

FRI : Fondements en Recherche d'Information

Cours 1 : Introduction - Indexation - Modèles de RI

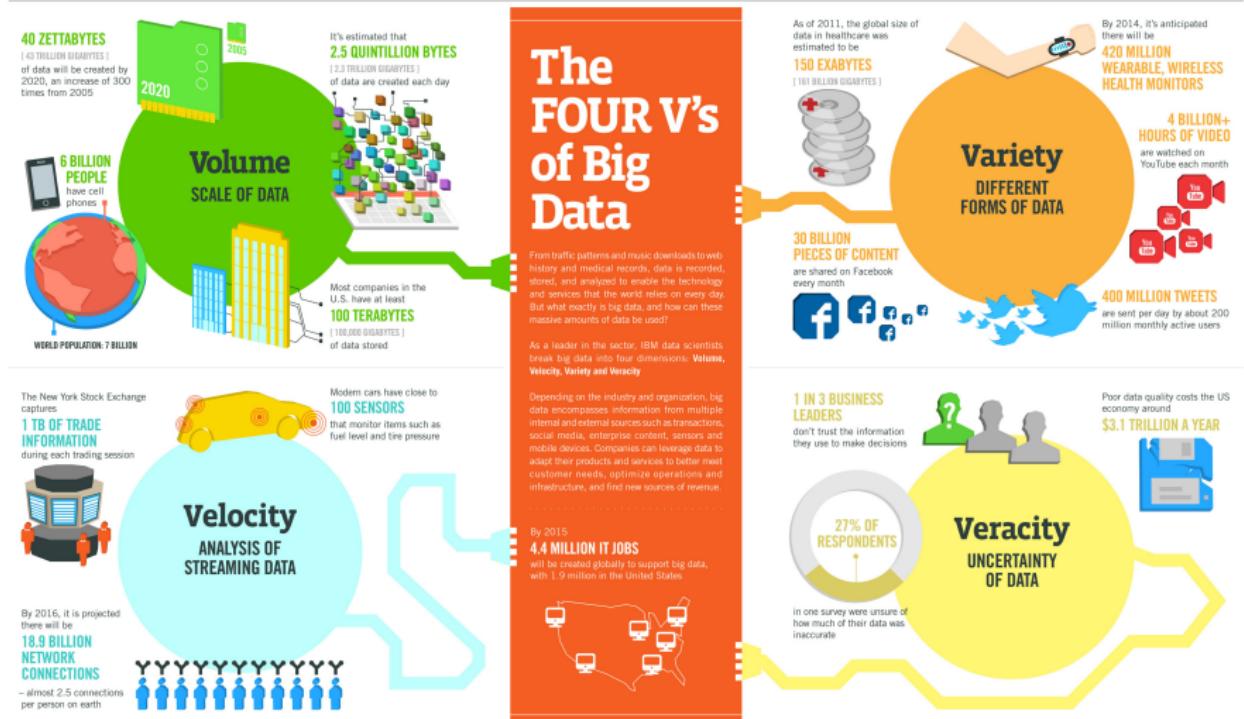
Céline Hudelot, Maître de conférences, Centrale Supelec

2015-2016

Plan

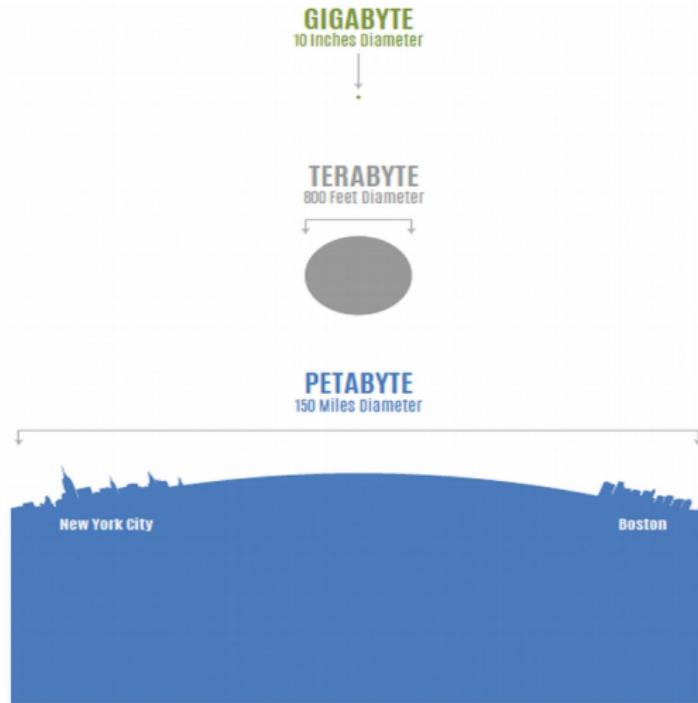
- 1 Contexte
- 2 Introduction
- 3 Historique des SRI
- 4 Indexation
 - Des documents aux termes du dictionnaire
 - Lois de base en RI
 - Représentation d'un document et de la collection - Index inversé
- 5 Modèles classiques de recherche d'Information
 - Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes
 - Modèle vectoriel
- 6 Bilan

Contexte : Explosion des données : *Big Data*



Source : IBM

Big Data : Volume



Source : The Executive's guide to big data & Apache Hadoop

Big Data : Volume et Variété

Database Size	Common Characteristics
1 gigabyte	<ul style="list-style-type: none">...> Information generated by traditional enterprise applications...> Typically consists of transactional data, stored in relational databases...> Uses Structured Query Language (SQL) as the access method
1 terabyte	<ul style="list-style-type: none">...> Standard size for data warehouses...> Often aggregated from multiple databases in the 1-100 gigabyte range...> Drives enterprise analytics and business intelligence
1 petabyte	<ul style="list-style-type: none">...> Frequently populated by mass data collection – often automated...> Regularly contains unstructured information...> Serves as a catalyst for exploring new Big Data-related technologies

Source : The Executive's guide to big data & Apache Hadoop

Big Data : Variété

Données structurées vs non-structurées

Données structurées

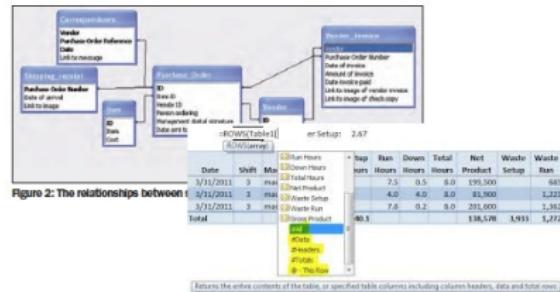
Données avec un degré d'organisation permettant la recherche rapide d'information et de connaissance.

- ex : SGBD, feuilles de calcul, ...

Données non-structurées

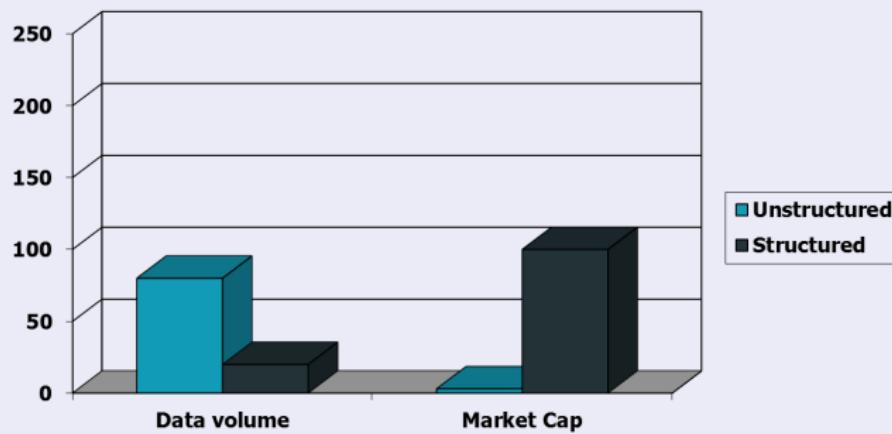
Données sans structuration (forte)

- ex : emails, documents, images, données issues des réseaux sociaux, des objets connectés...



Big Data : Variété

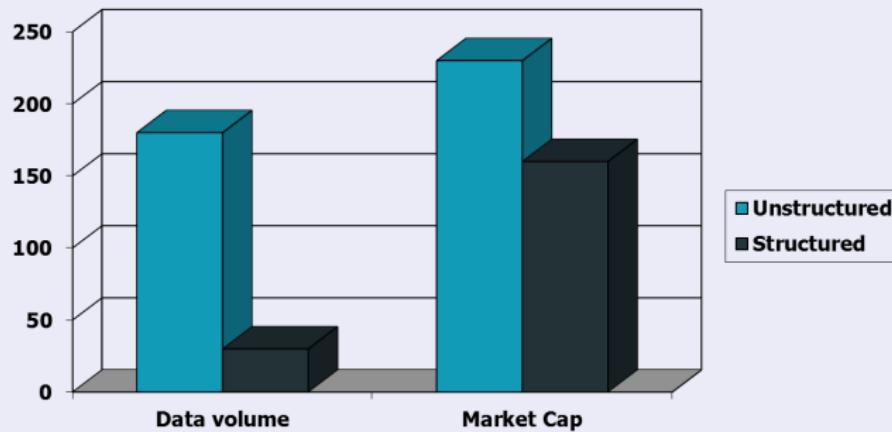
Données structurées (bases de données) vs non-structurées (texte) dans les années 90



Source : C. Manning

Big Data : Variété

Données structurées (bases de données) vs non-structurées (texte) aujourd'hui (2005)



Source : C. Manning

Big Data : Variété

Données non-structurées : quelques chiffres

- 175 millions de tweets chaque jour.
- 571 nouveaux sites chaque minute.
- 2.5 quintillions bytes de données non-structurées par jour.

Source : <http://www.digitalreasoning.com/resources/Holistic-Analytics.pdf>

Big Data : Vélocité

Des flux de données importants - Des temps de traitements courts

Comparing High-Velocity Data & Big Data

High-Velocity Data

- Real-Time
- Performance & Volume Challenges
- Use Cases: Operations & Analytics

Big Data

- Batch Process
- Volume Challenge
- Use Case: Analytics



Source : ScaledB

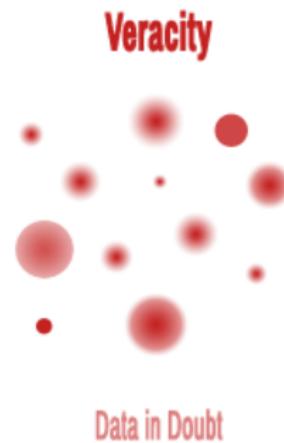
Big Data : Vélocité



Source : Go Globe

Big Data : Véracité

Confiance aux données pour une prise de décision.



Source : Halvena.net

Recherche d'information

La recherche d'information (*Information retrieval*)

- Gerard Salton, 1968 :
 - ▶ *La recherche d'information est un domaine qui s'intéresse à la structure, l'analyse, l'organisation, le stockage et la recherche de l'information.*
- **Recherche ad hoc** : Un système de recherche d'information est un système qui permet de retrouver **une information pertinente** par rapport à une **requête** dans une grande collection de **documents**.
- Recherche d'information structurée (Bases de données).
- Recherche d'information non-structurée : textuelle, visuelle, sonore.

Avant propos

Structure

structure

[Log in / create account](#)

[Article](#) [Discussion](#) [Read](#) [Edit](#) [View history](#) [Search](#)

Gerard Salton

From Wikipedia, the free encyclopedia

Gerard Salton (8 March 1927 in Nuremberg – 26 August 1995), also known as Gerry Salton, was a Professor of Computer Science at Cornell University. Salton was perhaps the leading computer scientist working in the field of information retrieval during his time. His group at Cornell developed the SMART Information Retrieval System, which he initiated when he was at Harvard.

Salton was born Gerhard Anton Sahlmann on March 8, 1927 in Nuremberg, Germany. He received a Bachelor's (1950) and Master's (1952) degree in mathematics from Brooklyn College, and a Ph.D. from Harvard in Applied Mathematics in 1958, the last of Howard Aiken's doctoral students, and taught there until 1968, when he joined Cornell University and co-founded its department of Computer Science.

Salton was perhaps most well known for developing the now widely used Vector Space Model for Information Retrieval^[1]. In this model, both documents and queries are represented as vectors of term counts, and the similarity between a document and a query is given by the cosine between the term vector and the document vector. In this paper, he also introduced TF-IDF, or term-frequency-inverse-document frequency, a model in which the score of a term in a document is the ratio of the number of terms in that document divided by the frequency of the number of documents in which that term occurs. (The concept of inverse document frequency, a measure of specificity, had been introduced in 1972 by Karen Sparck Jones^[2]). Later in life, he became interested in automatic text summarization and analysis^[3], as well as automatic hypertext generation^[4]. He published over 150 research articles and 5 books during his life.

Salton was editor-in-chief of the Communications of the ACM and the Journal of the ACM, and chaired SIGIR. He was an associate editor of the ACM Transactions on Information Systems. He was an ACM Fellow (elected 1995), received an Award of Merit from the American Society for Information Science (1989), and was the first recipient of the SIGIR Award for outstanding contributions to study of information retrieval (1983) -- now called the **Gerard Salton Award**.

References

[edit]

- ¹ A. G. Salton, A. Wong, C. S. Yang, A vector space model for automatic indexing, Communications of the ACM, v.18 n.11, p.613-620, Nov. 1975
- ² Sparck Jones, Karen (1972), "A statistical interpretation of term specificity and its application in retrieval", Journal of Documentation 28 (1): 11–21, doi:10.1108/eb026526

FIGURE: D'après Jaime Arguello

Avant propos

Structure



The screenshot shows a Wikipedia article page for "Gerard Salton". The page has a standard layout with a sidebar on the left containing links to main page, contents, featured content, current events, random article, and donate to Wikipedia. The main content area is highlighted with a red box. It includes the title "Gerard Salton", a summary from "Wikipedia, the free encyclopedia", and a detailed biography of Salton's life and work. The biography mentions his birth in 1927, education at Cornell University, and his work on the SMART Information Retrieval System. It also discusses his contributions to the Vector Space Model for information retrieval and his role in SIGIR. The page ends with a "References" section containing two academic citations.

Gerard Salton

From Wikipedia, the free encyclopedia

Gerard Salton (9 March 1927 in Nuremberg – 30 August 1995), also known as Gerry Salton, was a Professor of Computer Science at Cornell University. Salton was perhaps the leading computer scientist working in the field of information retrieval during his time. His group at Cornell developed the SMART Information Retrieval System, which he initiated when he was at Harvard.

Salton was born Gerhard Anton Salmann on March 9, 1927 in Nuremberg, Germany. He received a Bachelor's (1950) and Master's (1952) degree in mathematics from Brooklyn College, and a Ph.D. from Harvard in Applied Mathematics in 1956, the last of Howard Aiken's doctoral students, and taught there until 1956, when he joined Cornell University and co-founded its department of Computer Science.

Salton was perhaps most well known for developing the now widely used Vector Space Model for Information Retrieval^[1]. In this model, both documents and queries are represented as vectors of term counts, and the similarity between a document and a query is given by the cosine between the term vector and the document vector. In this paper, he also introduced TF-IDF, or term-frequency-inverse document frequency, a model in which the score of a term in the a document is the ratio of the number of terms in that document divided by the frequency of the number of documents in which that term occurs. (The concept of inverse document frequency, a measure of specificity, had been introduced in 1972 by Karen Sparck-Jones^[2].) Later in life, he became interested in automatic text summarization and analysis^[3], as well as automatic hypertext generation^[4]. He published over 150 research articles and 5 books during his life.

Salton was editor-in-chief of the *Communications of the ACM* and the *Journal of the ACM*, and chaired SIGIR. He was an associate editor of the *ACM Transactions on Information Systems*. He was an ACM Fellow (elected 1995), received an Award of Merit from the American Society for Information Science (1989), and was the first recipient of the SIGIR Award for outstanding contributions to study of information retrieval (1983) – now called the *Gerard Salton Award*.

