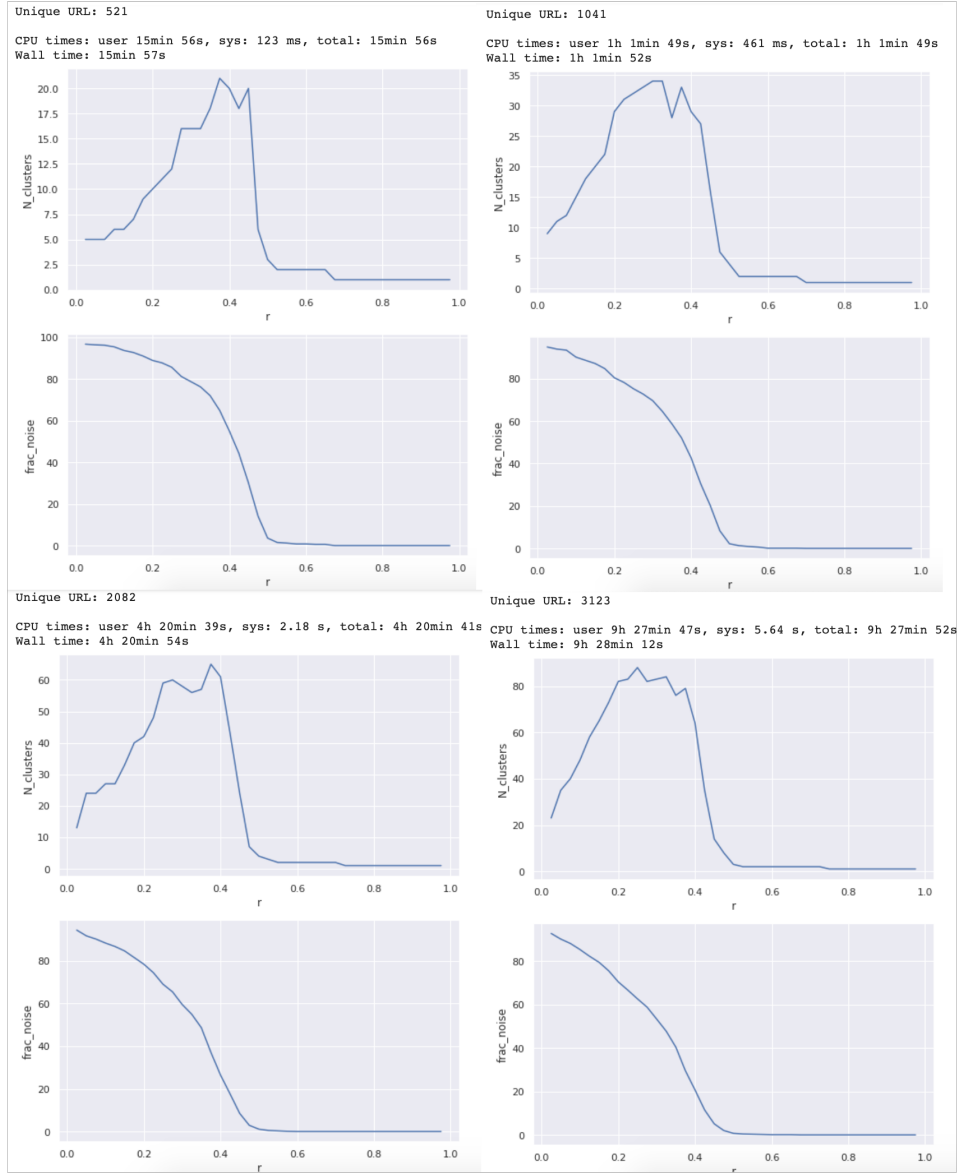


FIGURE 2 – Résultats de *clustering* par DBSCAN selon différentes valeurs de  $r$

7. Donc pas intégrés à un *cluster*

Précisons d'emblée que chacune des quatre paires de graphes correspond à un *clustering* réalisé sur un nombre croissant d'URL `Unique URL` en faisant varier le rayon  $r$ , `MinPoints` étant maintenu constant égal à 4. Le graphe du haut rend compte de l'évolution du nombre de *clusters* `N_clusters` obtenu en fonction de  $r$ , le graphe du bas rend compte de la proportion d'URL considérées comme du bruit `frac_noise` en fonction de  $r$ . L'idéal serait d'obtenir assez de *clusters* pour séparer efficacement les URL en segments sans toutefois maintenir trop d'URL à l'écart en les considérant comme du bruit.

Ces résultats nous apprennent plusieurs choses. Tout d'abord, la combinaison du calcul de distance entre URL par la fonction de Levenshtein modifiée  $d_{LVS2}$  et du *clustering* par DBSCAN pourrait être adaptée au traitement de notre problématique. Reste à vérifier que les *clusters* créés sur ces données d'entraînement rendent compte d'une certaine homogénéité et restent exploitables dans une optique de ciblage d'utilisateurs (qui reste l'objectif principal, rappelons-le à toutes fins utiles!). Nous y reviendrons.

