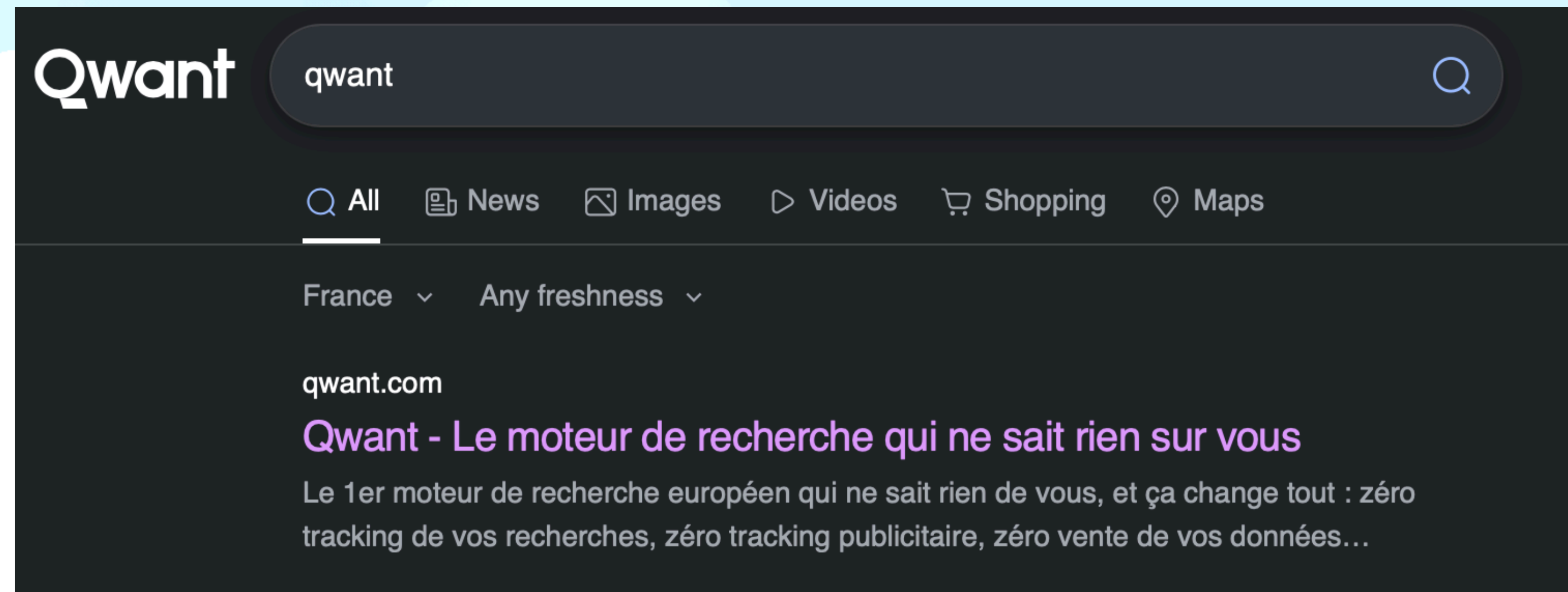


# **Indexation Web**

**Janvier 2023**

# Qwant



- Lara Perinetti  
[l.perinetti@protonmail.com](mailto:l.perinetti@protonmail.com)  
Data Scientist NLP | IR

# Parlez-moi de vous

- Combien vous êtes dans chaque groupe ?
- Quels langages de programmation vous connaissez ?
- A quel point vous êtes familier avec les notions du web ? (HTML, robots.txt etc.)
- Avez-vous déjà étudié le NLP ? IR ?

# Cours Indexation Web

- 6h CM en commun
- 3 \* 3h TP
  - Crawl
  - TF-IDF
  - Query Expansion



# Cours Indexation Web

## Rendus des TPs

- A chaque fin de TP il faudra m'envoyer le code et vous aurez une semaine de plus si vous n'avez pas fini pour me (re) envoyer le code amélioré Vous aurez une semaine de plus pour m'envoyer la version définitive
- TP Crawler - Fonctionnaires 10/01 -> 17/01  
- Ingénieurs 19/01 -> 26/01
- TP Index - Fonctionnaires 20/01 -> 27/01  
- Ingénieurs 23/01 -> 02/02
- TP Query - Fonctionnaires 24/01 -> 03/02  
- Ingénieurs 24/01 -> 09/02

# Cours Indexation Web

- Notation
  - Il y aura un mini projet + des features à faire en plus en points bonus
  - Une partie de la note du TP sera sur les commentaires et la simplicité de compréhension et de lancement du code
  - Il faudra toujours m'envoyer votre code avec un fichier main.py et un README.md avec comment lancer le programme + les paramètres par défaut
- Discord du cours: <https://discord.gg/nPrvscBV>

# Table of content

## Partie 1: Constitution d'un index web

1. Architecture d'un moteur de recherche
2. Crawler le web
3. Document Processing

## Partie 2: Traitement de la requête et ordonnancement des résultats

1. Index
2. Compréhension de la requête
3. Ordonnancement des résultats et évaluation



# Qu'est-ce qu'un moteur de recherche?

## Differents types

- **Web Search:** Google, Qwant, DuckDuckGo, Ecosia etc.
  - Differents formats: HTML, pdf, docx etc.
  - Champs les plus communs: title, content, page rank
- **Shopping Search:** Amazon, Cdiscount, La Redoute etc.
  - Leurs propres structures de données, appliquer des filtres
  - Champs les plus communs: nom du produit, prix, image
- **Desktop search:** sur les ordinateurs personnels
  - fichiers, emails, pages webs dans l'historiques des recherches
  - Séparer les sources de données



The image is a screenshot of a Qwant search engine results page for the query "canal +". The interface is dark-themed. At the top left is the Qwant logo. The search bar contains "canal +" with a clear (X) and search (Q) icon. Below the search bar are navigation tabs: All (selected), News, Images, Videos, Shopping, and Maps. Below the tabs are filters for "France" and "Any freshness".

The search results are divided into two main sections. The first section, highlighted with a red border, contains a paid advertisement from "boutique.canalplus.com" for "Série Limitée - CANAL+, DISNEY+ & PARAMOUNT+". The ad text describes a 12-month subscription for 25,99€/mois. The second section, highlighted with a blue border, contains an organic result from "canalplus.com" for "myCANAL : tv, sports, séries, films en streaming en direct live ou...". Below this is a Wikipedia result for "MyCanal".

On the right side, there is a detailed information panel for "Canal+" highlighted with a yellow border. It includes the description "Chaîne de télévision française", a paragraph about its history, a link to "Read more on Wikipedia", and key facts: "Pays : France", "Site Web : www.canalplus.com", and "Propriétaire : Vivendi". At the bottom of this panel is a link to "Wikipedia.org · How to contribute?" and a "Share your feedback" button.

On the far left, there are four colored boxes with labels: a green box for "SERP Search Engine Results Pages", a red box for "Paid Results", a blue box for "Organic Results", and a yellow box for "Search Engine Features".