References

1. ^ G. Salton, A. Wong, C. S. Yang, A vector space model for automatic indexing, *Communications of the ACM*, v.18 n.11, p.613-620, Nov. 1975
2. ^ Sparck-Jones, Karen (1972), "A statistical interpretation of term specificity and its application in retrieval", *Journal of Documentation* 28 (1): 11–21, doi:10.1108/002525262

FIGURE: D'après Jaime Arguello

Cependant, le principal contenu est du texte en langage naturel : information peu structurée pour un ordinateur

Avant propos

Structure



The screenshot shows a Wikipedia page for 'Gerard Salton'. The title 'Gerard Salton' is highlighted with a red box. The page content discusses Salton's work on the Vector Space Model for Information Retrieval and his introduction of TF-IDF. It also mentions his work on document frequency and specificity. The page includes a sidebar with navigation links and a 'References' section with two citations.

document structure

Gerard Salton

From Wikipedia, the free encyclopedia

Gerard Salton (8 March 1927 in Nuremberg – 26 August 1999), also known as Gerry Salton, was a Professor of Computer Science at Cornell University. Salton was perhaps the leading computer scientist working in the field of information retrieval during his time. His group at Cornell developed the SMART Information Retrieval System, which he initiated when he was at Harvard.

Salton was born Gerhard Anton Salmann on March 8, 1927 in Nuremberg, Germany. He received a Bachelor's (1950) and Master's (1953) degree in mathematics from Brooklyn College, and a Ph.D. from Harvard in Applied Mathematics in 1958, the last of Howard Aiken's doctoral students, and taught there until 1968, when he joined Cornell University and co-founded its department of Computer Science.

Salton was perhaps most well known for developing the now widely used Vector Space Model for Information Retrieval^[1]. In this model, both documents and queries are represented as vectors of term counts, and the similarity between a document and a query is given by the cosine between the term vector and the document vector. In this paper, he also introduced TF-IDF, or term-frequency-inverse document frequency, a model in which the score of a term in the a document is the ratio of the number of terms in that document divided by the frequency of the number of documents in which that term occurs. (The concept of inverse document frequency, a measure of specificity, had been introduced in 1972 by Karen Sparck Jones^[2].) Later in life, he became interested in automatic text summarization and analysis^[3], as well as automatic hypertext generation^[4]. He published over 150 research articles and 5 books during his life.

Salton was editor-in-chief of the *Communications of the ACM* and the *Journal of the ACM*, and chaired SIGIR. He was an associate editor of the *ACM Transactions on Information Systems*. He was an ACM Fellow (elected 1990), received an Award of Merit from the American Society for Information Science (1969), and was the first recipient of the SIGIR Award for outstanding contributions to study of information retrieval (1983) -- now called the *Gerard Salton Award*.

References

1. * G. Salton, A. Wang, C. B. Yang, A vector space model for automatic indexing [\[d\]](#), *Communications of the ACM*, v.18 n. 11, p.613-620, Nov. 1975.
2. * Sparck Jones, Karen (1972), "A statistical interpretation of terms specificity and its application in retrieval" [\[d\]](#), *Journal of Documentation* 28 (1): 11–21, doi:10.1108/eb026526 [\[d\]](#)

FIGURE: D'après Jaime Arguello

La compréhension du langage naturel par un ordinateur n'est cependant pas nécessaire pour retourner un document vraisemblablement pertinent étant donnée une requête.

Avant propos

Structure de la collection

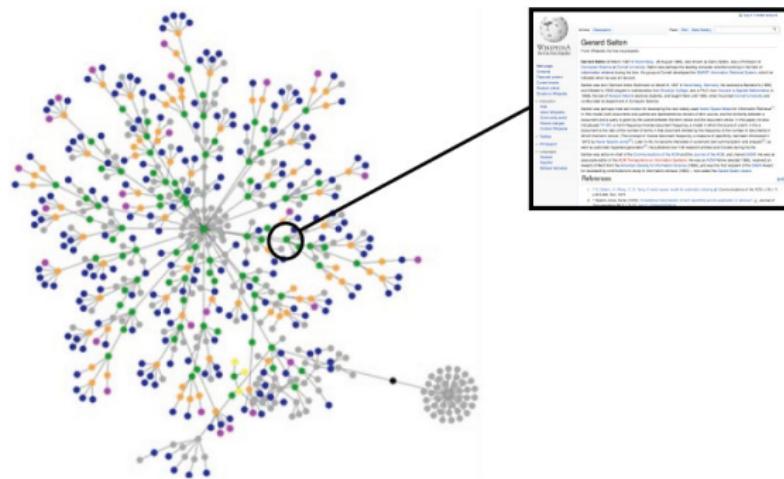


FIGURE: D'après Jaime Arguello

Avant propos

Analyse : classification, recommandations,...

Categories: 1927 births | 1995 deaths | American computer scientists | Computer pioneers | Harvard University alumni | Harvard University faculty | Cornell University faculty | Fellows of the Association for Computing Machinery | Guggenheim Fellows

Les clients ayant acheté cet article ont également acheté

			
Regarde dans la neige ► Emiri Hayashi (8)	Regarde dans la nuit ► Catherine Joussetme (28)	Regarde dans la mer ► Emiri Hayashi (13)	Où es-tu maman lapin ? : Mon premier album à ... ► Xavier Deneux (6)
Cartonné EUR 13,21	Cartonné EUR 13,21	Album EUR 13,21	Album EUR 13,21

Avant propos

Organisation

Exemple : catégorisation de sites web, Open Directory Project ou annuaire dmoz (www.dmoz.org)

The screenshot shows the homepage of the Open Directory Project (Dmoz) with a search bar and navigation links. A sub-menu for 'Computers' is highlighted, showing its sub-categories and a list of sub-topics under 'Information Retrieval'.

Computers
 Internet, Software, Hardware

Top: Computers: Software: Information Retrieval (96)

- Classification (16)
- Data Clustering (766)
- Fulltext (32)
- GILS (7)
- Internet Search Engines (314)
- Ranking (35)
- References (1)
- Text Clustering (11)
- Visual Information (6)
- Web Clustering (6)

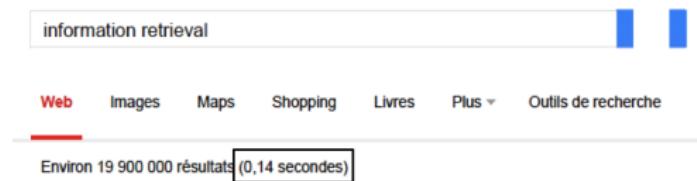
See also:

- Computers: Software: File Management: Search (37)
- Computers: Software: Internet: Servers: Search (41)
- Reference: Knowledge Management: Knowledge Retrieval (40)
- Reference: Libraries: Library and Information Science: Software (93)

FIGURE: D'après Jaime Arguello

Avant propos

Recherche



- Rapidité : retourner des résultats rapidement (0.14s)
- Efficacité : retourner des résultats satisfaisant le besoin d'information de l'utilisateur.

Dans le cours, on s'intéressera plus à l'efficacité qu'à la rapidité.

Objectifs du cours

Introduction des bases de la Recherche d'Information

Questions ?

- Comment fonctionne un moteur de recherche tel que Google ?
- Comment mesure t-on l'efficacité d'un moteur de recherche ?
- Quels sont les méthodes théoriques sous-jacentes ?
- Est ce que les approches classiques de la RI sont adaptées à la recherche d'information multimédia ?
- Quels sont les défis actuels de la RI ? Apprentissage et RI, Sémantique et RI...

Objectifs du cours

Plan

- Introduction à l'indexation et la recherche d'information
- Indexation - Index inversé
- Modèles de RI : modèles booléens, modèles vectoriels, modèles probabilistes, modèles de langue.
- Retour de pertinence
- Evaluation
- Recherche d'information sur le web
- Recherche d'information multimédia
- Apprentissage et RI
- Sémantique - Modèles de connaissance

Objectifs du cours

Organisation

- Chaque séance : 1 partie de cours et une mise en pratique (Projet à réaliser au fil des séances)
- Evaluation : projet + présentation d'un article scientifique.

Le matériel du cours et d'autres ressources sont disponibles sur Claroline
<http://cours.etudes.ecp.fr/claroline/course/index.php?cid=IS3013AA>

Bibliographie

Ressources (parmi d'autres)

- Livre : *Introduction to Information Retrieval*, (Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze), Cambridge University Press. 2008.
 - ▶ Intégralement en ligne <http://nlp.stanford.edu/IR-book/> avec des ressources pédagogiques.
- Livre : *Modern Information Retrieval*, (Ricardo Baeza-Yates, Berthier Ribeiro-Neto), Addison Wesley
 - ▶ Quelques chapitres en ligne <http://www.mir2ed.org/> avec des ressources pédagogiques.

Plan

- 1 Contexte
- 2 Introduction
- 3 Historique des SRI
- 4 Indexation
 - Des documents aux termes du dictionnaire
 - Lois de base en RI
 - Représentation d'un document et de la collection - Index inversé
- 5 Modèles classiques de recherche d'Information
 - Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes
 - Modèle vectoriel
- 6 Bilan

Un domaine très actuel



FIGURE: D'après Jaime Arguello

Un domaine très actuel

Ne concerne pas uniquement les moteurs de recherche du WEB :

- Solutions pour les entreprises : FAST, Autonomy, Eurospider, Cyveillance, Google Enterprise, Hoovers, ...
- Bibliothèques numériques ou non : BNF, European Digital Library, Wikipedia, ...
- Archives numériques ou non : INA, OpenCourseWare, ...
- Articles scientifiques : CiteSeer, WebOfScience, Elsevier, ...
- Nouveaux usages, nouvelles applications : réseaux sociaux, partage d'images et de vidéos, systèmes de recommandations, e-commerce, ...
- ...

Un domaine très actuel

- Communication : information non-structurée
 - ▶ Textes, images, parole,... : Flickr, Deezer, ...
- On stocke de plus en plus l'information de manière électronique
- Le volume augmente :
 - ▶ Moins de temps pour organiser l'information
 - ▶ Temps de réponse rapide
- Avec Internet et le WEB, accès facile à l'information

La recherche d'information

Les 3 composantes d'un système de RI

- **La collection** : ensemble des documents disponibles.
- **L'utilisateur** : a un besoin d'information ou une tâche à accomplir souvent exprimés par une requête.
- **Le système** : ensemble des méthodes et outils permettant de retrouver dans la collection les documents **pertinents** pour le besoin de l'utilisateur.

La recherche d'information

- Etant donnée une **requête** et un **corpus**, trouver les documents **pertinents** par rapport à la requête.
 - ▶ **Requête** : exprime le besoin d'information de l'utilisateur.
 - ▶ **Corpus** : collection de documents dans laquelle on recherche l'information.
 - ▶ **Pertinence** : satisfaction de l'utilisateur par rapport à son besoin d'information.

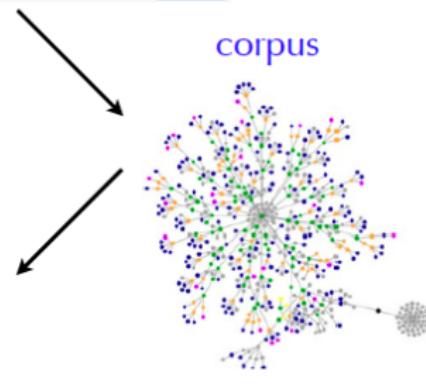
La recherche d'information

Recherche web

query

results

- [Study: Facebook use cuts productivity at work - Computerworld](#) www.computerworld.com - Internet - Web 2.0 and Web Apps - Cached
Jul 22, 2009 - A Nucleus Research study found that Facebook work in the workplace is cutting employee productivity.
- [Pulling the Plug on Facebook: Productivity/Time Management Article](#) www.inc.com - Leadership and Managing - Human Resources - Cached
Pulling the Plug on Facebook: Productivity/Time Management Article - All that friending and superposting wastes a lot of time at the office -- and could be ...
- [Twitter and Facebook: The New Tools of Productivity or Distraction](#) www.brighthub.com/.../twitter-and-facebook-the-new-tools-of-productivity - Cached
Mar 26, 2010 - RT Twitter and Facebook: Tools of Productivity or Distraction ... RT @PRSAccid: Twitter & Facebook: New tools of productivity or...
- [Twitter, Facebook Can Improve Work Productivity | PCWorld Business](#) www.pcworld.com/.../twitter_facebook_can_improve_work_productivity - Cached
Apr 2, 2009 - Reach Older Users on Facebook and Twitter: The Web's Best Productivity Sites. According to a study by the Australian University, ...
- [Is Facebook Killing Your Employees' Productivity? | WebProNews](#) www.webpronews.com/is-facebook-killing-your-employees-productivity - Cached
Jul 21, 2009 - On the heels of a study indicating that social media can significantly impact a brand's bottom line positively, another one has come out ...
- [Productivity Strategies | Facebook](#) www.facebook.com/beproductive - Cached
Productivity Strategies - To learn more about the Productive Today "Content Collaborative" faculty, click the "Info" tab or this direct link | Facebook.
- [Put Out IT: Facebook "Productivity Loss" Is No Concern of Yours](#) blogs.gartner.com/.../put-out-it-facebook-productivity-loss-is-no-concern-of-yours - Cached
Facebook "Productivity Loss" Is No Concern of Yours, by Brian Peacock | November 23, 2008 | 13 Comments. Like my colleague Anthony Bradley, I also speak to ...
- [Productivity Levels Plummet After Yale Student Makes Facebook Look Like](#) www.buzzfeed.com/.../yale-student-makes-facebook-look-like-because... - Cached
6 days ago - Productivity Levels Plummet After Yale Student Makes Facebook Look Like Excel. By Rebecca Palkov 7/25 8:11pm ...



corpus

web pages

FIGURE: D'après Jaime Arguello

La recherche d'information

Recherche dans des bases d'articles scientifiques

query

facebook productivity

results

- 1 [Effective teaching practices using free Google services: conference tutorial](#)
Paul Gestwicki, Brian McNely
 October 2010 [Journal of Computing Sciences in Colleges](#), volume 26 issue 1
Publisher: Consortium for Computing Sciences in Colleges
 Full text available: [pdf](#) (22.76 KB)
Bibliometrics: Downloads (6 Weeks): 2, Downloads (12 Months): 48, Downloads (Overall): 48,
 In this 90-minute tutorial, we will share our experiences using free Web services from Google to teach effectiveness. Participants will engage with these services as part of the tutorial. We have used and studied these technologies, ...
- 2 [Model-Based Engineering of Software: Three Productivity Perspectives](#)
Shawn A. Bohner, Sriram Mohan
 October 2009 [SEW '09: Proceedings of the 2009 33rd Annual IEEE Software Engineering Workshop](#)
Publisher: IEEE Computer Society
 Full text available: [Publisher Site](#)
Bibliometrics: Downloads (6 Weeks): n/a, Downloads (12 Months): n/a, Downloads (Overall): n/a, Citation Count: 0
 Evolving software products is a tricky business, especially when the domain is complex and changing rapidly. Like other fields of engineering, software engineering productivity advances have come about largely through abstraction reuse, process, and ...
Keywords: Agent-Based Software Systems, Model-Driven Architecture, Model-Driven Development, Model-Based Software Development, Model-Based Software Engineering
- 3 [Absolute Beginner's Guide to Computer Basics, 5th edition](#)
Michael Miller
 September 2009 [Absolute Beginner's Guide to Computer Basics, 5th edition](#)
Publisher: Que Publishing Company
Bibliometrics: Downloads (6 Weeks): n/a, Downloads (12 Months): n/a, Downloads (Overall): n/a, Citation Count: 0
 Everything casual users need to know to get the most out of their new Windows 7 PCs, software, and the Internet. The best-selling beginner's guide, now completely updated for Windows 7 and today's most popular Internet tools -

corpus



scientific publications

La recherche d'information

Recherche d'actualités

query



results

[Tropical Storm Emily Heads Toward Haiti, Dominican Republic](#)

[Value of America](#) - 25 minutes ago

August 03, 2011 Tropical Storm Emily Heads Toward Haiti, Dominican Republic VOA News Tropical storm warnings and watches are posted for parts of the ...