Ensuite, il semble exister une valeur de  $r$  pour laquelle le nombre de *clusters* obtenus est maximal. S'il s'avère que les URL sont correctement distribuées (i.e. si le contenu des *clusters*) nous satisfait, alors on pourrait considérer cette valeur de  $r$  comme optimale, dans la mesure où elle permet une segmentation efficace. Là encore, reste à vérifier le contenu des *clusters*.

Enfin, à l'instar de l'évolution du nombre de *clusters*, il semble exister une valeur de  $r$  pour laquelle la proportion d'URL considérées comme du bruit est acceptable, par exemple autour de 20 %. Reste à placer `N_clusters` en regard de `frac_noise` et à vérifier que ce *clustering* est satisfaisant dans une optique de ciblage...

#### 4.4. Limites de la méthode

Un écueil n'a cependant pu être évité lors de la réalisation de ces différents essais de *clustering*. Le lecteur averti aura en effet remarqué que le temps d'exécution est fourni pour chaque calcul, et qu'il augmente de manière non linéaire avec le nombre d'URL traitées. Ainsi, si 521 URL peuvent être passées à la moulinette en un peu plus de 15 minutes, il faut près d'1 h pour en traiter le double (521 contre 1041), et il faut presque 9 h 30 pour réaliser un essai de *clustering* sur 3123 URL (!)

La conclusion de ces essais est sans appel : dans une optique de ciblage publicitaire, des centaines de milliers d'URL devront être traitées chaque jour, un temps de calcul aussi long que celui observé ici pour calibrer notre modèle n'est pas envisageable. Il faut donc s'orienter vers une méthode plus adaptée au traitement de grands volumes de données.

### 5. Approche sémantique de l'analyse des URL et segmentation automatique par la méthode des $n$ -grammes

La dernière approche a fait appel à un concept développé pour le traitement de problématiques liées à la théorie de l'information. Ici, l'idée est de classer automatique-



ment les URL (de créer des segments) sans avoir à les décortiquer ou à les diviser en leurs caractères constitutifs. Le modèle  $n$ -grammes s'est révélé plutôt bien adapté.

### 5.1. Le modèle $n$ -grammes

D'une manière générale, un  $n$ -gramme est une séquence de  $n$  éléments construite à partir d'une séquence donnée [1]. On se donne un ensemble de documents  $D = \{d_1, d_2, \dots, d_n\}$  et un ensemble de classes auxquelles on cherche à assigner chaque document  $C = \{c_1, c_2, \dots, c_k\}$ . Pour un document  $d_i$ , la probabilité  $p(c_j|d_i)$  qu'il appartienne à la classe  $c_j$  peut s'exprimer à l'aide du théorème de Bayes par :

$$p(c_j|d_i) = \frac{p(d_i|c_j) \cdot p(c_j)}{p(d_i)}$$

avec  $p(d_i)$  constant pour chaque classe  $c_j$ , et  $p(c_j)$  pouvant être considéré comme la fréquence de la classe  $j$  dans les données d'entraînement, par exemple. Reste à traiter le terme  $p(d_i|c_j)$ ... La réalité n'est en fait pas si complexe. Pour une URL constituée par les caractères  $w_1, w_2, \dots, w_L$ ,  $p(d_i|c_j)$  s'exprime comme :  $p(w_1, w_2, \dots, w_L|c_j)$ , et il s'agit en fait de la **probabilité de voir apparaître la séquence**  $w_1, w_2, \dots, w_L$  **dans la classe**  $c_j$ , qui peut être calculée selon :

$$p(w_1, w_2, \dots, w_L|c_j) = \prod_{i=1}^L p((w_i|w_{i-1}, w_{i-2}, \dots, w_1)|c_j)$$

En fait, on considère la plupart du temps que chaque caractère  $w_i$  est uniquement dépendant des  $n - 1$  caractères précédents, ce qui donne une équation légèrement simplifiée<sup>8</sup> :

$$p(w_1, w_2, \dots, w_L|c_j) = \prod_{i=1}^L p((w_i|w_{i-1}, w_{i-2}, \dots, w_{i-n+1})|c_j)$$

Le modèle  $n$ -gramme est donc la distribution de probabilité de chaque séquence de caractères de longueur  $n$  basée sur l'ensemble des données d'entraînement disponibles. On dit alors que  $p(w_1, w_2, \dots, w_L|c_j)$  est le *le modèle  $n$ -gramme pour la classe  $c_j$* .

### 5.2. Construction du modèle $n$ -grammes à partir des données d'URL disponibles

### 5.3. Résultats de segmentation d'URL

---

8. Ici, le lecteur pourra remarquer que si cette approximation peut être plausible lorsqu'on travaille avec des documents textes où les  $w_i$  représentent des mots, il peut en aller tout autrement lorsque l'on travaille avec des URL, où l'on observe parfois aucune structure syntaxique

## Références

- [1] Tarek Amr Abdallah and Beatriz De La Iglesia. Url-based web page classification : A new method for url-based web page classification using n-gram language models. *KDIR 2014 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, 2014.
- [2] Andrea Morichetta et al. Clue : Clustering for mining web urls. *Politecnico de Milano*, 2016.
- [3] Tao Lei et al. A pattern tree-based approach to learning url normalization rules. *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, 2010.