## Paid Results

## Search Engine Features

# Information Retrieval

- *“Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information.”* (Salton, 1968)
- La recherche d'information consiste à trouver des documents non structurés qui répondent à un besoin d'information dans une grande collection de documents.
- Recommendation systems (Netflix, Spotify) —> ne seront pas traités dans ce cours
- Deux problématiques majeures
  - De nombreux facteurs influencent la décision d'une personne concernant ce qui est pertinent —> évaluation subjective
  - Les requêtes par mot-clé sont souvent de mauvaises descriptions des besoins réels

# Information Retrieval

## Qu'est-ce qu'un document web?

- On va se concentrer sur les pages HTML

Balises les plus communes:

```
<head>
  <title></title>
  <link></link>
</head>
<body>
  <p></p>
  <div></div>
  <h1></h1>
</body>
```

### Exemple

- La structure et le nom des balises sont reglementés
- Peuvent contenir du texte, des films, des images, ou tout à la fois
- Dans le web, les pages peuvent avoir plusieurs sujets/topics que l'on va essayer d'extraire
- Sont augmentées de métadonnées
  - > Les métadonnées sont des informations sur un document qui ne font pas partie du contenu du texte, comme le type de document



# Information Retrieval

## Qu'est-ce qu'une requête web ?

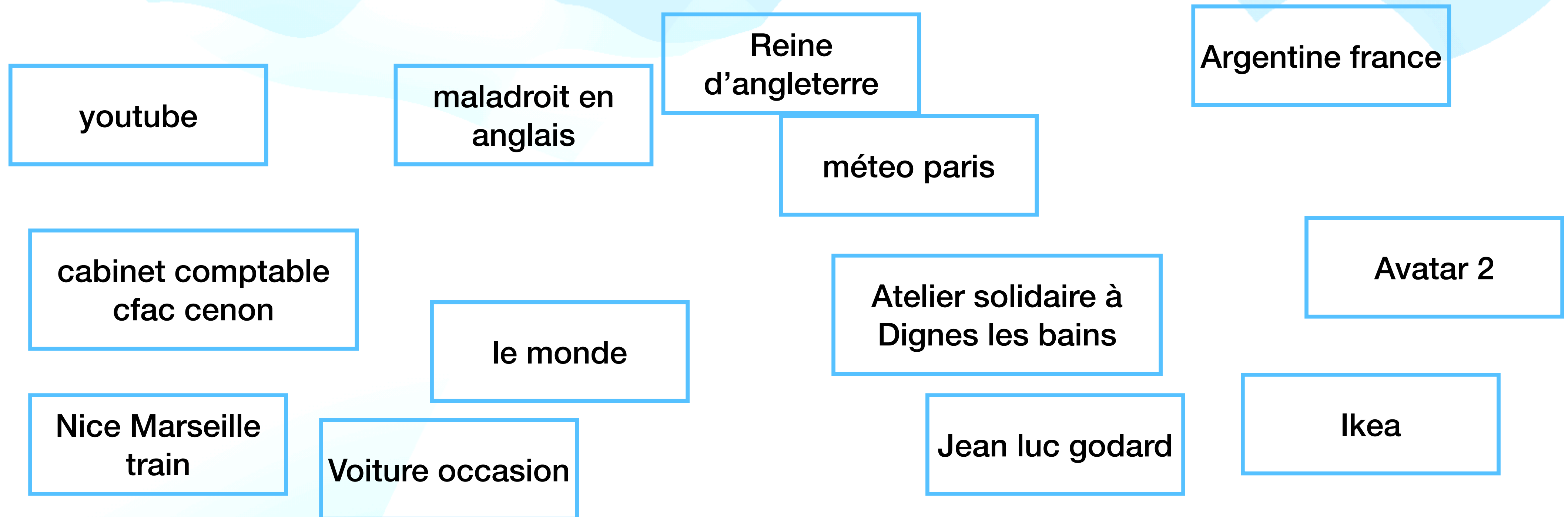
- Spécificités d'une requête web: souvent courte (un ou deux termes) et ambiguë
- On définit 3 intentions principales pour une requête web
  - Navigationnelle
  - Transactionnelle
  - Informationnelle
- On peut définir plusieurs topics pour une requête web
  - Exemple:  
requête: Jaguar  
topics: animal / voiture



# Information Retrieval

## What is a web query ?

Exemples de requetes d'utilisateurs Qwant en décembre 2022

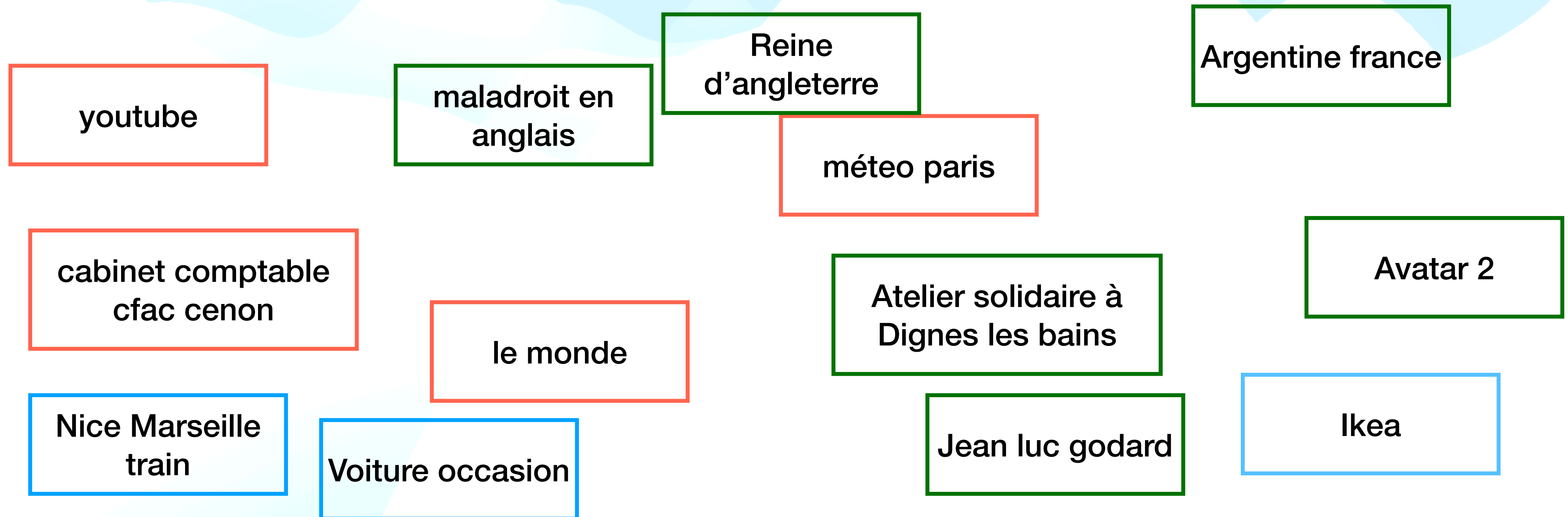


Les classer par intention (informationnelle, transactionnelle, navigationnelle)