[+ Video: Tropical Storm Emily on Path toward Haiti](#) The Associated Press

Local: [Emily could reach hurricane strength](#) Sarasota Herald-Tribune (blog)

Blog: [Mubarak Trial Begins; Tropical Storm Emily Threatens East Coast](#) [NPR](#) (blog)

[The Guardian](#) • [For News](#)

[all 1024 news articles](#) x

[Emily Heads For Hispaniola](#)

[WABC-TV](#) (blog) - Jeremy Wheeler - 2 hours ago

Emily has had little change in strength since yesterday. She has winds of 50mph. The pressure is down a little to 1003 mb (millibars of pressure). ...

[Tropical Storm Emily remains weak, hits Haiti tonight](#) [Examiner.com](#)

August 2, 2011 Midday Tropical Update: TS Emily Now Moving More ... [Website.com](#)

[Emily the Effervescent](#) [Cartoon Hurricane Nation](#):

[all 14 news articles](#) x

['Snow White' Writer to Pen Universal's 'Emily the Strange' \(Exclusive\)](#)

[Hollywood Reporter](#) - [Gavia Ni](#) - 10 hours ago

Melissa Wallack, who wrote the script that became Relativity's high-profile Snow White project, has been

drafted by Universal to pen *Emily the Strange* ...

[Early Edition: Emily the Strange to Darken Big Screen; More News](#) [MovieFone](#) (blog)

[Writer Melissa Wallack Will Follow SNOW WHITE with EMILY THE STRANGE](#) [Collider.com](#)

[Details on the upcoming 'Emily the Strange' movie](#) [Examiner.com](#)

[ComicBook.net](#) - [411mania.com](#)

[all 7 news articles](#) x

[High heat in Midwest and South](#)

[Reuters](#) - [Tom Shep](#) - [Kevin Murphy](#) - 22 hours ago

By Wendell Marsh **WASHINGTON** (Reuters) - Record-breaking heat continued to broil central and southern states on Tuesday as Tropical Storm Emily threatened to ...



Carica

corpus



news articles



Collider.com



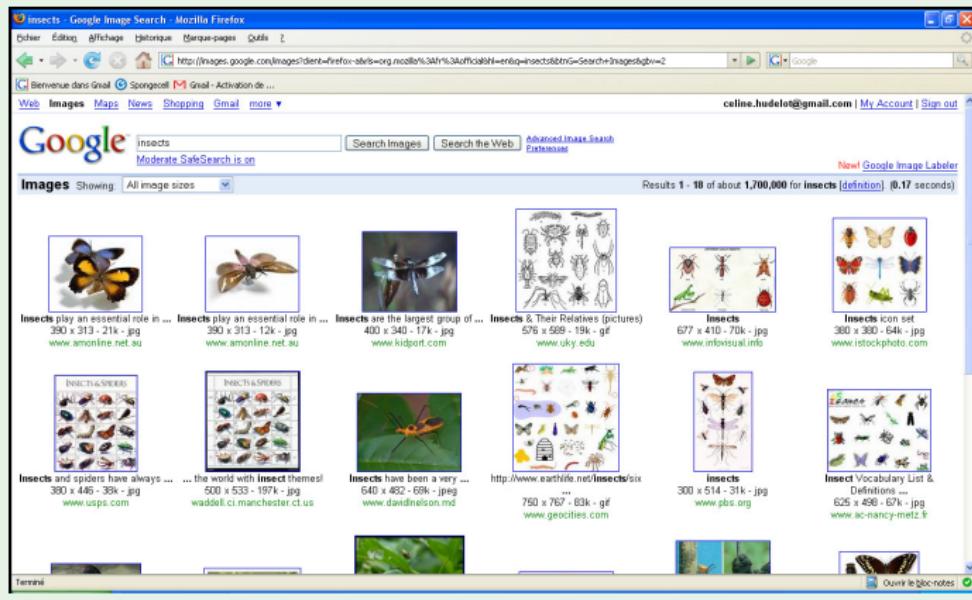
Reuters

FIGURE: D'après Jaime Arguello

La recherche d'information

Recherche d'images

Avec plusieurs modalités



La recherche d'information

Recherche géo-localisée de commerces

query

results

- Places for mexican food near Chapel Hill, NC
- A** [Bandido's Mexican Cafe & Cantina](#) - ★★★★½ ★ 14 reviews - Place page
www.bandidoscafe.com - 150 ½ East Franklin Street, Chapel Hill - (919) 967-5048
 - B** [Las Potrillos Mexican Restaurant](#) - ★★★★½ ★ 9 reviews - Place page
www.laspotrillos.net - 220 West Rosemary Street, Chapel Hill - (919) 932-4301
 - C** [monterrey mexican restaurant](#) - ★★★★½ ★ 17 reviews - Place page
monterreychapelhill.com - 237 South Elliot Road, Chapel Hill - (919) 989-8750
 - D** [Margaret's Cantina](#) - ★★★★½ ★ 16 reviews - Place page
www.margaretscantha.com - 1129 Weaver Dairy Road, Chapel Hill - (919) 942-4745
 - E** [Qocba Mexican Grill](#) - ★★★★½ ★ 19 reviews - Place page
www.qocba.com - 100 West Franklin Street, Chapel Hill - (919) 929-8990
 - F** [Cinco de Mayo](#) - ★★★★½ ★ 11 reviews - Place page
www.cincodemayorestaurants.net - 1502 East Franklin Street, Chapel Hill - (919) 929-8566
 - G** [Chipotle Mexican Grill](#) - ★★★★½ ★ 15 reviews - Place page
www.chipotle.com - 301 W. Franklin St., Chapel Hill - (919) 942-2091

corpus

Google maps

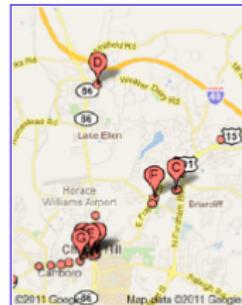
curated/synthesized
business listings

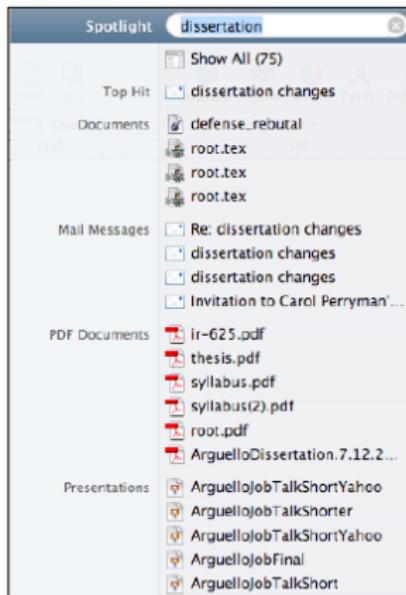
FIGURE: D'après Jaime Arguello

La recherche d'information

Recherche dans des archives locales

query

results



corpus



files in my laptop

FIGURE: D'après Jaime Arguello

La recherche d'information

Définition

- **Recherche de documents** ou d'information dans des documents par rapport à une **requête utilisateur**
- La recherche doit être efficace : **comment mesurer cette efficacité ?**
- L'étude de données **non-structurées**
- Un **document** est un conteneur pour les données
 - ▶ Sa structure n'est pas forcément connue
 - ▶ Différents formats : HTML, XML, ...
 - ▶ Les données n'ont pas de syntaxe ou de sémantique associées.
- Types de documents
 - ▶ Textes, hypertextes, audios, images, vidéos, données biologiques, ...

La recherche d'information

De manière générale, une discipline qui traite de

- **Recherche** :
 - ▶ Représentation
 - ▶ Stockage
 - ▶ Organisation
 - ▶ Accès
- **Données** : structurées, semi-structurées ou **non-structurées**
- Réponse à une **requête**
 - ▶ Structurée : expression booléenne
 - ▶ Non-structurée : phrase, document

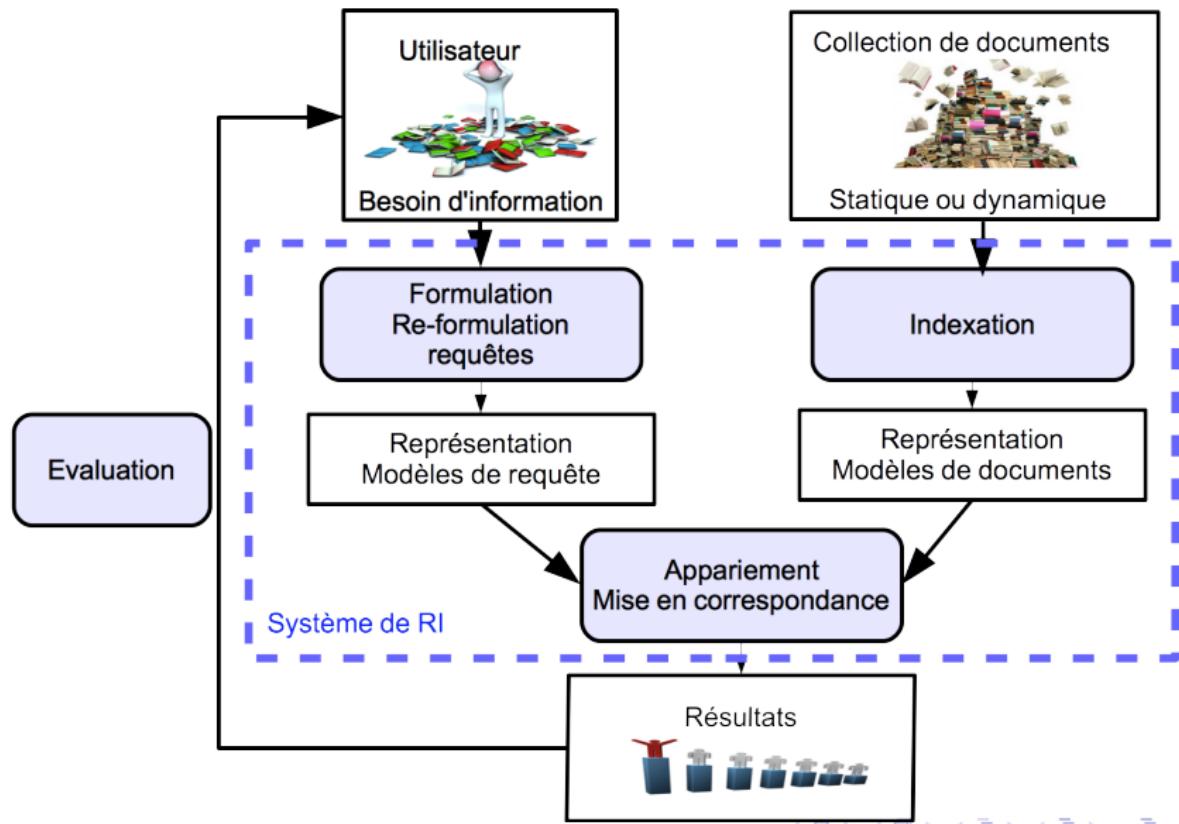
Information vs données : ensemble de données qui correspond à un besoin particulier.

Notions importantes : La pertinence

Un document **pertinent** est un document qui doit contenir l'information que l'utilisateur recherche.

- Notion complexe qui dépend :
 - ▶ De l'utilisateur
 - ▶ De la requête
- Notion qui permet aussi d'évaluer les systèmes de recherche d'information.

La recherche d'information : principe



Besoin d'information

Rôle de l'utilisateur

- Les besoins en information ont des aspects qualitatifs et quantitatifs
- Expression des besoins sous la forme d'une requête
- Deux tâches principales :
 - ▶ *Retrieval* : requête avec un objectif bien précis et donc recherche d'une information précise.
 - ▶ *Browsing* : requête moins bien définie, navigation, exploration.

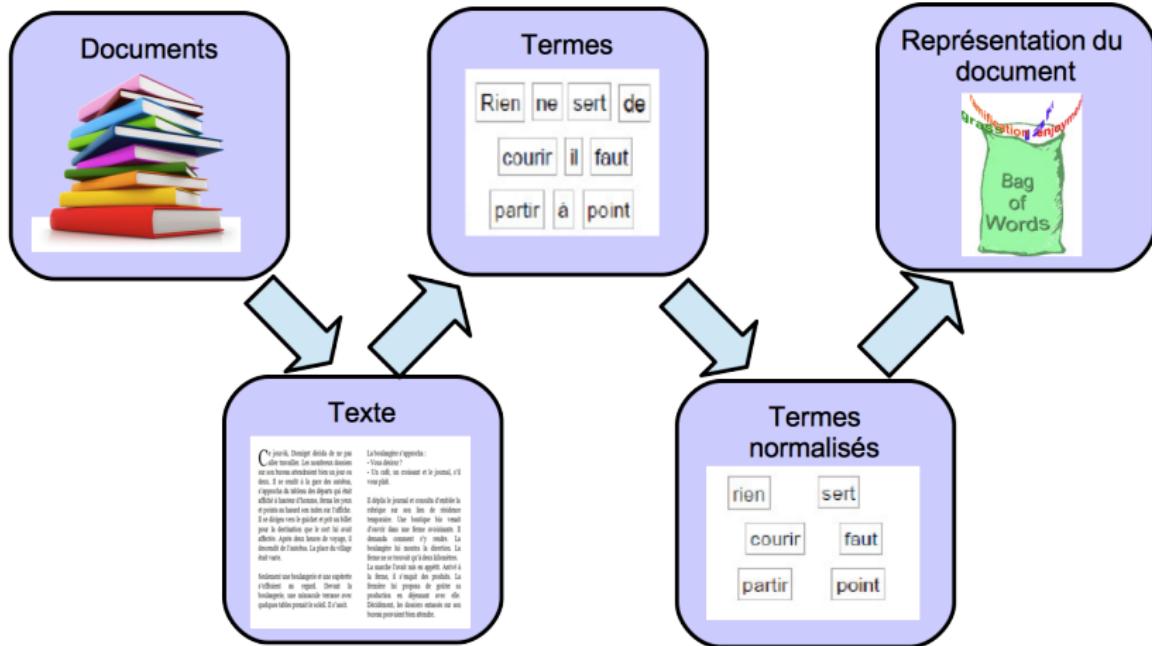
Indexation

Pourquoi ?

- Le parcours de l'ensemble des documents avec les termes d'une requête est impossible.
- **Indexation** : traitement préalable.
- Indexation automatique : *Transformer des documents en substituts capables de représenter le contenu de ces documents* (Salton et McGill, 1983)
- Les index peuvent prendre plusieurs formes : mots simples, mots complexes, entrées de thésaurus, ...