# Information Retrieval

## What is a web query ?

Requetes d'utilisateurs Qwant en décembre 2022

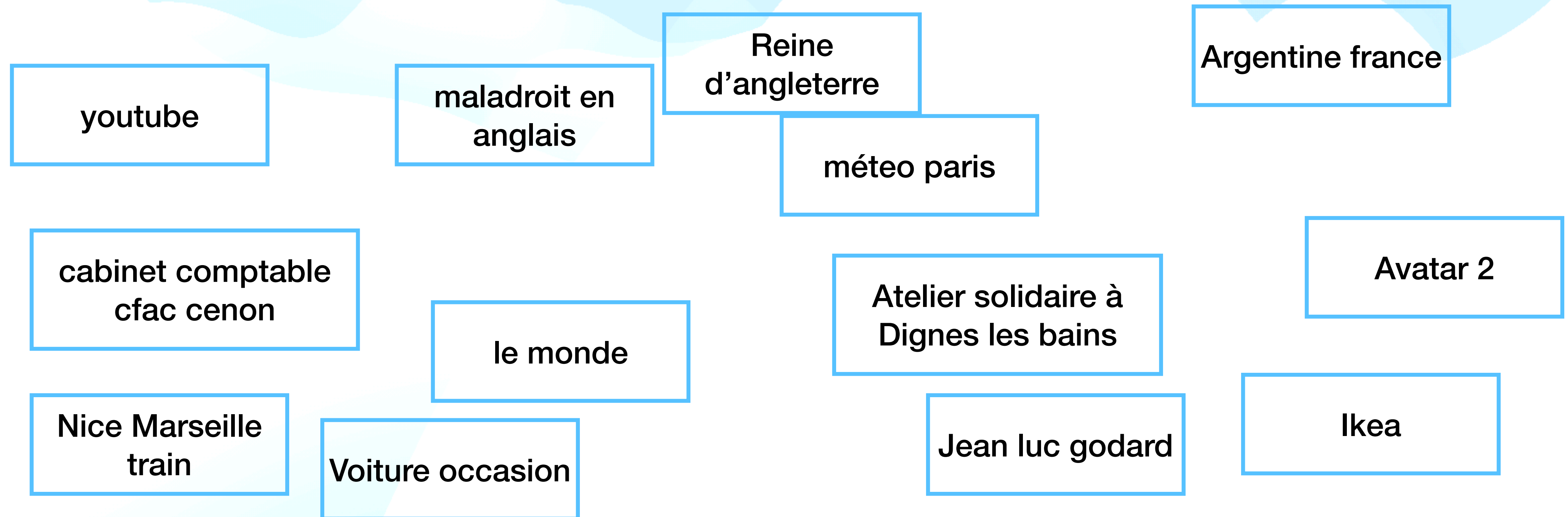


Les classer par intention (informationnelle, transactionnelle, navigationnelle)

# Information Retrieval

## What is a web query ?

Requêtes d'utilisateurs Qwant en décembre 2022



Les classer par topics (pas plus de deux topics par requête)

# Information Retrieval

## What is a web query ?

Requêtes d'utilisateurs Qwant en décembre 2022

youtube

maladroit en  
anglais

Reine  
d'angleterre

Argentine france

météo paris

cabinet comptable  
cfac cenon

le monde

Atelier solidaire à  
Dignes les bains

Avatar 2

Nice Marseille  
train

Voiture occasion

Jean luc godard

Ikea

shopping, news, entertainment, sports, etc.



# **Partie 1**

# **Constitution d'un index web**

Architecture d'un moteur de recherche

# Architecture d'un moteur de recherche

## Objectifs

- **Qualité** : Nous voulons être en mesure de récupérer l'ensemble de documents les plus pertinents possibles pour une requête donnée.
- **Rapidité** : Nous voulons traiter les requêtes des utilisateurs aussi rapidement que possible.

# Architecture d'un moteur de recherche

## Des pages webs à l'index

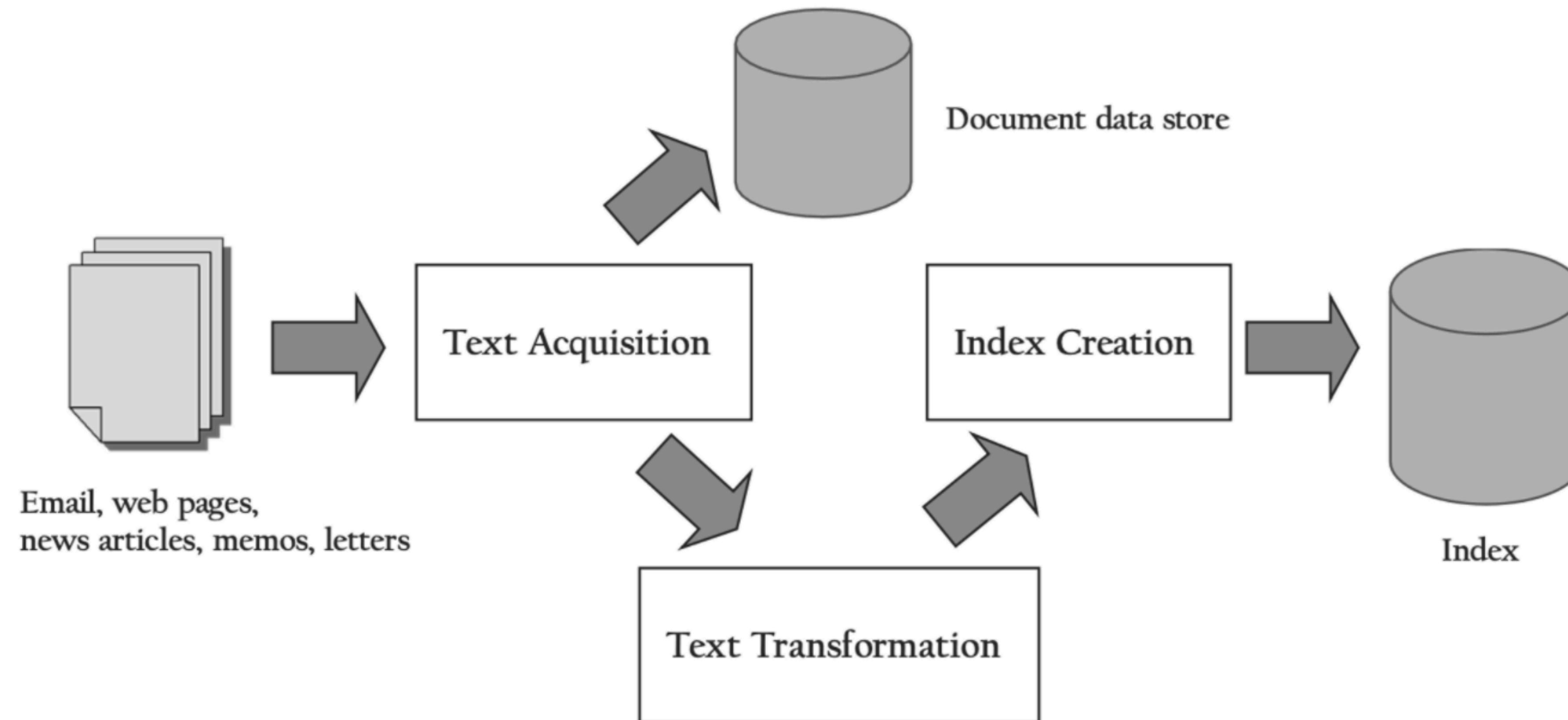


Fig. 2.1. The indexing process

# Des pages webs à l'index

## Text Acquisition

- identifie et rend disponible les documents que l'on voudra chercher

### 1. Quels documents ?

- partir d'une collection de documents clés en main
- chercher ces documents via une tâche de **crawl**

### 2. Où les stocker?

Une **base de données relationnelle** peut être utilisée pour stocker les documents et leurs métadonnées.

### 3. La notion de crawler

- Un crawler est conçu pour suivre les liens sur les pages web pour découvrir et télécharger de nouvelles pages
- Un bon crawler doit être capable de gérer efficacement l'énorme volume de nouvelles pages sur le Web, tout en veillant à ce que les pages qui ont pu être modifiées depuis la dernière visite d'un crawler restent "fraîches" pour le moteur de recherche



# Des pages webs à l'index

## Text Transformation

- Parsing —> hierarchy extraction
- Stopping —> removing stop words
- Stemming / Lemmatization
- Link extraction / analysis
- Information extraction (language, topic etc.)

# Des pages webs à l'index

## Index creation

- Prend la sortie de la transformation de texte pour créer les index qui contiendront les informations sur les documents
- Compte tenu du grand nombre de documents, la création d'un index doit être efficace, tant en termes de temps que d'espace.

# Architecture d'un moteur de recherche

## De la requête aux résultats finaux

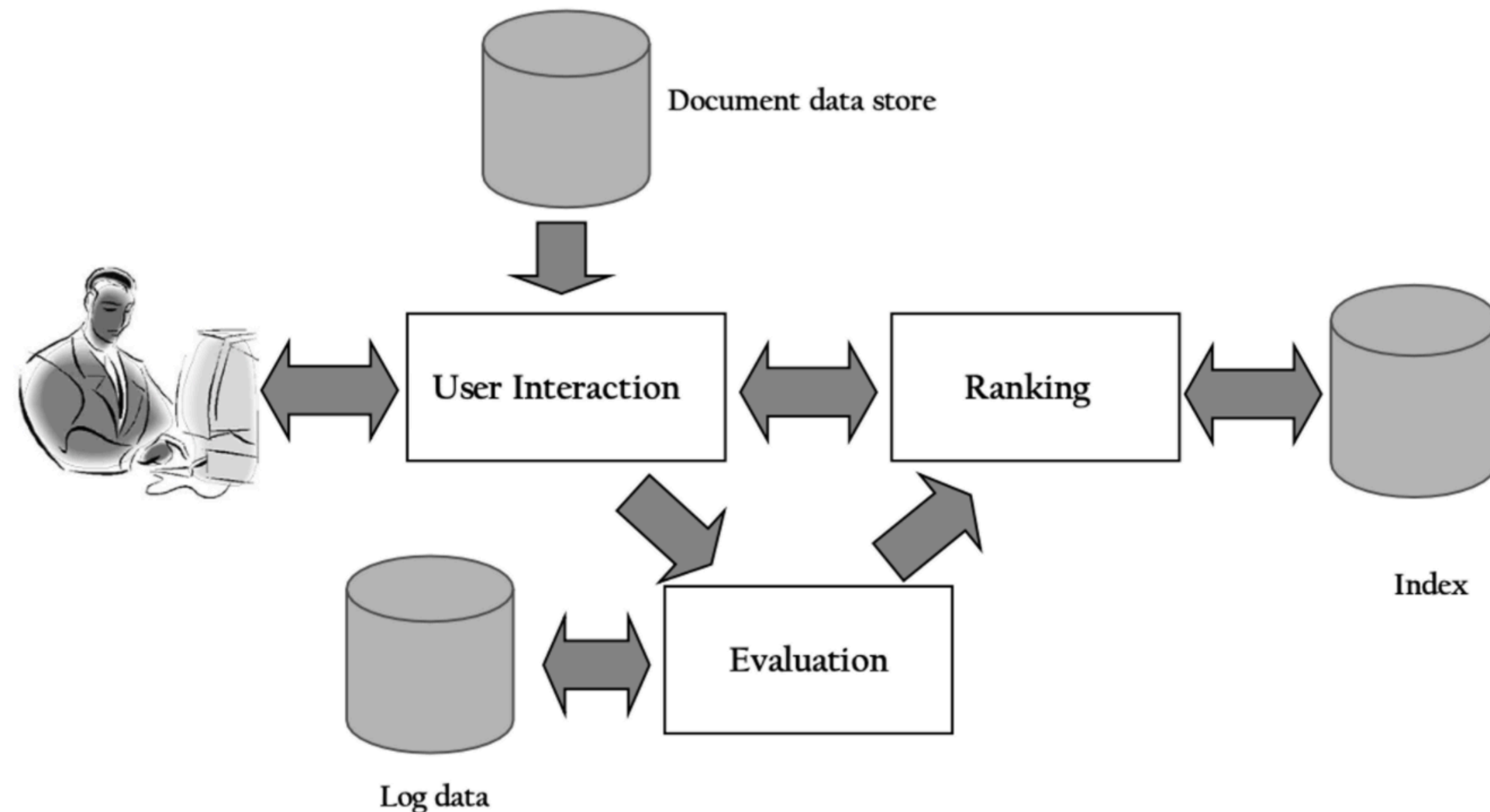


Fig. 2.2. The query process

# De la requête aux résultats finaux

## User interaction

- Reçoit en input la requête de l'utilisateur et la transformer pour qu'elle soit comprise par l'index
- La requête de l'utilisateur peut comprendre le texte meme de la requête mais aussi toutes les métadonnées associées (langue de l'utilisateur, topics de la requête, heure, jour de la semaine etc.)
- Inclut la transformation de la requête
  - Spell checking
  - Augmentation de la requête
  -



# De la requête aux résultats finaux

## Ranking

- Reçoit en input les documents qui ont survécu au filtre de la requête et leur donne un score en fonction de la requête.
- L'ordonnancement doit être à la fois efficace, puisque de nombreuses requêtes doivent être traitées en peu de temps, et effectif, puisque la qualité du classement détermine si le moteur de recherche atteint l'objectif de trouver des informations pertinentes.

# De la requête aux résultats finaux

## Evaluation

- Evaluation “offline”
- Evaluation “online”

# **Partie 1**

# **Constitution d'un**

# **index web**

Crawler le web

# Crawler le web

## Décider de ce qu'il faut rechercher

- Le Web est immense et en constante expansion
- Les pages Web changent constamment  
Même les documents utiles peuvent devenir moins utiles avec le temps.
- On ne peut pas se permettre de stocker toutes les pages du web
- On évalue un crawler par la **couverture (coverage)** qui mesure la quantité de documents pertinents que l'on réussit à récupérer  
Et par la **fraicheur (freshness)** des pages stockées.
- Un crawler se doit d'être poli, il se doit d'espacer les crawl pour une url donnée