Indexation : Représentation d'un document

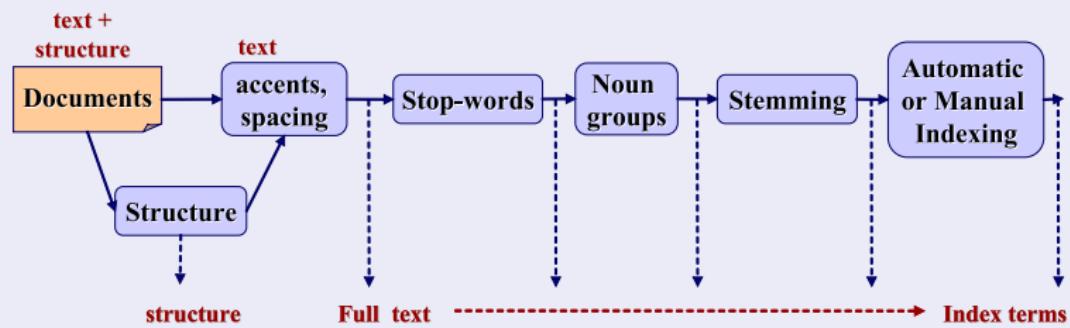
Vue générale



Indexation : Représentation d'un document

Vue logique d'un document

Traditionnellement, pour un document textuel, un ensemble de termes d'index ou de mots-clés



Représentation d'un document

Chaîne de traitements linguistiques

- **Segmentation** (*Tokenisation*) : séparation d'une suite de caractères en éléments sémantiques ou mots.
- **Filtrage** : suppression des mots fréquents ou stop-words : articles, conjonctions, ...
- **Normalisation** (Lemmatisation - Racinisation - *Stemming*) : transformation d'un mot en sa forme canonique

La recherche d'information : Ranking

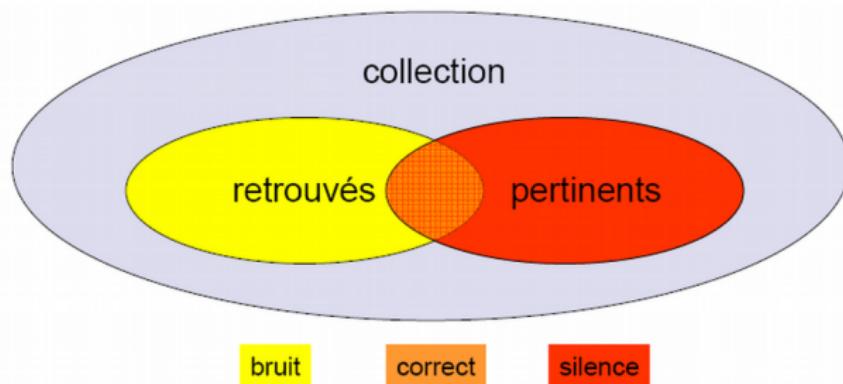
Importance du classement

- L'ordre des documents retrouvés reflète leur pertinence par rapport à la requête utilisateur
- Question de la pertinence et la mesure de la pertinence

La recherche d'information : Evaluation

Rappel et précision

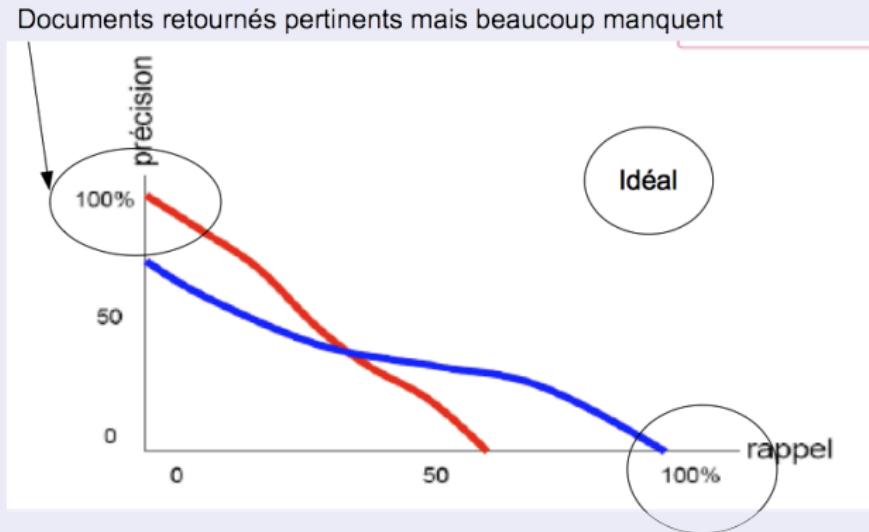
- Rappel : nombre de documents pertinents retrouvés par rapport au nombre de documents pertinents
- Précision : nombre des documents pertinents retrouvés par rapport au nombre de documents retrouvés



La recherche d'information

Rappel et précision

RI : compromis entre rappel et précision



Beaucoup de documents pertinents
Mais beaucoup d'erreurs aussi

Plan

- 1 Contexte
- 2 Introduction
- 3 Historique des SRI
- 4 Indexation
 - Des documents aux termes du dictionnaire
 - Lois de base en RI
 - Représentation d'un document et de la collection - Index inversé
- 5 Modèles classiques de recherche d'Information
 - Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes
 - Modèle vectoriel
- 6 Bilan

Un peu d'histoire

1930 : Apparition des mémoires de masse

Apparition de la carte perforée

- Issue de la mécanographie
- Premières mémoires de masse et premiers supports d'entrée-sortie en Informatique



http://en.wikipedia.org/wiki/Punched_card

Un peu d'histoire

1945 : Memex de Vannevar Bush

Ordinateur analogique théorique permettant de naviguer à travers des informations.

http://web.archive.org/web/20070113233131/http://www.dynamicdiagrams.com/case_studies/mit_memex.html

- Description dans l'article *As We May Think* (1945) (Description d'un ancêtre de WWW)
(<http://www.theatlantic.com/doc/194507/bush>)
- Base du projet MyLifeBits de Gordon Bell (*a lifetime store of everything*) : <http://research.microsoft.com/en-us/projects/mylifebits/>

<http://en.wikipedia.org/wiki/Memex>

Un peu d'histoire

1948 : Théorie de l'information de Shannon

- Publication de *A Mathematical Theory of Communications*
- Théorie pour le codage de l'information
- Mesure quantitative de l'information ($I = \log(\frac{1}{p(x)})$, quantité d'information apportée par le message x)

Un peu d'histoire

Et la suite

- 1948 : Le terme **Information retrieval** est introduit par Calvin N. Mooers (Informaticien américain)
- 1950 : Premiers articles scientifiques.
- 1958 : Première conférence internationale : *International Conference on Scientific Information*
- 1963 : Système SMART (System for the Mechanical Analysis and Retrieval of Text) de Gerald Salton (Cornell University) (modèle booléen, modèle vectoriel, retour de pertinence).
- Projet d'évaluation CRANFIELD qui définit les mesures d'évaluation.
- Grands systèmes de gestion de bases documentaires.
 - ▶ 1964 : Medline : Medical Literature Analysis and Retrieval System Online (<http://medline.cos.com/>)
 - ▶ 1967 : Système DIALOG de Lockheed (langage d'interrogation d'un serveur de bases de données documentaires)

Un peu d'histoire

Suite

- 1990 : Révolution dans le stockage numérique à bas coût
- Moteurs de recherche de documents ftp
 - ▶ ARCHIE : considéré comme le premier moteur de recherche WEB
 - ▶ WAIS (Wide Area Information System) :
- Recherche d'information sur le Web
 - ▶ 1994- ... : Moteurs de recherche pour le WEB : Lycos, Yahoo, Altavista
 - ▶ 1998 : PageRank de Google
 - ▶ ...
- Systèmes de recommandations
 - ▶ Amazon
- Recherche d'information multimédia

Domaines connexes

- SGBD
- Ingénierie documentaire
- Intelligence artificielle
- Traitement du langage naturel et des données multimédias
- Apprentissage statistique

SRIIs vs SGDBs

	Databases	IR
Data	Structured	Unstructured
Fields	Clear semantics (SSN, age)	No fields (other than text)
Queries	Defined (relational algebra, SQL)	Free text ("natural language"), Boolean
Recoverability	Critical (concurrency control, recovery, atomic operations)	Downplayed , though still an issue
Matching	Exact (results are <i>always</i> "correct")	Imprecise (need to measure effectiveness)

FIGURE: From James Allan

SRLs vs Ingénierie documentaire

- Focalisé sur les aspects humains de la recherche d'information
- Catégorisation effective de la connaissance humaine
- Importance des citations et de la bibliométrie
- Un domaine de plus en plus proche avec les bibliothèques numériques

SRIs vs Artificial Intelligence

- Focalisé sur la représentation de la connaissance et du raisonnement
- Formalisme pour représenter la connaissance et les requêtes
 - ▶ Logique des prédictats du 1er ordre
 - ▶ Réseaux bayesiens
- Web sémantique et ontologies : vers une convergence avec le domaine de la RI.
- Recherche sémantique - Question Answering

SRLs vs Traitement du Langage naturel

- Focalisé sur l'analyse syntaxique, sémantique et pragmatique du langage naturel et du discours
- Vers une recherche sémantique et plus à base de mots clés
- Convergences :
 - ▶ Desambiguation d'un mot selon son contexte
 - ▶ Extraction d'information pertinente dans un document
 - ▶ Multi-linguisme

SRLs vs Apprentissage statistique

- Focalisé sur l'amélioration de systèmes en se basant sur l'expérience.
Deux approches :
 - ▶ Apprentissage supervisé : classification d'exemples en se basant sur un ensemble d'apprentissage labélisé
 - ▶ Apprentissage non supervisé : création de groupes d'exemples
- Convergence :
 - ▶ Ordonnancement par apprentissage.
 - ▶ Catégorisation de textes ou de documents multimédias
 - ▶ Clustering de textes
 - ▶ Fouille de données textuelles et multimédias

Avant propos

Lectures conseillées

- Chapitres 1, 2 et 6 du livre *Introduction to Information Retrieval*.
<http://nlp.stanford.edu/IR-book/>
- Chapitres 1,2,3 du livre *Information Retrieval : Implementing and Evaluating Search Engines*.
<http://www.ir.uwaterloo.ca/book/>.

Plan

- 1 Contexte
- 2 Introduction
- 3 Historique des SRI
- 4 Indexation
 - Des documents aux termes du dictionnaire
 - Lois de base en RI
 - Représentation d'un document et de la collection - Index inversé
- 5 Modèles classiques de recherche d'Information
 - Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes
 - Modèle vectoriel
- 6 Bilan

Indexation

Pourquoi ?

- L'idée principale d'un moteur de recherche est de retrouver les documents qui *traitent* de la requête en utilisant ce qui est disponible : **les mots**
- Le parcours de l'ensemble des documents avec les termes d'une requête est impossible.
- **Indexation** : traitement préalable.
- Indexation automatique : *Transformer des documents en substituts capables de représenter le contenu de ces documents* (Salton et McGill, 1983)
- Les index peuvent prendre plusieurs formes : mots simples, mots complexes, entrées de thésaurus, ...

Indexation

Définitions

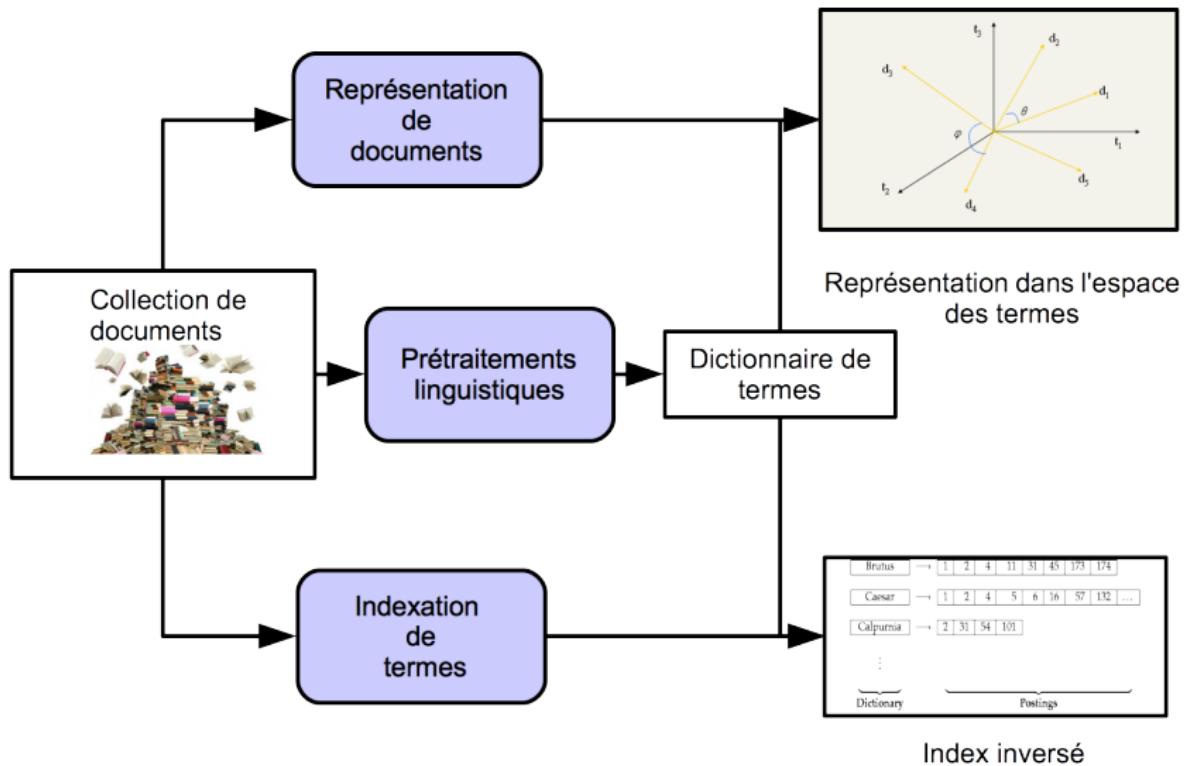
- **Index** : structure permettant d'organiser un ensemble de fichiers pour rendre efficace la recherche d'information dans ces fichiers
- La représentation des textes (i.e. **leur indexation**) est le processus de décision des éléments qui vont servir à représenter chaque document
- **Termes d'indexation** (termes d'index)

Indexation

Un processus à plusieurs étapes

- ① Création d'un vocabulaire de termes (**termes d'index**) à partir des documents de la collection : les termes choisis doivent permettre de caractériser la collection.
Etape qui fait appel à de nombreux algorithmes et méthodes du traitement du langage naturel.
- ② Représentation de chaque document de la collection à partir du dictionnaire, e.g. approche sac de mots.
- ③ Construction d'un **index inversé** pour représenter la collection.

Indexation : principe général



Indexation : manuelle ou automatique ?

On associe aux documents les termes qui en représentent le mieux le sens

Indexation manuelle

- Mots-clés dans un **vocabulaire contrôlé** (souvent organisé hiérarchiquement) : PubMed (MeSH), Yahoo, ACM, ...
- Très précis mais très couteux en temps

Indexation automatique

- Extraire les termes d'index de manière automatique des documents
- Approche sac de mots (*bag of words*) :
 - ▶ L'ordre des mots n'est pas important.
 - ▶ On raisonne en terme de présence ou absence des termes dans un document ou en terme de fréquence de ces termes.

Vocabulaire contrôlé vs indexation libre

	Cost of assigning index terms	Ambiguity of index terms	Detail of representation
Controlled Vocabularies	High	Not ambiguous	Can't represent arbitrary detail
Free-text Indexing	Low	Can be ambiguous	Any level of detail

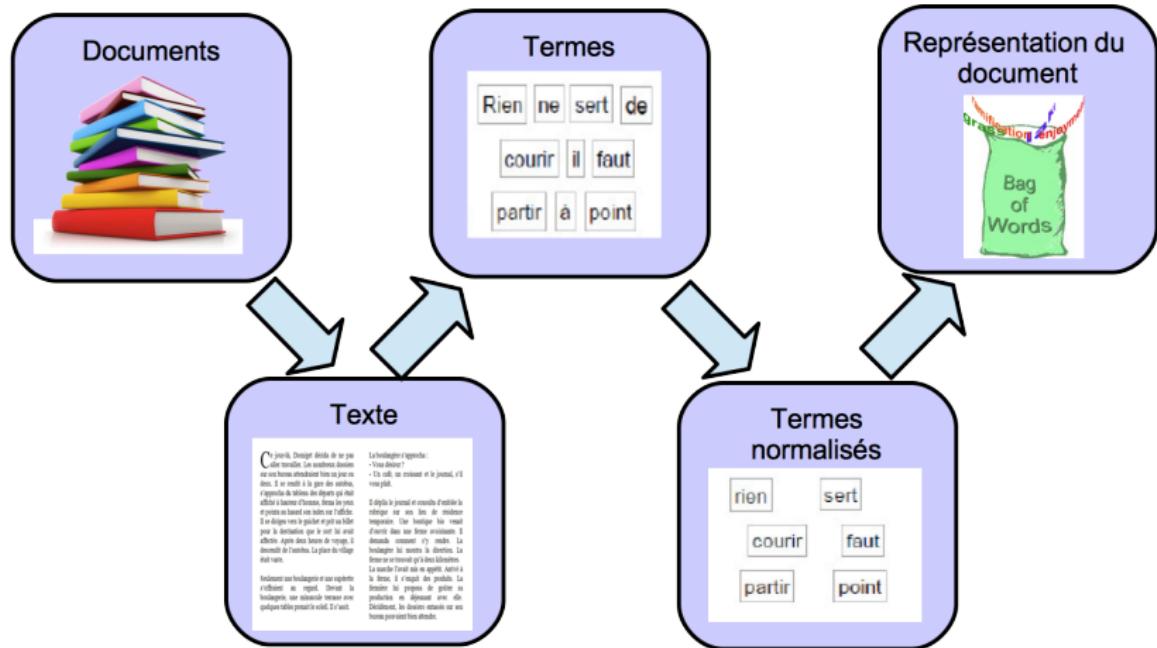
FIGURE: D'après Jaime Arguello

Plan

- 1 Contexte
- 2 Introduction
- 3 Historique des SRI
- 4 Indexation
 - Des documents aux termes du dictionnaire
 - Lois de base en RI
 - Représentation d'un document et de la collection - Index inversé
- 5 Modèles classiques de recherche d'Information
 - Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes
 - Modèle vectoriel
- 6 Bilan

Indexation : Représentation d'un document

Vue générale



Indexation automatique : une suite de traitements linguistiques

- Du texte aux termes : Comment reconnaît-on un mot ?
- Un certain nombre d'étapes de traitement du texte.
 - ▶ Identification de la langue
 - ▶ **Tokenisation** (i.e. Segmentation en unités)
 - ★ Espace, apostrophes, tirets, points, ...
 - ▶ **Filtrage - Stop lists** : utilisation de listes pour filtrer les mots non souhaités (mots vides, i.e. mots courants de la langue qui n'apportent pas d'information discriminante)
 - ▶ **Normalisation - Stemming** : procédure de troncature pour trouver une forme plus générale des mots (forme plurielle, forme conjuguée)

Quelques définitions

- **Mot** : chaîne de caractères délimitée, telle qu'elle apparaît dans le texte.
- **Terme** : un mot *normalisé* (casse, morphologie, orthographe,...), une classe de mots équivalents.
- **Token** : instance d'un mot ou d'un terme apparaissant dans un document.

Indexation :

Du texte aux termes : La segmentation

- Identification des unités élémentaires.
- Un problème qui peut être très complexe dans certaines langues mais étape initiale **indispensable** pour tout travail sur le texte.
- En sortie, des **mot**s, ou des **termes**, ou des **tokens**.
- Seront les candidats à l'indexation et à la recherche d'une requête.

Indexation :

Du texte aux termes : La segmentation

Dans les langues européennes

- Des **délimiteurs** de mots parfois ambigus
exemples : *etc.* ; *T.A.L* ; *aujourd'hui* ; *Jean-Paul* ; *prend-il* ; ...
- Les mots peuvent avoir des variantes
exemple : accent ou pas sur des lettres majuscules en Français.
- L'espace n'est pas toujours le bon délimiteur
exemples : *San Francisco* ; ...
- Les nombres et les dates sont problématiques.

Indexation : Filtrage

Mots outils

- N'apportent pas de sens au texte.
 - ▶ Ex : déterminants : *le, la, ...*
 - ▶ Ex : pronoms : *je, vous, ...*
 - ▶ Ex : prépositions : *sur, contre, ...*
- Les mots les plus fréquents de la langue : leur suppression permet d'économiser beaucoup de place dans l'index.
- Mais :
 - ▶ Utiles pour des requêtes multi-termes : *sapin de noël, ...*
 - ▶ Parfois porteurs de sens : *être ou ne pas être, ...*

Indexation : Normalisation

Processus de transformation des mots d'une même famille sous une forme normale ou canonique pour faciliter la mise en correspondance entre la requête et les documents de la collection.

Deux types principaux

- **Normalisation textuelle** : création de la forme canonique par application d'un ensemble de règles de transformations textuelles
 - ▶ Ponctuations, casse, accents, date et valeur monétique
- **Normalisation linguistique**
 - ▶ **Racinement (Stemming)** : regroupement des différentes variantes morphologiques d'un mot autour de la même racine.
 - ▶ **Lemmatisation** : suppression des variantes flexionnelles des mots pour les ramener sous leur forme encyclopédique.

Indexation : Lemmatisation

Lemmatisation

- Obtention de la **forme canonique (le lemme)** à partir du mot :
 - ▶ Pour un verbe : sa forme à l'infinitif
montrer, montreras, montraient → *montrer*
 - ▶ Pour un nom, adjectif, article : sa forme au masculin singulier
vert, vertes, verts → *vert*
- Difficulté : demande des ressources et un traitement linguistique dépendant de la langue

Le lemme correspond à un mot réel de la langue

Indexation : Steeming (Racinement)

Racinement : procédé de transformation des flexions en leur radical ou racine (après suppression du préfixe et du suffixe).

Les besoins

Normalisation de mots identiques

- Dans les documents et dans la requête.
- Difficulté : l'identité des mots
 - ▶ *Tuebingen, Tübingen et Tubingen* → Tübingen
 - ▶ *pêche* et *péché* ne sont pas identiques.
- Problèmes des fautes d'orthographe.

Indexation : Steeming (Racine)

Racine(stemming)

- Obtention de la **racine**, une forme tronquée du mot, commune à toutes les variantes morphologiques :
 - ▶ Suppression des flexions et des suffixes
cheval, chevaux, chevalier, chevalerie, chevaucher → *cheval*
- Généralement à base de règles
- Agrège plus que la lemmatisation.

La racine ne correspond pas forcément à un mot réel de la langue.

Indexation : Steeming (Racinement)

Racinement : algorithme de Porter

<http://tartarus.org/~martin/PorterStemmer/>

- Algorithme proposé en 1980 pour l'anglais (adapté au français) et qui consiste en une cinquantaine de règles classées en 7 phases :
 - ▶ Traitement des pluriels et verbes à la troisième personne.
 - ▶ Traitement du passé et du progressif
 - ▶ Exemple : supprimer le suffixe *ement* si le reste est plus long qu'un caractère.
 - ▶ *replacement* → *replac*
 - ▶ *cement* → *cement*
- Quand plusieurs règles peuvent s'appliquer, on choisit celle qui supprime le plus long suffixe.

Indexation : Steeming (Raciny)

Algorithme de Porter : quelques exemples de règles

Rule

SSES → SS
IES → I
SS → SS
S →

Example

caresses → caress
ponies → poni
caress → caress
cats → cat

Indexation : Etiquetage

Etiquetage

- Associer aux mots leur **catégorie morphosyntaxique** (nom, verbe,adjectif, ...)
- Peut être utile en RI pour :
 - ▶ Supprimer des mots inutiles.
 - ▶ Regrouper en termes complexes.
 - ▶ Rechercher des mots ambigus.
- Un processus long.

Indexation : Etiquetage

- Le choix des mots d'index peut aussi se faire avec une **analyse morpho-syntaxique**.
- On attribut à chaque mot une catégorie grammaticale et un lemme

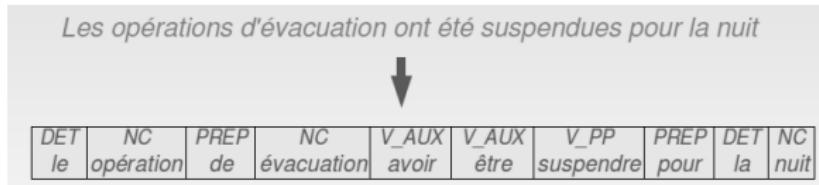


FIGURE: Source : A.Rozenkrop

- Analyse syntaxique : on peut garder les relations syntaxiques (extraction de mots composés)

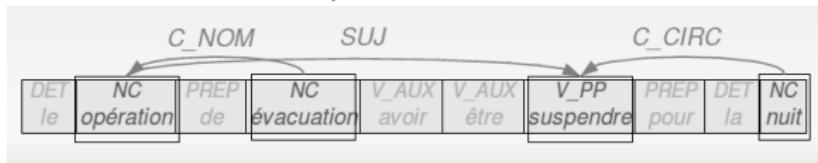


FIGURE: Source : A.Rozenkrop

Indexation : Etiquetage

Information morpho-syntaxique et extraction de mots composés

air-NC	hélicoptère_chinook-NC	nuit-NC
aller-V	hélicoptère-NC	nuit_tir_hélicoptère_chinook-NC
américain-ADJ	immense-ADJ	nuit_tir_hélicoptère-NC
annoncer-V	impliquer-V	nuit_tir-NC
armée_air-NC	inclure-V	opération_évacuation-NC
armée_américain-NC	jeudi-NC	opération_immense-NC
armée-NC(x3)	marine-NC	opération_immense_secours-NC
armée_régulier-NC	membre_corps_militaire-NC	opération-NC(x2)
chinook-NC	membre_corps-NC	opération_secours-NC
corps_militaire-NC	membre-NC	pouvoir-V
corps-NC	militaire-ADJ	régulier-ADJ
envoyer-V	nuit_hélicoptère_chinook-NC	secours-NC
évacuation-NC	nuit_hélicoptère-NC	...

FIGURE: Source : A.Rozenkrop

Indexation : Ordre de grandeur

Collection de Wikipédia Français.

Variables	Symboles	Valeurs
# de documents de la collection	N	1 349 539
# total d'occurrences des mots	M	696 668 157
# moyen de mots par document		416
Taille de la collection segmentée sur le disque		4,6 Go
# de types de mots	M_y	757 476
# de types de mots après racinisation	M_{Nor}	604 444
# de termes du vocabulaire	V	604 244
# moyen de termes par document		225
Taille de la collection prétraitée sur le disque		2,8 Go

FIGURE: Source : Amini - Gaussier

Quelques outils

Outils pour les traitements linguistiques

- Plate-forme NLTK (Natural Language Toolkit) : python, de nombreuses fonctionnalités
<http://www.nltk.org/>
- Stanford Core NLP : java,
<http://nlp.stanford.edu/software/index.shtml>
- IBM Bluemix, Alchemy API :
<https://console.ng.bluemix.net/catalog/>,
<https://console.ng.bluemix.net/catalog/alchemyapi/>
- Outils de normalisation :
<http://snowball.tartarus.org/index.php>

Plan

- 1 Contexte
- 2 Introduction
- 3 Historique des SRI
- 4 Indexation
 - Des documents aux termes du dictionnaire
 - **Lois de base en RI**
 - Représentation d'un document et de la collection - Index inversé
- 5 Modèles classiques de recherche d'Information
 - Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes
 - Modèle vectoriel
- 6 Bilan

Indexation : Taille du vocabulaire

La taille du vocabulaire M croît exponentiellement en fonction du nombre de mots présents dans une collection donnée T

Loi empirique de Heaps (1978)

$$M = kT^b$$

- M : taille du vocabulaire.
- T : nombre de tokens dans la collection.
- b et k : paramètres dépendant de la collection considérée.

Indexation : Filtrage

Loi de Zipf (1935)

La fréquence d'occurrence $f_c(m)$ d'un mot m dans une collection de documents est inversement proportionnelle à son rang (ordre décroissant des fréquences des mots dans la collection).

$$\forall m, f_c(m) = \frac{\lambda}{\text{rang}(m)}$$

Plan

1 Contexte

2 Introduction

3 Historique des SRI

4 Indexation

- Des documents aux termes du dictionnaire
- Lois de base en RI
- Représentation d'un document et de la collection - Index inversé

5 Modèles classiques de recherche d'Information

- Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes
- Modèle vectoriel

6 Bilan

Indexation : représentation d'un document

Approche sac de mots (Modèle Vectoriel de Salton)

- L'ordre des termes n'a pas d'importance.
- On raisonne en termes de **présence-absence** des termes dans un document ou en terme de **fréquence** de ces termes.
- Représentation des documents et des requêtes comme des vecteurs.
 - ▶ Chaque document peut être vu comme un vecteur de valeurs. On a une composante par terme
 - ▶ Ensemble vectoriel :
 - ★ Les termes = axes (n termes)
 - ★ Documents sont des vecteurs dans cet espace
 - $\vec{d}_i = (w_{i,1}, \dots, w_{i,n})$
 - $w_{i,j}$ poids pour le terme j dans le document i
 - ★ La longueur du vecteur est proportionnelle au poids des termes.

A chaque document d'une collection \mathcal{C} est associé un vecteur \vec{d} dont la dimension est la taille du vocabulaire.

Indexation : représentation d'un document

Pondération des termes

- Approche simple binaire : $w_{i,j} = 0$ si le terme t_j n'apparaît pas dans le document d_i et 1 sinon.
- Approche avec pondération :
 - ▶ Plus le poids est grand, plus il a un impact sur la représentation vectorielle du document.
 - ▶ On veut donner plus de poids aux termes les plus *importants*.
 - ▶ Qu'est ce qu'un terme important ?

Indexation - Représentation d'un document

Pondération des termes

Analogie avec la classification non supervisée

- Les documents sont une collection de C objets
- La requête est une description imprécise d'un sous ensemble de C
- RI : partitionner C en A et non A
- Il faut déterminer :
 - ▶ Quelles sont les caractéristiques qui décrivent au mieux A ?
 - ▶ Quelles sont les caractéristiques qui différencient le mieux A et non A ?
- Pour un document :
 - ▶ Fréquence des termes dans un document.
 - ▶ Fréquence d'un terme dans la collection.

Indexation - Représentation d'un document

Pondération des termes

Dans une requête comme dans un document, les termes n'ont pas tous la même importance.

Modèle Tf-idf (modèle de Salton)

- TF : Term Frequency
 - ▶ Mesure comment un terme décrit le document
- IDF : Inverse Document Frequency
 - ▶ Mesure la répartition du terme dans la collection

Indexation - Représentation d'un document

Pondération des termes

Idée : plus un document contient d'occurrences d'un terme plus il est à propos de ce terme et plus il sera pertinent par rapport à une requête contenant ce terme.

Tf : fréquence du terme

- Nombre d'occurrences de ce terme dans le document
- Soit d le document et t un terme, la fréquence est le nombre d'occurrences du terme t dans d . On le note $tf_{t,d}$.

Indexation - Représentation d'un document

Pondération des termes

Idée : plus un document contient d'occurrences d'un terme plus il est à propos de ce terme et plus il sera pertinent par rapport à une requête contenant ce terme, mais :

- Un document avec $tf = 10$ occurrences d'un terme est plus pertinent qu'un document avec $tf = 1$ occurrence du terme.
- Mais il n'est pas 10 fois plus pertinent.
- La pertinence ne croît pas proportionnellement avec la fréquence des termes.

Echelle dans la pertinence

Variante du tf : logarithme

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,d} & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$tf_{t,d} \rightarrow w_{t,d}$:

0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.

Indexation - Représentation d'un document

Pondération des termes

Idée : des termes très fréquents dans tous les documents ne sont pas si importants. Ils sont moins discriminants. Un terme rare l'est plus.

idf : fréquence inverse de document

- Mesure l'importance du terme dans l'ensemble du corpus
- df_t : fréquence du terme t dans la collection (nombre de documents dans lesquels le terme est présent)
- df_t est une mesure inverse de degré d'information du terme t
- $IDF = \text{Logarithme de l'inverse de la proportion de documents du corpus qui contiennent le terme}$
- $idf_t = \log_{10}\left(\frac{N}{df_t}\right)$
- N : nombre total de documents dans le corpus

Indexation - Représentation d'un document

Pondération des termes

Pondération tf-idf

- Le poids s'obtient en multipliant les deux mesures : $w_{d,t} = tf_{d,t} \times idf_t$
-

$$w_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t}$$

- Un terme apparaissant fréquemment dans le document mais rarement dans le reste de la collection aura un poids élevé
- Beaucoup d'autres méthodes de pondération dans la littérature
- Expérimentalement, tf-idf a de bons résultats

Indexation - Représentation d'un document

Pondération des termes

Pondération tf-idf

- Le poids s'obtient en multipliant les deux mesures : $w_{d,t} = tf_{d,t} \times idf_t$
-

$$w_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t}$$

- Un terme apparaissant fréquemment dans le document mais rarement dans le reste de la collection aura un poids élevé
- Beaucoup d'autres méthodes de pondération dans la littérature
- Expérimentalement, tf-idf a de bons résultats

Indexation - Représentation d'un document

Pondération des termes

Pondération tf-idf

- Le poids s'obtient en multipliant les deux mesures : $w_{d,t} = tf_{d,t} \times idf_t$
-

$$w_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t}$$

- Un terme apparaissant fréquemment dans le document mais rarement dans le reste de la collection aura un poids élevé
- Beaucoup d'autres méthodes de pondération dans la littérature
- Expérimentalement, tf-idf a de bons résultats

Indexation - Représentation d'un document

Pondération des termes

Pondération tf-idf

- Le poids s'obtient en multipliant les deux mesures : $w_{d,t} = tf_{d,t} \times idf_t$
-

$$w_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t}$$

- Un terme apparaissant fréquemment dans le document mais rarement dans le reste de la collection aura un poids élevé
- Beaucoup d'autres méthodes de pondération dans la littérature
- Expérimentalement, tf-idf a de bons résultats

Indexation - Représentation d'un document

Pondération des termes

Pondération tf-idf

- Le poids s'obtient en multipliant les deux mesures : $w_{d,t} = tf_{d,t} \times idf_t$
-

$$w_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t}$$

- Un terme apparaissant fréquemment dans le document mais rarement dans le reste de la collection aura un poids élevé
- Beaucoup d'autres méthodes de pondération dans la littérature
- Expérimentalement, tf-idf a de bons résultats

Indexation - Représentation d'un document

Pondération des termes

De nombreuses variantes de pondération dans la littérature selon le schéma :

$$w_{t,d} = n_d \times p_{tf_{t,d}} \times p_{df_t}$$

- n_d facteur de normalisation dépendant de la taille du document.
- $p_{tf_{t,d}}$ poids fondé sur la fréquence du terme dans le document.
- p_{idf_t} poids fondé sur la fréquence du terme dans la collection.