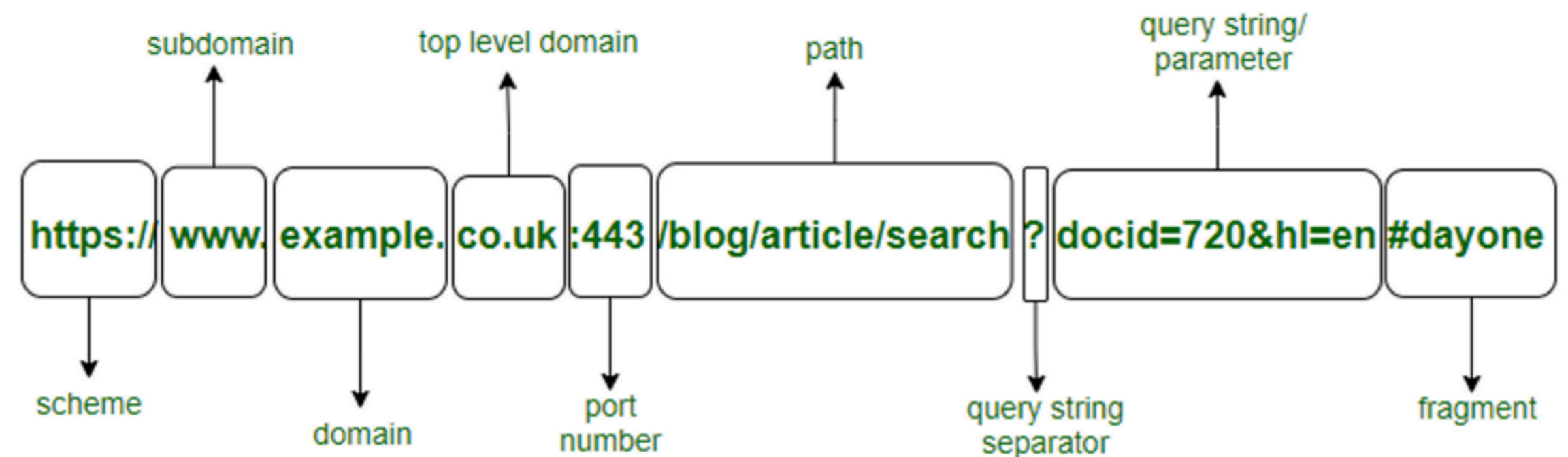
# Crawler le web

## Qu'est-ce qu'une page web ?

- Scheme  
http | https | ftp | smtp
- Subdomain  
indique le type de ressource  
www | blog | audio etc.
- Domain  
indique l'organisation
- TLD  
indique le type d'organisation à laquelle le site web est enregistré.  
pays: fr | co.uk | de  
region: re  
autre: com | org | net

### Parts of a URL

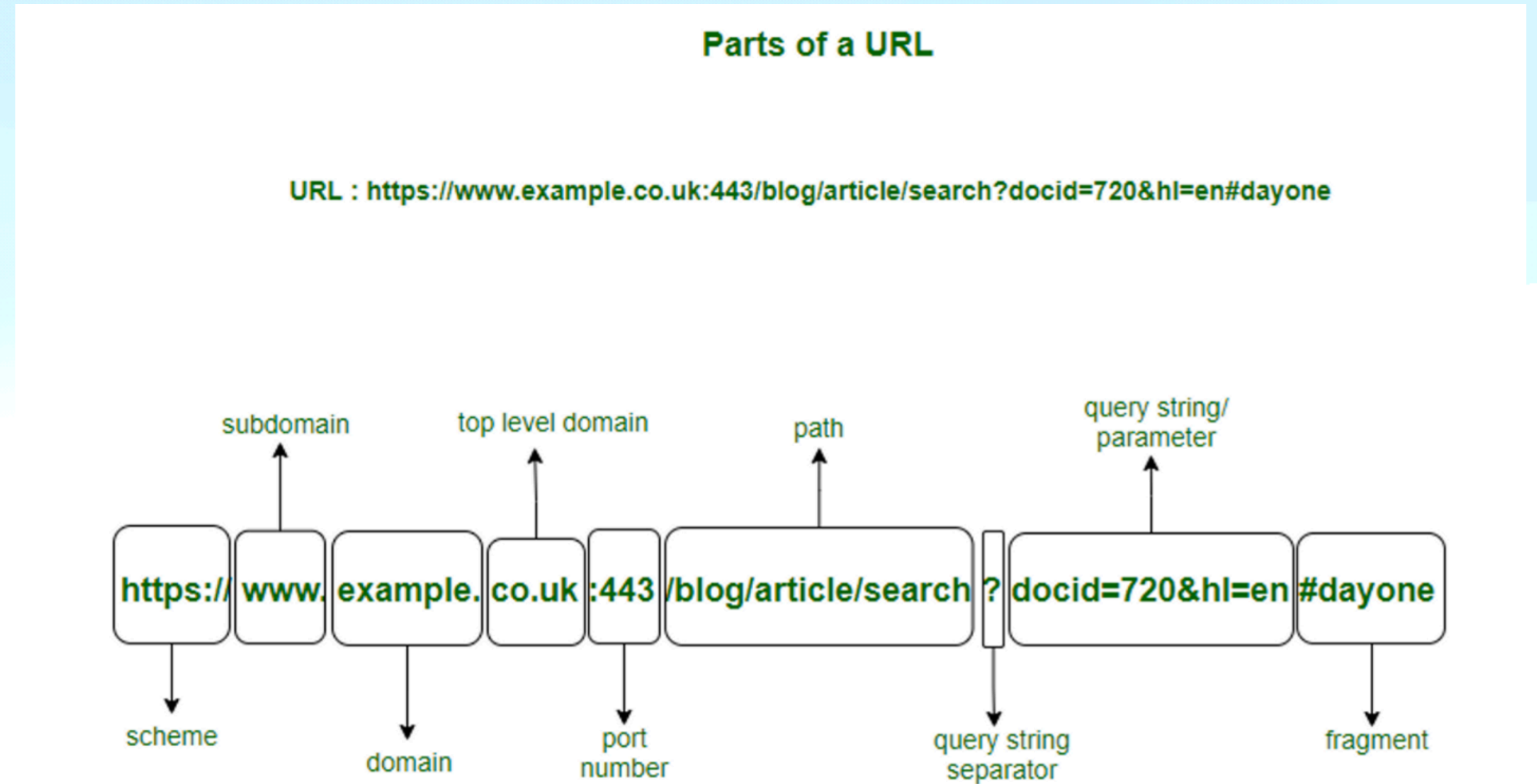
URL : <https://www.example.co.uk:443/blog/article/search?docid=720&hl=en#dayone>



# Crawler le web

## Qu'est-ce qu'une page web ?

- Path  
indique la localisation de la page dans son domaine
- Query  
les paramètres d'un site web.
- Fragment  
pour arriver directement à une certaine balise dans le document html





# Crawler le web

## Algorithme

- Input: un ensemble d'urls
- Les urls sont ajoutées à une **file d'attente (ou frontier)** de requêtes URL
- Le crawler commence à récupérer les pages de la file d'attente des requêtes
- Les pages téléchargées sont analysées pour trouver des balises de liens qui pourraient contenir d'autres URL utiles à récupérer.
- Si le crawler trouve une nouvelle URL qu'il n'a pas vue auparavant, elle est ajoutée à la frontier.
- Le robot continue jusqu'à ce qu'il n'y ait plus de nouvelles URL à ajouter à la file d'attente ou que le disque soit plein.
- La frontier peut être une file d'attente standard ou être ordonnée de manière à ce que les pages importantes soient placées en tête de liste.

# Crawler le web

## Controler le crawler: robots.txt

- Les propriétaires de sites Web utilisent le fichier /robots.txt pour donner des instructions sur leur site aux crawlers
- Exemples de robots.txt
  - Site important
  - Site de niche
- Guidelines de google
- Informations sur le crawler de Qwant



# Crawler le web

## La notion de fraîcheur

- Il n'est pas possible de vérifier constamment la fraîcheur de toutes les pages
- On choisit donc de vérifier les pages importantes et celles qui changent fréquemment
- La fraîcheur est la proportion des pages explorées qui sont actuellement fraîches.  
Pb: si une page se met à jour toutes les 2 secondes

# Crawler le web

## La notion de fraîcheur

- Il n'est pas possible de vérifier constamment la fraîcheur de toutes les pages
- On choisit donc de vérifier les pages importantes et celles qui changent fréquemment
- La **fraîcheur** est la proportion des pages explorées qui sont actuellement fraîches.  
Pb: si une page se met à jour toutes les 2 secondes
  - La meilleure stratégie est alors d'arrêter de la crawler
- C'est pour cela que l'on va plutôt vouloir calculer l'âge d'une page

# Crawler le web

## La notion de fraîcheur

- Une page est **fraîche** si elle représente la copie la plus récente d'une page Web, et périmée dans le cas contraire.  
La fraîcheur est la fraction des pages crawlées qui sont actuellement fraîches.
- Les pages deviennent tout de suite fraîches quand elles sont crawlées, mais dès que la page change, la page explorée devient périmée.
- La page a un **âge** de 0 jusqu'à ce qu'elle soit modifiée, puis son âge augmente jusqu'à ce que la page soit à nouveau explorée.
- Optimiser l'âge ne nous fera jamais dire qu'il vaut mieux arrêter de crawler une page



# Crawler le web

## Sitemaps

- Certains sites web exposent une sitemap aux crawler dans le robots.txt
- Exemple
- Quel est l'intérêt de créer cette sitemap pour l'administrateur du site?