Indexation - Représentation d'un document

Pondération des termes

Quelques pondérations classiques

$$w_{t,d} = n_d \times p_{tf_{t,d}} \times p_{df_t}$$

$p_{tf_{t_i,d}}$	$p_{df_{t_i}}$	n_d
$tf_{t_i,d}$	1	1
$\frac{1 + \ln(tf_{t_i,d})}{1 + \ln(\text{moy_tf}(d))}$	idf_{t_i}	$\frac{1}{\ d\ } = \frac{1}{\sqrt{\sum_{i=1}^V w_{id}}}$
$\begin{cases} 1 + \ln(tf_{t_i,d}) & \text{si } tf_{t_i,d} > 0, \\ 0 & \text{sinon.} \end{cases}$		
$0.5 + 0.5 \times \frac{tf_{t_i,d}}{tf_{\max,d}}$	$\max\{0, \ln \frac{N - df_{t_i}}{df_{t_i}}\}$	$\frac{1}{(Char_d)^\alpha}, 0 < \alpha < 1$
$\begin{cases} 1 & \text{si } tf_{t_i,d} > 0, \\ 0 & \text{sinon.} \end{cases}$		

FIGURE: Source : Amini - Gaussier

Indexation - Représentation de la collection

Représentation d'une collection de documents

- Représentation par une matrice terme-document
- Une entrée de la matrice est le poids du terme dans le document

Indexation - Représentation de la collection

Exemple

- Recherche des pièces de Shakespeare qui contiennent les mots BRUTUS, CESAR mais pas CALPURNIA.
- Une méthode simple : le *grep* d'UNIX.
- Mais :
 - ▶ Lent pour des corpus volumineux
 - ▶ Des requêtes complexes ne sont pas faciles : NOT CALPURNIA, ROMANS à côté de PAYSAN
 - ▶ Pas de classement des résultats

Indexation - Représentation de la collection

Incidence Terme-Document

Termes : unités d'indexation

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Brutus AND Caesar BUT NOT Calpurnia

1 if play contains word, 0 otherwise

FIGURE: From <http://www.informationretrieval.org>

Indexation - Représentation de la collection

Incidence Terme-Document

Réponse à la requête :

- On prend les vecteurs d'incidence pour BRUTUS et CAESAR et le complément de CALPURNIA
- Soit $110100 \text{ AND } 110111 \text{ AND } 101111 = 100100$ (ET bit à bit)

Indexation - Représentation de la collection

Réponse à la requête

Exemple

Anthony et Cléopatre et Hamlet

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to Domitius Enobarbus]: Why, Enobarbus,
When Antony found Julius Caesar dead,
He cried almost to roaring; and he wept
When at Philippi he found Brutus slain.

Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius Caesar: I was killed i' the
Capitol; Brutus killed me.

FIGURE: From <http://www.informationretrieval.org>

Indexation : index inversé

Matrice terme-incidence : limitations

- Dans le cas de grands corpus, la matrice terme-incidence est de très grande dimension et on ne peut donc pas l'utiliser en pratique.
- C'est aussi dans la plupart des cas une matrice creuse.

Index Inversé

- L'index inversé est un nouveau mode de représentation du texte (concept fondateur de l'IR).

Indexation : index inversé

Matrice terme-incidence : Ordre de grandeur

- On considère une collection de $N = 10^6$ documents qui ont en moyenne 1000 mots soit en tout 10^9 mots.
- On considère qu'il y a $M = 500000$ termes distincts dans la collection.
- 500000×10^6 entrées dans la matrice d'incidence
- La matrice contient beaucoup de 0.
- ⇒ Ne stocker que les 1.

Indexation : index inversé

Objectifs de l'index

- Stocker de grandes quantités d'information en limitant autant que possible l'espace utilisé.
- Extraire les éléments nécessaires à la recherche d'information visée.
- Permettre un accès efficace aux index pendant la recherche.
- Permettre de gérer la spécificité de certaines collections comme par exemple les collections dynamiques.

Indexation : index inversé

Principe

Associer aux termes les documents qui les contiennent. Chaque document possède un identifiant unique.

- Sauvegarde d'un **dictionnaire** de termes.
- Pour chaque terme, une liste qui enregistre dans quel document le terme apparaît, i.e. liste des *postings* (occurrences) (Occurrences des termes dans un document et souvent aussi la position dans le document).

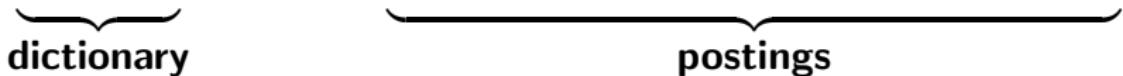
Indexation : index inversé

BRUTUS	→	1	2	4	11	31	45	173	174
--------	---	---	---	---	----	----	----	-----	-----

CAESAR	→	1	2	4	5	6	16	57	132	...
--------	---	---	---	---	---	---	----	----	-----	-----

CALPURNIA	→	2	31	54	101
-----------	---	---	----	----	-----

⋮



dictionary

postings

D'après Hinrich Schütze

Indexation : index inversé

BRUTUS	→	1	2	4	11	31	45	173	174
--------	---	---	---	---	----	----	----	-----	-----

CAESAR	→	1	2	4	5	6	16	57	132	...
--------	---	---	---	---	---	---	----	----	-----	-----

CALPURNIA	→	2	31	54	101
-----------	---	---	----	----	-----

⋮



dictionary

postings

D'après Hinrich Schütze

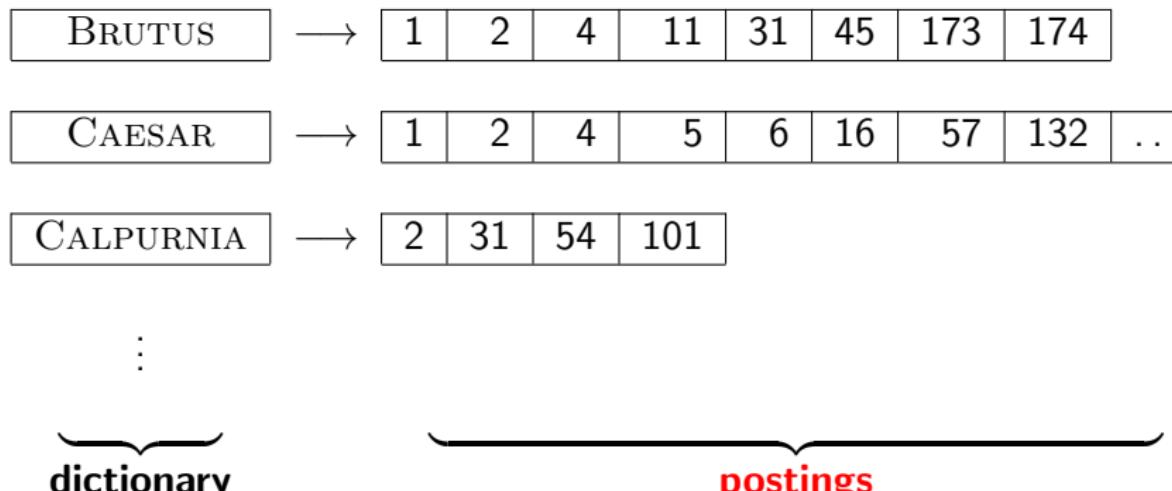
Indexation : index inversé

BRUTUS	→	1	2	4	11	31	45	173	174
--------	---	---	---	---	----	----	----	-----	-----

CAESAR	→	1	2	4	5	6	16	57	132	...
--------	---	---	---	---	---	---	----	----	-----	-----

CALPURNIA	→	2	31	54	101
-----------	---	---	----	----	-----

⋮



dictionary

postings

D'après Hinrich Schütze

Construction de l'index inversé

Principe

- Association d'un identifiant unique ($docID$) à chaque document de la collection.
- Construction d'une liste de paires ($terme, docID$).
- Tri de cette liste (ordre alphabétique) et fusion des occurrences multiples d'un même terme dans un document.
- Groupement des instances d'un même terme : liste de postings.
- Stockage d'informations statistiques dans le dictionnaire : nombre de documents contenant un terme ($document frequency$).

Construction de l'index inversé

- ① Collecter les documents à indexer :

Friends, Romans, countrymen. So let it be with Caesar ...

- ② Tokenization du texte : chaque document est une liste de tokens :

Friends Romans countrymen So ...

- ③ Pré-traitement linguistique : liste de tokens normalisés (termes d'index) :
- friend roman countryman so ...

- ④ Indexer les documents : création d'un index inversé (dictionnaire-liste de postings)

Construction de l'index inversé

Exemple

Doc 1

I did enact Julius Caesar: I was killed
i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus
hath told you Caesar was ambitious:

FIGURE: Source : <http://www.informationretrieval.org>

Tokenization et pré-traitement

Doc 1. I did enact Julius Caesar :
I was killed i' the Capitol; Brutus
killed me.

Doc 2. So let it be with Caesar. The
noble Brutus hath told you Caesar
was ambitious :



Doc 1. i did enact julius caesar i was
killed i' the capitol brutus killed me

Doc 2. so let it be with caesar the
noble brutus hath told you caesar was
ambitious

Génération des postings

Doc 1. i did enact julius caesar i was
killed i' the capitol brutus killed me
Doc 2. so let it be with caesar the
noble brutus hath told you caesar was
ambitious

term	docID
i	1
did	1
enact	1
julius	1
caesar	1
i	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Tri des postings

term	docID	term	docID
i	1	ambitious	2
did	1	be	2
enact	1	brutus	1
julius	1	brutus	2
caesar	1	capitol	1
i	1	caesar	1
was	1	caesar	2
killed	1	caesar	2
i'	1	did	1
the	1	enact	1
capitol	1	hath	1
brutus	1	i	1
killed	1	i	1
me	1	i'	1
so	2	⇒	
let	2	it	2
it	2	julius	1
be	2	killed	1
with	2	killed	1
caesar	2	let	2
the	2	me	1
noble	2	noble	2
brutus	2	so	2
hath	2	the	1
told	2	the	2
you	2	told	2
caesar	2	you	2
was	2	was	1
ambitious	2	was	2
		with	2

Création des listes de postings, détermination de la fréquence des documents

term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
i	1
i	1
i'	1
it	2
it	1
julius	1
killed	1
killed	1
let	2
me	1
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	2
was	1
was	2
with	2



term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	1	→	1 → 2
caesar	2	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i	1	→	1
i'	1	→	1
it	1	→	2
it	2	→	1
julius	1	→	1
killed	1	→	1
killed	1	→	2
let	1	→	1
me	1	→	1
me	1	→	2
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
the	1	→	2
the	2	→	2
told	2	→	1 → 2
you	1	→	2
was	2	→	1 → 2
was	1	→	2
was	2	→	2
with	1	→	2

Séparer le résultat en dictionnaire et postings

BRUTUS →

1	2	4	11	31	45	173	174
---	---	---	----	----	----	-----	-----

CAESAR →

1	2	4	5	6	16	57	132	...
---	---	---	---	---	----	----	-----	-----

CALPURNIA →

2	31	54	101
---	----	----	-----

⋮


dictionary


postings file

Construction de l'index inversé

Exemple

term	docID	term	docID	term	doc. freq.	→	postings lists
I	1	ambitious	2	ambitious	1	→	2
did	1	be	2	be	1	→	2
enact	1	brutus	1	brutus	2	→	1 → 2
julius	1	brutus	2	capitol	1	→	1
caesar	1	capitol	1	caesar	2	→	1 → 2
I	1	caesar	1	capitol	1	→	1
was	1	caesar	2	caesar	2	→	1 → 2
killed	1	caesar	2	did	1	→	1
i'	1	did	1	enact	1	→	1
the	1	enact	1	hath	1	→	2
capitol	1	hath	1	I	1	→	1
brutus	1	I	1	i'	1	→	1
killed	1	I	1	it	1	→	2
me	1	i'	1	julius	1	→	1
so	2	it	2	killed	1	→	1
let	2	julius	1	let	1	→	2
it	2	killed	1	me	1	→	1
be	2	killed	1	noble	1	→	2
with	2	let	2	so	1	→	2
caesar	2	me	1	the	2	→	1 → 2
the	2	noble	2	told	1	→	2
noble	2	so	2	you	1	→	2
brutus	2	the	1	was	2	→	1 → 2
hath	2	the	2	with	1	→	2
told	2	told	2				
you	2	you	2				
caesar	2	was	1				
was	2	was	2				
ambitious	2	with	2				

Construction de l'index inversé

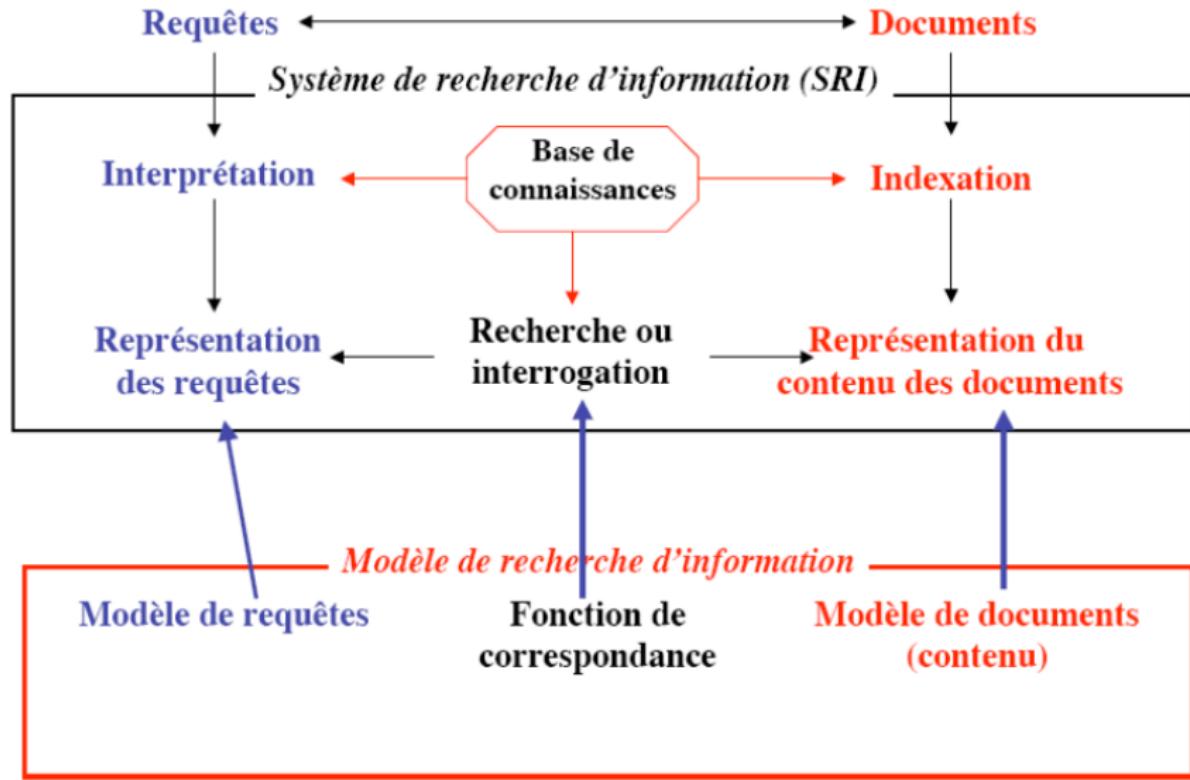
Questions

- Pourquoi conserver la fréquence ?
- Comment construire cet index de façon efficace et économique ?
Quelles structures de données ?
- Comment le conserver ?
- Comment faire une requête sur cet index ?