# Crawler le web

## Sitemaps

- Certains sites web exposent une **sitemap** aux crawler dans le robots.txt
- Exemple
- Quel est l'intérêt de créer cette sitemap pour l'administrateur du site?
  - Il indique aux moteurs de recherche des pages qu'ils ne trouveraient pas autrement.
  - Cela permet de réduire le nombre de requêtes que le crawler envoie à un site web sans sacrifier la fraîcheur des pages.

# Crawler le web

## Stocker les documents

- La recherche de documents est couteuse en termes de CPU et de réseau.
- On va vouloir conserver une copie des documents déjà crawlés au lieu d'essayer de les récupérer à chaque fois qu'on lance un tour de crawl.
- En général on utilise une BDD relationnelle avec pour identifiant unique du document un hash de l'URL
- Exemple de BDD: Big Table

# Crawler le web

## Détecter les documents dupliqués

- Tache plutôt simple
- On va souvent utiliser des techniques qui calculent les bytes des documents
- Baseline: **checksum technique**:  
T r o p i c a l f i s h  
54 72 6F 70 69 63 61 6C 20 66 69 73 68      result = 508
- **cyclic redundancy check**, une autre technique qui prend en compte la position des bytes.
- Attention: <http://monsite.com> != <https://monsite.com>



# Crawler le web

## Détecter les documents “presque” dupliqués (ou near duplicates)

- Tache plus complexe
- Pour une url donnée, on va chercher à trouver tous ses “presque” duplicats dans la BDD.
- Pour cela, il faut représenter les documents
  - Soit en le hashant (e.g: sim-hash)
  - Soit avec des embeddings (vecteur dense de représentation d'un document)



# Crawler le web

## Le deep web

- Le deep web représente toutes les pages web qui sont compliquées à trouver pour un crawler et qui sont bien souvent non indexées par les moteurs de recherche
- Ces urls sont bien plus nombreuses que les pages web conventionnelles  
Et elles sont légales
- Contrairement aux pages du dark web

# **Partie 1**

# **Constitution d'un**

# **index web**

Document Processing

# Document Processing

## Extraire des informations des documents

- Avant d'ajouter nos documents dans un index, on va vouloir en extraire le plus d'informations possibles qui nous aideront à répondre de la manière la plus pertinente
  - Text Processing
  - Structure du document
  - Analyse des liens
  - Document embeddings

# Text processing



# Document Processing

## Text Processing | Vocabulaire

- Quelques mots apparaissent très souvent, beaucoup de mots n'apparaissent presque jamais.
- La taille du vocabulaire augmente avec le corpus  
Mais il y a de moins en moins de mots nouveaux lorsque le corpus est déjà important
- Le vocabulaire est constitué de
  - mots bien orthographiés (dans différentes langues)
  - mots avec fautes d'orthographe
  - mots inventés (e.g: noms de produits, de sociétés, etc.)
  - adresses mails, suites de chiffres
  - etc.

# Document Processing

## Text Processing | Des mots aux tokens

- On ne va plus parler de mots mais de token
- On va subdiviser le texte en plusieurs tokens
- Une tokenization par langue:
  - En anglais ou en français, plutôt simple:  
“The apple is red” —> [“the”, “apple”, “is”, “red”]
  - Les langues CJK ont une tokenization adapté aux signes.
- Il doit y avoir la meme tokenization du côté du document que du côté de la requête
- Une étape clé dans le text processing, si la tokenization est mal faite, les étapes d’après vont en pâtir. Elle est tellement important qu’aujourd’hui on utilise souvent un modèle entraîné pour tokenizer.
- On parle aussi de tokenization pour diviser les documents en phrases, en characters, en morphemes ou “word-pieces”

# Document Processing

## Text Processing | Stemming et lemmatisation

- En fonction des langues, on peut avoir besoin de faire référence à une représentation simplifiée des tokens, pour des questions de match avec la requête ou de taille du vocabulaire
- Documents: “nous allons à la boulangerie”

Token original	Token stemmé	Token Lemmatisé
allons	all	aller
- En fonction des langues peut etre plus ou moins intéressant:
  - Les langues CKJ n’ont pas de variation morphologique
  - Les langues slaves ont parfois plus de 7 déclinaisons possibles pour un meme nom commun



# Structure du document



- Tache: Boilerplate removal
- De nombreuses pages web contiennent du texte, des liens et des images qui ne sont pas directement liés au contenu principal de la page.
- Ces éléments supplémentaires sont pour la plupart du bruit et peuvent avoir un impact négatif sur le classement de la page.

WIKIPÉDIA

L'encyclopédie libre

Rechercher sur Wikipédia

Sommaire [masquer]

Début

Dénominations

> Évolution du langage

> Description de HTML

Interopérabilité de HTML

Notes et références

> Voir aussi

ArticleDiscussion

HTML

*HyperText Markup Language*

Caractéristiques	
<b>Extensions</b>	.html , .htm <span></span>
<b>Type MIME</b>	text/html <span></span>
<b>PUID</b>	fmt/99 <span></span>
<b>Développé par</b>	World Wide Web Consortium & WHATWG <span></span>
<b>Version initiale</b>	1993 <span></span>
<b>Type de format</b>	Langage de balisage <span></span>
<b>Basé sur</b>	Standard Generalized Markup Language <span></span>
<b>Origine de</b>	XHTML <span></span>
<b>Norme</b>	ISO/IEC 15445 <span></span> <div>W3C HTML 4.01 <span></span></div> <div>W3C HTML5 <span></span></div>
<b>ISO</b>	15445 <span></span>
<b>Spécification</b>	Format ouvert <span></span>
<b>Site web</b>	html.spec.whatwg.org/multipage <span></span> <span></span>
<span>modifier - modifier le code - modifier Wikidata</span> <span></span>	

Le ***HyperText Markup Language***, généralement abrégé **HTML** ou, dans sa dernière version, **HTML5**, est le **langage de balisage** conçu pour représenter les **pages web**.

Ce langage permet d'écrire de l'**hypertexte** (d'où son nom), de structurer **sémantiquement** une page web, de mettre en forme du contenu, de créer des formulaires de saisie ou encore d'inclure des **ressources multimédias** dont des **images**, des **vidéos**, et des **programmes informatiques**. L'HTML offre également la possibilité de créer des documents **interopérables** avec des équipements très variés et conformément aux exigences de l'**accessibilité du web**.

Il est souvent utilisé conjointement avec le **langage de programmation JavaScript** et des **feuilles de style en cascade** (CSS). HTML est inspiré du *Standard Generalized Markup Language* (SGML). Il s'agit d'un **format ouvert**.

## Dénominations [ modifier | modifier le code ]

L'**anglais** « *Hyper**t**ext Markup Language* » se traduit littéralement en « langage de balisage d'hypertexte »<sup>1</sup>. On utilise généralement le **sigle** « HTML », parfois même en répétant le mot « langage » comme dans « langage HTML ». *Hypertext* est parfois écrit *HyperText* pour marquer le *T* du sigle HTML.

Le public non averti parle parfois de HTM au lieu de HTML, HTM étant l'**extension de nom de fichier** tronquée à trois lettres, une limitation que l'on trouve sur d'anciens **systèmes d'exploitation** de Microsoft.

## Évolution du langage [ modifier | modifier le code ]

Durant la première moitié des années 1990, avant l'apparition des **technologies Web** comme le langage **JavaScript** (JS), les **feuilles de style en cascade** (CSS) et le *Document Object Model* (DOM), l'évolution de HTML a dicté l'évolution du *World Wide Web*. Depuis la création de l'HTML 4 en 1997, l'évolution de HTML a fortement ralenti ; dix ans plus tard, HTML 4 reste utilisé dans les **pages web**. En 2008, la spécification du **HTML5** est à l'étude<sup>2</sup> et devient d'usage courant dans la seconde moitié des années 2010.

### 1989-1992 : Origine [ modifier | modifier le code ]

HTML est une des trois inventions à la base du *World Wide Web*, avec le *Hyper**t**ext Transfer Protocol* (HTTP) et les **adresses web** (URL). HTML a été inventé pour permettre d'écrire des documents **hypertextuels** liant les différentes ressources d'**Internet** avec des **hyperliens**. Aujourd'hui, ces documents sont appelés « **page web** ». En août 1991, lorsque **Tim Berners-Lee** annonce publiquement le web sur **Usenet**, il ne cite que le langage **Standard Generalized Markup Language** (SGML), mais donne l'**URL** d'un document de suffixe **.html**.

Dans son livre *Weaving the web*<sup>3</sup>, **Tim Berners-Lee** décrit la décision de baser HTML sur SGML comme étant aussi « diplomatique » que technique : techniquement, il trouvait SGML trop complexe, mais il voulait attirer la communauté **hypertexte** qui considèrait que SGML était le langage le plus prometteur pour standardiser le format des documents hypertexte. En outre, SGML était déjà utilisé par son employeur, l'**Organisation européenne pour la recherche nucléaire** (CERN) ;

Les premiers **éléments du langage HTML** comprennent :

- le titre du document,
- les **hyperliens**.

# Document Processing

## Document Structure | Extraire d'autres informations que le contenu principal

- Interet de l'HTML: le nom des balises est codifié.
- Il y a certaines balises qui nous aident à extraire des informations:
  - `<meta>` pour les métadonnées
  - `<img>` pour les images
  - `<title>` pour les titres
  - `<a>` pour les ancrs



# Analyse des liens



# Document Processing

## Analyse des liens

- Le crawler s'aide des liens entre les pages pour trouver de nouvelles pages à crawler
- Ces liens nous permettent
  - de créer un graphe du web
  - de comprendre les relations entre les pages
  - d'ordonner les pages en fonction de leur importance
- Exemple

# Document Processing

## Analyse des liens | Le texte des ancres

- En général, un texte court qui décrit la page vers laquelle on va cliquer  
Il peut avoir les memes caractéristiques qu'une requête web
- Extraire le texte des ancres nous permet ensuite de matcher ce texte avec la requête
- Peut etre très utile pour les abréviations:  
YouTube.com -> yt, ytb
- Mais doit etre nettoyé:  
“Cliquez ici”, “check here”, “découvrir”

# Document Processing

## Analyse des liens | La popularité d'une page

- La majeure partie du web est composée de pages assez peu intéressantes, datées, ou de pages que l'on qualifie de spam
- On va donner un ou plusieurs score de qualité, de confiance aux urls
- Le plus connu: Page Rank



# Document Processing

## Analyse des liens | Page rank

- Algorithme qui calcule un score de popularité pour une url donnée
- Idée:
  - Plus il y a d'urls qui pointent vers une url A, plus A est populaire
  - Tous les liens ne comptent pas de la même manière:  
Si toutes les urls qui pointent vers A ne sont pas populaires, la popularité de A sera faible —> si on prend en compte la popularité des pages qui pointent vers A, on ne donnera pas un gros score aux spams
- Keywords:
  - Page = document = site internet = url
  - Inlinks: liens pointant vers une page web
  - Outlinks: liens d'une page vers les autres pages

# Document Processing

## Analyse des liens | Page rank

- Algorithme “Random surfer model”

```
func random_surfer_model(threshold)  
  r <- random(0,1)
```

```
  if r < threshold
```

```
    choisis une page web random
```

```
  else
```

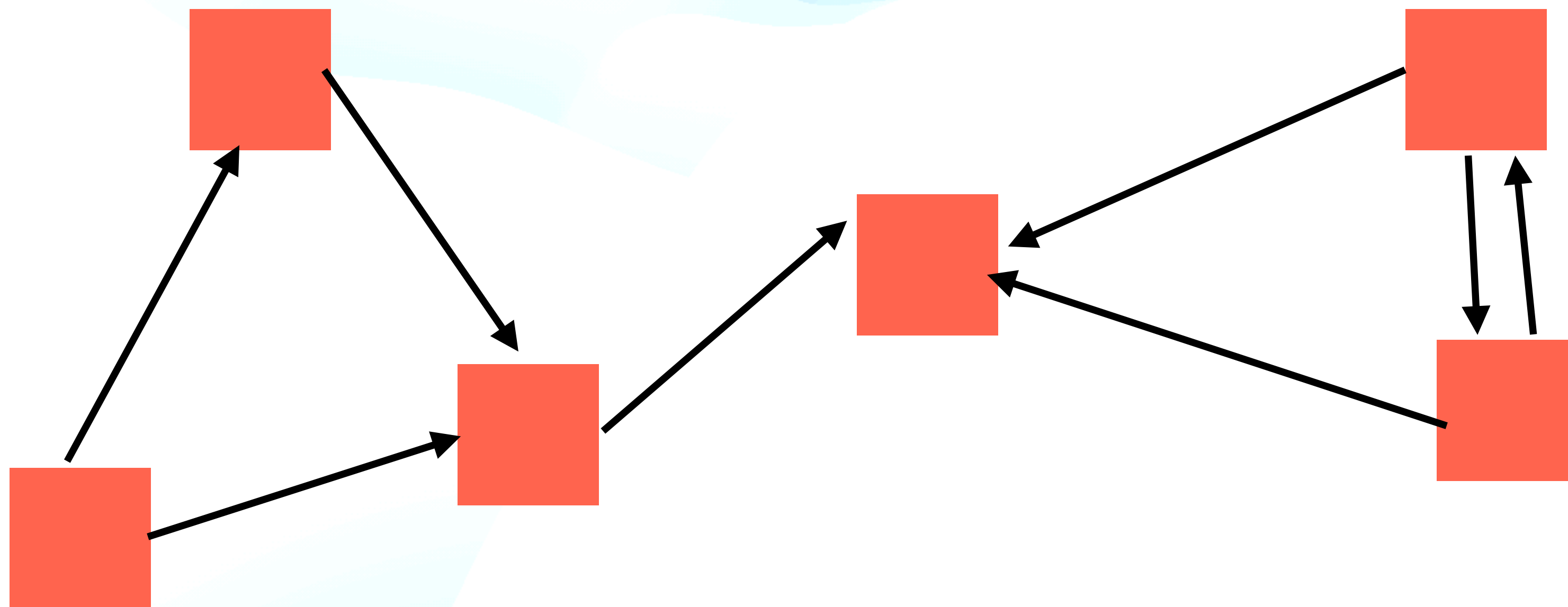
```
    click sur un lien dans la page web courante
```

```
  random_surfer_model(threshold)
```