Plan

- 1 Contexte
- 2 Introduction
- 3 Historique des SRI
- 4 Indexation
 - Des documents aux termes du dictionnaire
 - Lois de base en RI
 - Représentation d'un document et de la collection - Index inversé
- 5 Modèles classiques de recherche d'Information
 - Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes
 - Modèle vectoriel
- 6 Bilan

Modélisation



Modélisation

Un modèle de recherche spécifie :

- La représentation des documents
- La représentation de la requête
- La fonction de recherche
- Une notion de pertinence

Description formelle de la RI

Modèle

$$\langle D, Q, F, R(q_i, d_i) \rangle$$

- D - ensemble des vues logiques des documents
- Q - ensemble des vues logiques des besoins de l'utilisateur : requêtes
- F - framework de modélisation des documents, des requêtes et de leurs relations
- $R(q_i, d_j)$ - fonction de ranking : associe un nombre réel à une requête $q_i \in Q$ et une représentation de document $d_j \in D$. Il permet de définir un ordre parmi les documents en rapport avec la requête q_i

Plan

1 Contexte

2 Introduction

3 Historique des SRI

4 Indexation

- Des documents aux termes du dictionnaire
- Lois de base en RI
- Représentation d'un document et de la collection - Index inversé

5 Modèles classiques de recherche d'Information

• Modèle booléen

- Traitement des requêtes booléennes
- Optimisation de requêtes
- Modèle vectoriel

6 Bilan

Modèle Booléen

- Modèle simple basé sur la théorie des ensemble et l'algèbre de boole.
- **Poids binaires** : présence ou absence des termes.
- **Pertinence binaire** : un document est soit pertinent soit non-pertinent (**recherche exacte**).
- La requête est une expression booléenne qui s'exprime donc avec des opérateurs logiques.
- Un document est pertinent si et seulement si sa représentation satisfait la formule booléenne correspondant à la requête.

Modèle Booléen

Principe

Mise en correspondance **exacte**

- Modèle :
 - ▶ Les documents sont trouvés si ils satisfont une expression booléenne
 - ▶ Pas d'ordre dans les documents retrouvés
- Documents : Ensemble de mots
- Termes : Présents ou absents dans le document - $w_{ij} \in (0, 1)$
- Requêtes : Expressions booléennes
 - ▶ Termes d'index connectés par AND(\wedge), OR(\vee), NOT(\neg)
 - ▶ Une requête est une disjonction de vecteurs conjonctifs (forme normale disjonctive)

Requête booléenne avec un index inversé

BRUTUS AND CALPURNIA

- On localise BRUTUS dans le dictionnaire
- On recherche son posting
- On localise CALPURNIA dans le dictionnaire
- On recherche son posting
- On fusionne (ou intersection) les deux postings

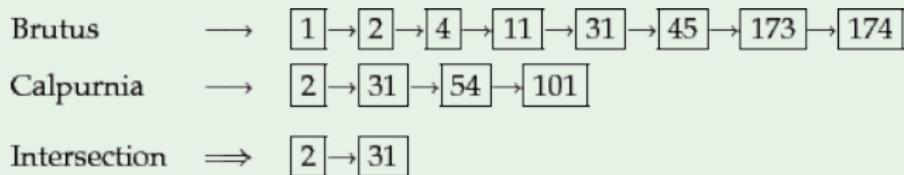


FIGURE: Source : <http://www.informationretrieval.org>

Fusion (Intersection) de deux postings

BRUTUS → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA → 2 → 31 → 54 → 101

Intersection ⇒

Fusion (Intersection) de deux postings

BRUTUS → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA → 2 → 31 → 54 → 101

Intersection ⇒

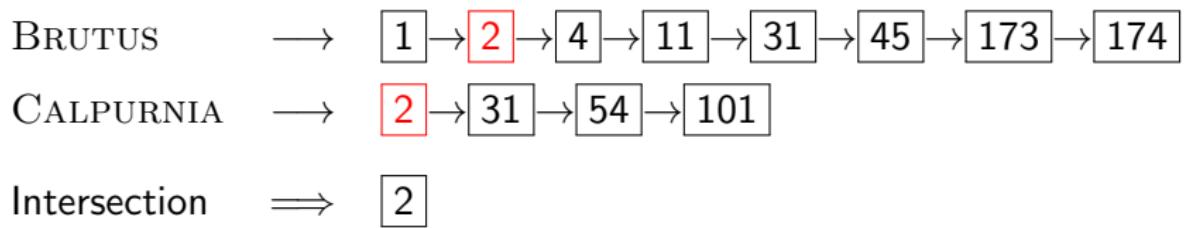
Fusion (Intersection) de deux postings

BRUTUS → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

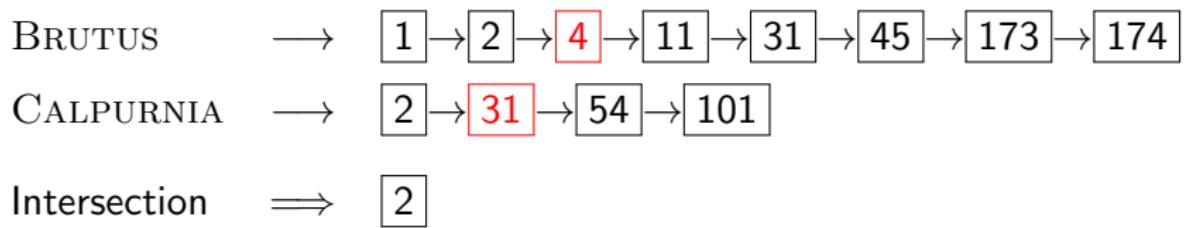
CALPURNIA → 2 → 31 → 54 → 101

Intersection →

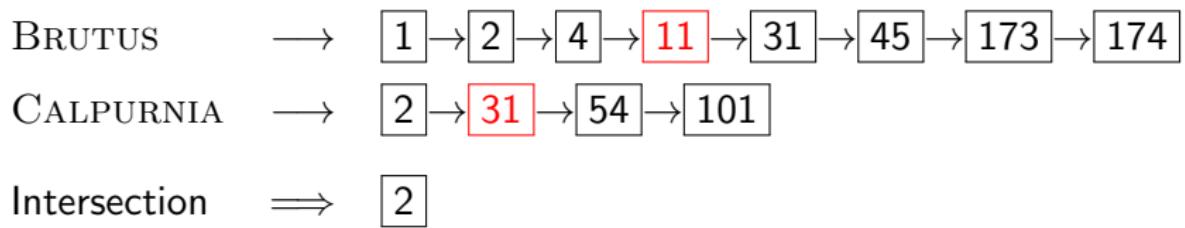
Fusion (Intersection) de deux postings



Fusion (Intersection) de deux postings



Fusion (Intersection) de deux postings



Fusion (Intersection) de deux postings

BRUTUS → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA → 2 → 31 → 54 → 101

Intersection ⇒ 2

Fusion (Intersection) de deux postings

BRUTUS → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA → 2 → 31 → 54 → 101

Intersection ⇒ 2 → 31

Fusion (Intersection) de deux postings

BRUTUS →
$$1 \rightarrow 2 \rightarrow 4 \rightarrow 11 \rightarrow 31 \rightarrow 45 \rightarrow 173 \rightarrow 174$$

CALPURNIA →
$$2 \rightarrow 31 \rightarrow 54 \rightarrow 101$$

Intersection ⇒
$$2 \rightarrow 31$$

Fusion (Intersection) de deux postings

BRUTUS \rightarrow $1 \rightarrow 2 \rightarrow 4 \rightarrow 11 \rightarrow 31 \rightarrow 45 \rightarrow 173 \rightarrow 174$
CALPURNIA \rightarrow $2 \rightarrow 31 \rightarrow 54 \rightarrow 101$
Intersection \implies $2 \rightarrow 31$

Fusion (Intersection) de deux postings

BRUTUS \rightarrow
$$1 \rightarrow 2 \rightarrow 4 \rightarrow 11 \rightarrow 31 \rightarrow 45 \rightarrow 173 \rightarrow 174$$

CALPURNIA \rightarrow
$$2 \rightarrow 31 \rightarrow 54 \rightarrow 101$$

Intersection \implies
$$2 \rightarrow 31$$

Fusion (Intersection) de deux postings

- Opération cruciale pour une recherche rapide des documents
- Parcours simultané des deux listes
 - ▶ A chaque étape, comparaison des deux $docID$ pointés
 - ▶ Si identique, on ajoute à la liste résultat
 - ▶ Sinon, on avance le pointeur qui pointe sur le plus petit $docID$

Fusion (Intersection) de deux postings

Avec le TDA List vu en cours d'algo prog

Algorithm 1: INTERSECT(L_1, L_2)

```
1 answer ← emptyList()
2 while ≠ isEmpty( $L_1$ ) and ≠ isEmpty( $L_2$ ) do
3     if head( $L_1$ ) == head( $L_2$ ) then
4         answer ← add(answer, head( $L_1$ ))
5          $L_1$  ← rest( $L_1$ )
6          $L_2$  ← rest( $L_2$ )
7     else if head( $L_1$ ) < head( $L_2$ ) then
8          $L_1$  ← rest( $L_1$ )
9     else
10         $L_2$  ← rest( $L_2$ )
11 return answer
```

Complexité $O(x + y)$ avec x et y les tailles des deux listes.

Traitement d'une requête booléenne

Logical operator	Set (or List) operation
AND	Intersection
OR	Union
NOT	Complement

Exemple

FRANCE → 1 → 2 → 3 → 4 → 5 → 7 → 8 → 9 → 11 → 12 → 13 → 14 → 15

PARIS → 2 → 6 → 10 → 12 → 14

LEAR → 12 → 15

Requête : ((paris AND NOT france) OR lear)

Exemple de SRI reposant sur le modèle booléen

WestLaw

<http://web2.westlaw.com/>

- SRI dans le domaine de la loi
- 10 terabytes de données, plus de 700000 utilisateurs
- Utilisation des requêtes booléennes
- Requêtes structurées

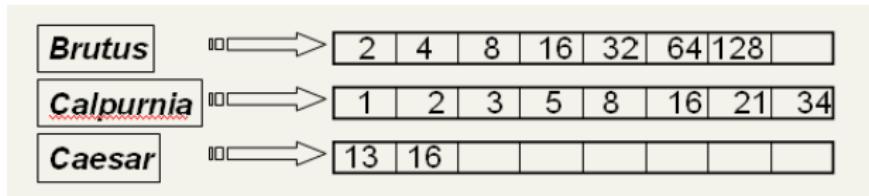
Modèle booléen : rapide bilan

- Un modèle de recherche **simple** et **transparent**
- Un modèle de recherche pouvant répondre à toute requête exprimée comme une expression booléenne :
 - ▶ Opérateurs : AND, NOT, OR.
 - ▶ Documents = un ensemble de termes.
 - ▶ Recherche exacte : les documents contiennent la requête ou non.
- Premiers moteurs de recherche commerciaux basés sur ce modèle (AltaVista, Lycos,...)
- Utilisé par certains moteurs professionnels : ex : Westlaw
- D'autres moteurs de recherche : spotlight, email, intranet, ...

Améliorations du modèle booléen

Optimisation de requête

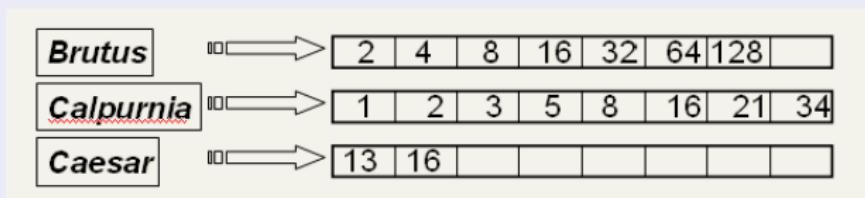
- Ordre dans le traitement de la requête
- Exemple : requête de type AND de t termes
- Pour chaque terme, on cherche son posting et on les fusionne



Améliorations du modèle booléen

Optimisation de requête

- Une solution est de traiter la requête par ordre de fréquence croissante
- Intérêt de mémoriser la fréquence dans le dictionnaire



On exécute (CAESAR AND BRUTUS) AND CALPURNIA

Optimisation de requête

Algorithme d'intersection pour les requêtes conjonctives

Algorithm 2: INTERSECT($\langle t_1, \dots t_n \rangle$)

```
1 terms ← SORTBYINCREASINGFREQUENCY( $\langle t_1, \dots t_n \rangle$ )
2 result ← postings(head(terms))
3 terms ← rest(terms)
4 while  $\neq isEmpty(terms)$  and  $\neq isEmpty(result)$  do
5   result ← INTERSECT(result, postings(head(terms)))
6   terms ← rest(terms)
7 return result
```

Optimisation plus générale

Exemple

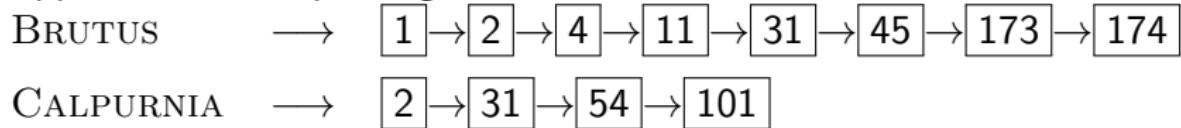
(MADDING OR CROWD) AND (IGNOBLE OR STRIFE)

Principe

- On cherche la fréquence pour chaque terme
- On estime la taille de chaque OR par la somme de ses fréquences
- On traite la requête par ordre croissant de la taille des OR

Optimisation de requête : skip list ou fusion par sauts

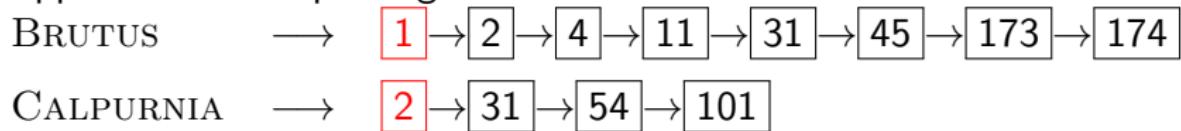
Rappel : fusion de postings



Intersection →

Optimisation de requête : skip list ou fusion par sauts

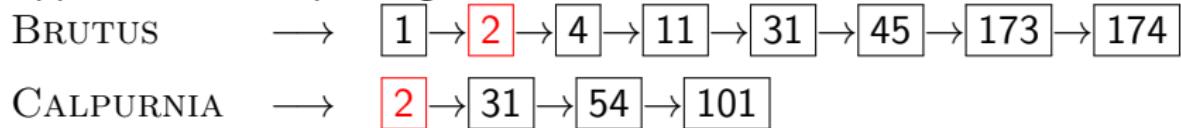
Rappel : fusion de postings



Intersection →

Optimisation de requête : skip list ou fusion par sauts

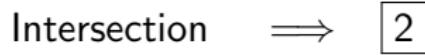
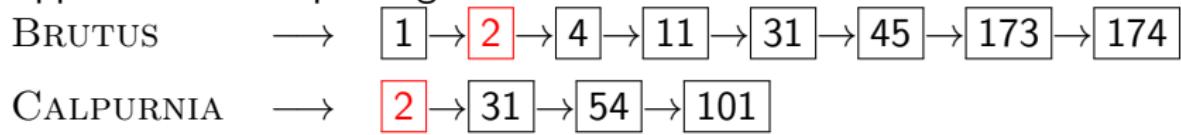
Rappel : fusion de postings