- En général, la variable threshold est plutôt petite, par exemple 0.15, pour que l’algorithme ait plus de chances de cliquer sur un lien que de choisir une autre page web
- Le score de page rank d’une page est la probabilité que le surfer arrive sur cette page

# Document Processing

## Analyse des liens | Page rank





# Document Processing

## Analyse des liens | Page rank

- L'algorithme tourne "à l'infini" et va donc visiter tous les sites webs plusieurs fois. Mais il est plus probable qu'il visitera un site populaire des milliers de fois plus souvent qu'un site impopulaire.
- Si on ne lui permettait pas de chercher une nouvelle page random (premier if) il resterait bloqué
  - sur des pages qui n'ont pas de liens,
  - des pages dont les liens ne pointent plus vers aucune page,
  - ou des pages qui forment une boucle.

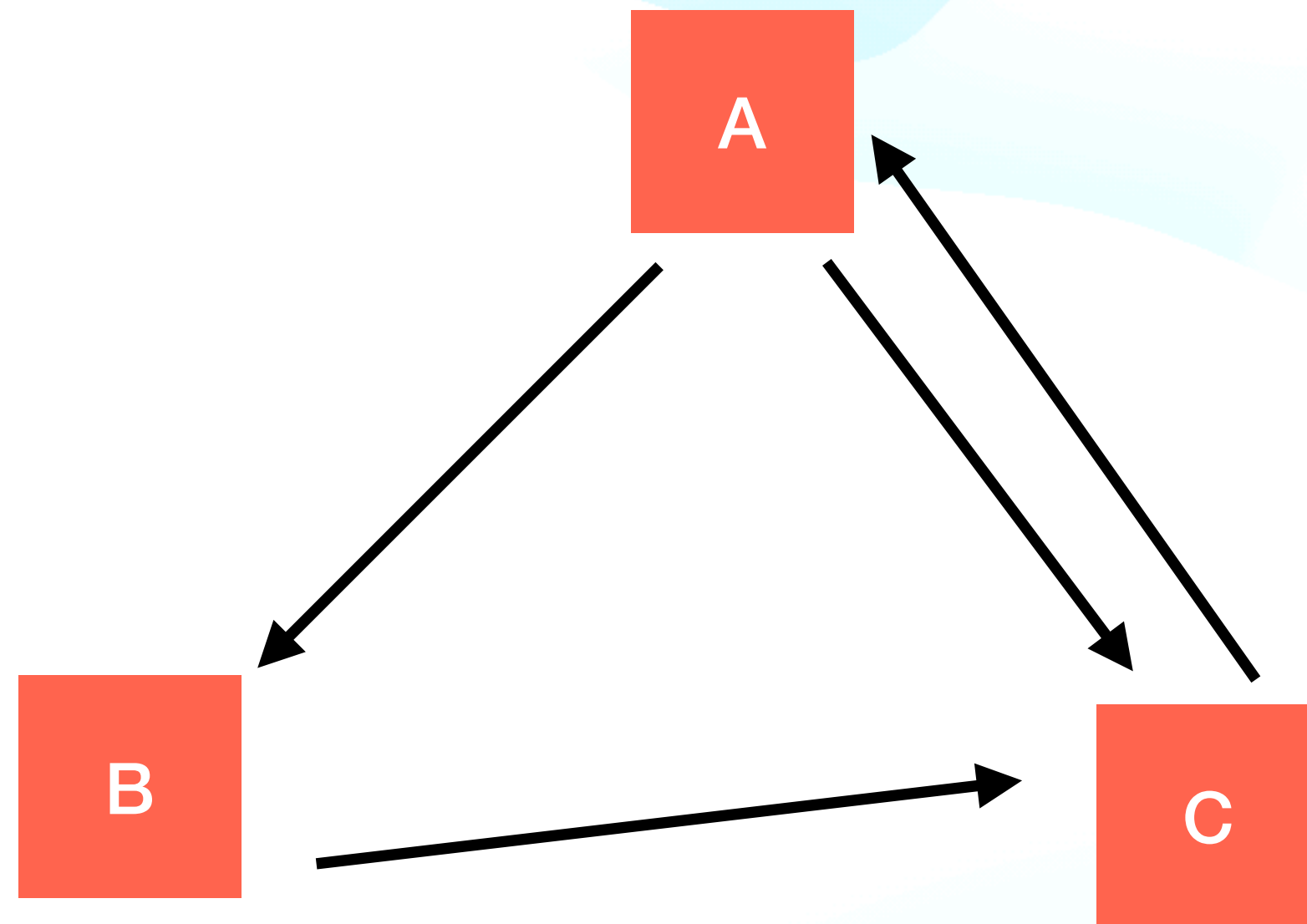
# Document Processing

## Analyse des liens | Page rank

- Au départ toutes les pages ont le meme page rank

# Document Processing

## Analyse des liens | Page rank



- Le page rank d'une page dépend de celui des autres pages
- Plus une page a de liens, moins ses liens ont un poids dans le calcul du page rank

- **Si on ne prend en compte que les inlinks**

Temps 1:

$$PR(C) = PR(A) / 2 + PR(B) / 1 = 0.33 / 2 + 0.33 = 0.5$$

$$PR(A) = PR(C) / 1 = 0.33$$

$$PR(B) = PR(A) / 2 = 0.17$$

Temps 2:

$$PR(C) = 0.33/2 + 0.17 = 0.33$$

$$PR(A) = 0.5$$

$$PR(B) = 0.25$$



# Document Processing

## Analyse des liens | Page rank

- Si on ne prend en compte que les inlinks ET le random sur les pages

$$PR(u) = \frac{\lambda}{N} + (1 - \lambda) \cdot \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

- N -> le nombre de documents
- Lambda -> probabilité de choisir une page random  
1-lambda -> probabilité de cliquer sur un lien
- Bu -> inlinks de l'url u
- Lv -> # outlinks depuis l'url v

# Document Embeddings

# Document Embeddings

## Dense representations | BERT-like embeddings

- Embeddings: représentation vectorielle d'une string (plus ou moins longue)
- BERT-like embeddings: représentation contextualisée au niveau du word-piece  
Exemple:
  - Danone est mon yaourt préféré
  - Je déteste DanoneLe vecteur de Danone sera différente
- Problèmes en recherche d'information:
  - Ce sont des representation assez longues à générer  
Comme elles dependent du contexte, on ne peut pas simplement avoir une hashmap de token -> vecteur
  - Ces embeddings ont une contrainte de taille de la string en input (en général max 512 word pieces), un document web est souvent bien plus long  
On choisit alors
    - soit de représenter le début du document
    - soit de diviser le document en chunks de max 512 wp et de multiplier les vecteurs résultants
    - Soit de partir d'un summary du document



# Document Embeddings

## Dense representations | ColBERT

- Similarité entre la requête et le document
- Un meme modele qui encode la requête et le document séparément et donne un score de similarité entre un document et une requête
- On va encoder le document au moment de l'indexation  
On n'aura plus qu'à encoder la requête

# **Partie 2**

## **Traitement de la requête et ordonnancement des résultats**

Index