Intersection →

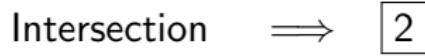
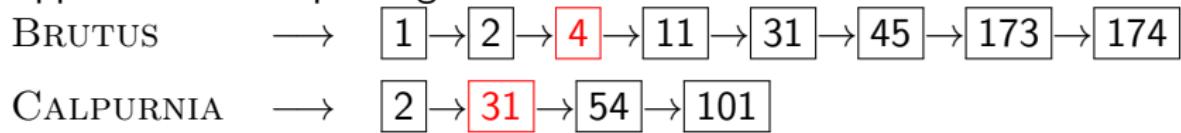
Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings



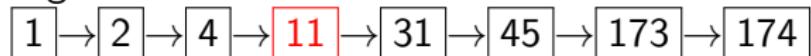
Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings



Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings

BRUTUS →  1 → 2 → 4 → 11 → 31 → 45 → 173 → 174
CALPURNIA →  2 → 31 → 54 → 101

Intersection →  2

Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings

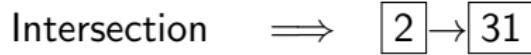
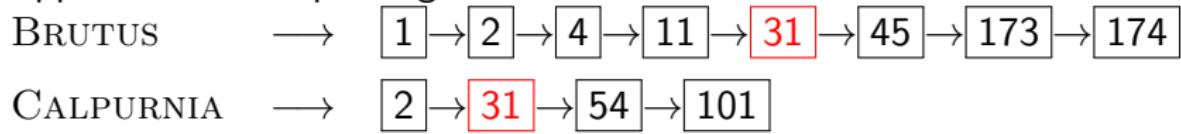
BRUTUS → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA → 2 → 31 → 54 → 101

Intersection → 2

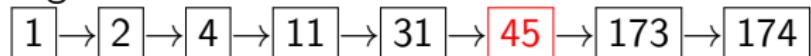
Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings



Optimisation de requête : skip list ou fusion par sauts

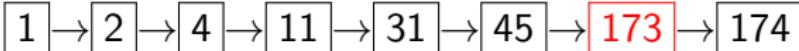
Rappel : fusion de postings

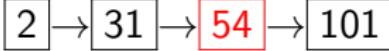
BRUTUS → 
CALPURNIA → 

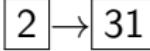
Intersection → 

Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings

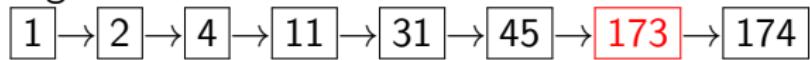
BRUTUS → 

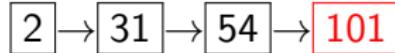
CALPURNIA → 

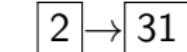
Intersection → 

Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings

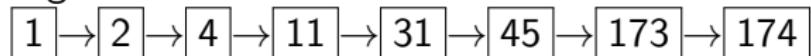
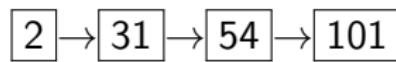
BRUTUS →  1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA →  2 → 31 → 54 → 101

Intersection →  2 → 31

Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings

BRUTUS → 
CALPURNIA → 

Intersection → 

Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings

BRUTUS → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

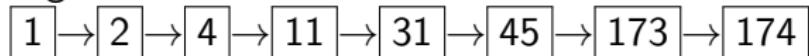
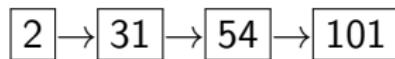
CALPURNIA → 2 → 31 → 54 → 101

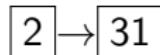
Intersection → 2 → 31

- Complexité linéaire (en taille des liste de postings)

Optimisation de requête : skip list ou fusion par sauts

Rappel : fusion de postings

BRUTUS → 
CALPURNIA → 

Intersection → 

- Complexité linéaire (en taille des liste de postings)
- Peut-on faire mieux ?

Skip lists (Fusion par sauts)

- Ajouts de **pointeurs de saut (skip pointers)** aux listes de postings pour sauter les entrées qui ne sont pas concernées.

Skip lists (Fusion par sauts)

- Ajouts de **pointeurs de saut (skip pointers)** aux listes de postings pour sauter les entrées qui ne sont pas concernées.
- Accélérer (rendre plus efficace) le calcul des intersections notamment quand on fusionne des listes de postings pour des termes fréquents et des termes non fréquents.

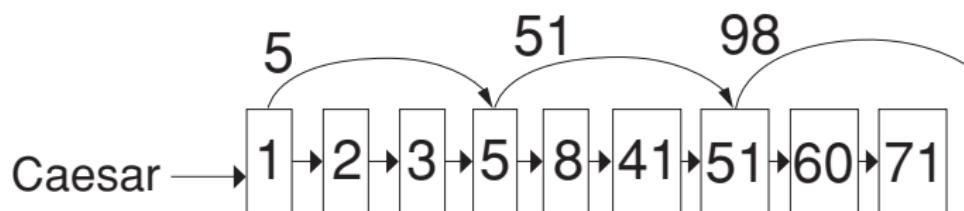
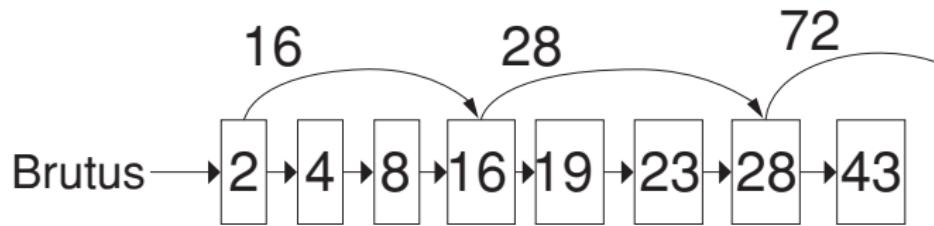
Skip lists (Fusion par sauts)

- Ajouts de **pointeurs de saut (skip pointers)** aux listes de postings pour sauter les entrées qui ne sont pas concernées.
- Accélérer (rendre plus efficace) le calcul des intersections notamment quand on fusionne des listes de postings pour des termes fréquents et des termes non fréquents.
- Les listes de postings peuvent contenir plusieurs millions d'entrées

Skip lists (Fusion par sauts)

- Ajouts de **pointeurs de saut (skip pointers)** aux listes de postings pour sauter les entrées qui ne sont pas concernées.
- Accélérer (rendre plus efficace) le calcul des intersections notamment quand on fusionne des listes de postings pour des termes fréquents et des termes non fréquents.
- Les listes de postings peuvent contenir plusieurs millions d'entrées
- Comment répartir les pointeurs ?

Skip lists : exemple



Parcours jusqu'à 8 puis comparaison des valeurs suivantes et des valeurs suivantes après saut. Comme $28 < 41$, on peut effectuer le saut en évitant donc 19 et 23.

Intersection avec des pointeurs de saut : algorithme

Algorithm 3: INTERSECT_WITH_SKIPS(L_1, L_2)

```
1 answer ← emptyList()
2 while ≠ isEmpty( $L_1$ ) and ≠ isEmpty( $L_2$ ) do
3     if head( $L_1$ ) == head( $L_2$ ) then
4         answer ← add(answer, head( $L_1$ ))
5          $L_1$  ← rest( $L_1$ )
6          $L_2$  ← rest( $L_2$ )
7     else if head( $L_1$ ) < head( $L_2$ ) then
8         if hasSkip( $L_1$ ) and head(skip( $L_1$ )) ≤ head( $L_2$ ) then
9             while hasSkip( $L_1$ ) and head(skip( $L_1$ )) ≤ head( $L_2$ ) do
10                 $L_1$  ← skip( $L_1$ )
11            else
12                 $L_1$  ← rest( $L_1$ )
13        else if hasSkip( $L_2$ ) and head(skip( $L_2$ )) ≤ head( $L_1$ ) then
14            while hasSkip( $L_2$ ) and head(skip( $L_2$ )) ≤ head( $L_1$ ) do
15                 $L_2$  ← skip( $L_2$ )
16            else
17                 $L_2$  ← rest( $L_2$ )
18 return answer
```

Skip Lists : comment répartir les sauts ?

- Uniquement valable pour des requêtes ET.
- Compromis : nombre d'éléments à sauter versus fréquence des sauts.
- Plus de sauts : chaque saut ne saute que peu d'éléments, mais il faut sauter de manière fréquente.
- Moins de sauts : chaque saut saute beaucoup d'éléments mais on ne les utilise pas souvent.
- Heuristique simple = pour une liste de taille P , utiliser \sqrt{P} pointeurs (ne tient pas compte de la distribution des termes de la requête).

Fusion par sauts : bilan

- Gain de temps pour la fusion.
- Mais :
 - ▶ Plus de pointeurs donc plus de place dans l'index.
 - ▶ Plus d'entrée/sortie pour lire l'index si il est stocké sur le disque.
 - ▶ Complexité de mise en place en cas d'index dynamique.

Améliorations du modèle booléen

Requête plus riche

- Prise en compte de phrase : Ecole Centrale
- Prise en compte de la proximité : TROUVER Gates A COTE DE Microsoft
- Prise en compte d'information structurelle : auteur, ...

Problème : requête *phrase*

- Requête : *Ecole Centrale Paris*, comme un tout.
- Le document : *je suis allée à l'école à Paris avant de travailler dans une centrale* ne devrait pas correspondre.
- Des requêtes très compréhensibles pour l'utilisateur : environ 10% des requêtes du web sont des phrases.
- Stocker les numéros de documents dans les index inversés ne suffit pas.
- Deux possibilités :
 - ▶ Index bi-mot (n-grams).
 - ▶ Index de position.

Recherche de groupes de mots : n-gramme

n-gramme

- n-gramme : une sous-séquence de n éléments extraite d'une séquence donnée.
- Ici, n-gramme de mots :
 - ▶ uni-gramme : tous les mots
 - ▶ bi-gramme : une sous-séquence de deux éléments
 - ▶ ...
- Notion différente de celle du groupe de mots du point de vue linguistique :
 - ▶ Combien de bi-grammes théoriquement possibles pour m mots uniques dans un vocabulaire ?
 - ▶ Jusqu'à quelle valeur de n pour couvrir raisonnablement des besoins d'un utilisateur de moteur de recherche ?

Recherche de groupes de mots : n-gramme

Index de bi-grammes

Indexer (en plus des mots simples) toutes les paires de termes du texte.

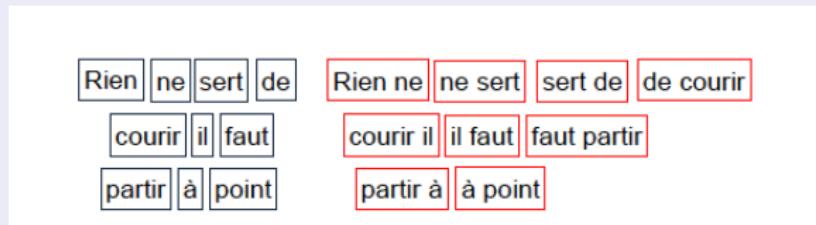


FIGURE: D'après X. Tannier

On considère chaque bi-gramme comme un terme du dictionnaire.

Recherche de groupes de mots : n-gramme

Index de bi-grammes

- Dictionnaire beaucoup plus gros et index vite ingérable.
- Impraticable pour $n > 2$
- On peut utiliser les index de bi-grammes dans certaines conditions ou pour certains groupes de mots mais c'est n'est pas la solution standard pour la recherche de groupes de mots.

Recherche de groupes de mots : index de position

Index de position

Idée : dans les listes de documents de l'index, on ajoute la position de chaque occurrence de terme dans le document.

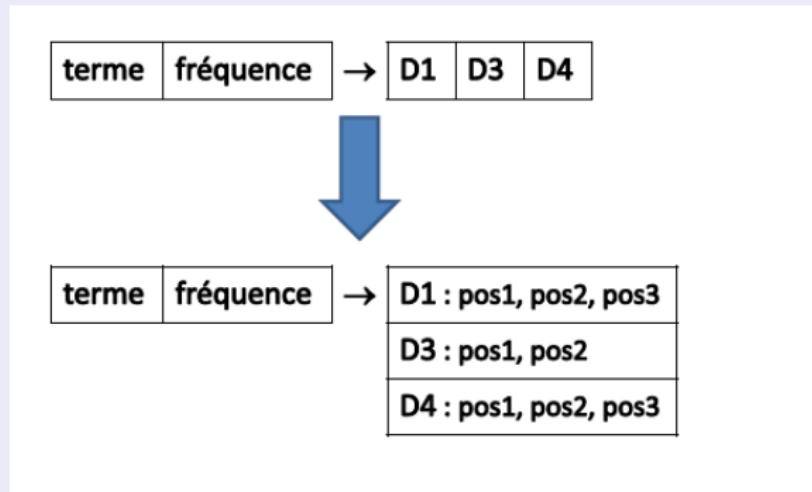


FIGURE: D'après X. Tannier

Recherche de groupes de mots : index de position

Parcours de l'index de position

- Extraction des entrées du dictionnaire.
- Utilisation récursive de l'algorithme de fusion pour les documents puis pour les positions.
- Comparaison incrémentale plutôt que égalité stricte pour les positions.

Exemple d'Index inversé

family	d_1, d_3, d_5, d_6
football	d_4
jaguar	$d_1, d_2, d_3, d_4, d_5, d_6$
new	d_1, d_2, d_5
rule	d_6
us	d_4, d_5
world	d_1

Exemple d'Index inversé avec information de position

family	$d_1/11, d_3/10, d_5/16, d_6/4$
football	$d_4/8$
jaguar	$d_1/2, d_2/1, d_3/2, d_4/3, d_5/4, d_6/8 + 13$
new	$d_1/5, d_2/5, d_5/15$
rule	$d_6/3$
us	$d_4/7, d_5/11$
world	$d_1/6$

...

- Requête phrase
- Opérateur NEAR

Exemple d'Index inversé avec information de position et pondération

family	$d_1/11/.13, d_3/10/.13, d_6/4/.08, d_5/16/.07$
football	$d_4/8/.47$
jaguar	$d_1/2/.04, d_2/1/.04, d_3/2/.04, d_4/3/.04, d_6/8 + 13/.04,$ $d_5/4/.02$
new	$d_2/5/.24, d_1/5/.20, d_5/15/.10$
rule	$d_6/3/.28$
us	$d_4/7/.30, d_5/11/.15$
world	$d_1/6/.47$

Modèle booléen : bilan

Avantages

- Un modèle transparent et simple à comprendre.
- La raison de sélection d'un document est claire : satisfaction d'une formule logique.
- Très adapté pour des SRI spécialistes avec un vocabulaire restreint et contraint.

Inconvénients

- Difficile d'exprimer des requêtes booléennes longues.
- La représentation binaire est peu efficace :
 - ▶ Il est connu que la pondération des termes améliore les résultats (Modèle booléen étendu)
- Pas d'ordonnancement des résultats (souvent préférable quand la liste de résultats retournés est grande).

Plan

1 Contexte

2 Introduction

3 Historique des SRI

4 Indexation

- Des documents aux termes du dictionnaire
- Lois de base en RI
- Représentation d'un document et de la collection - Index inversé

5 Modèles classiques de recherche d'Information

- Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes

• Modèle vectoriel

6 Bilan

Modèle Vectoriel

- Revient sur deux défauts du modèle booléen : la pondération et la pertinence binaire.
- Modèle *statistique* :
 - ▶ Aspect quantitatif des termes et des documents.
 - ▶ Degré de similarité entre une requête et un document.
- Importance du *classement* :
 - ▶ Ne montrer que les k premiers documents.
 - ▶ Classer les documents selon leur pertinence par rapport à la requête.
 - ▶ Un score entre $[0, 1]$ qui mesure comment la requête et un document correspondent entre eux.

Première idée = coefficient de Jaccard

Similarité entre ensembles

Indice de Jaccard

- Mesure de la similarité entre ensembles : rapport entre le cardinal de l'intersection et de l'union.
- Soit A et B deux ensembles, la mesure de Jaccard est définie par :

$$JACCARD(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

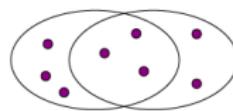
avec $A \neq \emptyset$ or $B \neq \emptyset$

- $JACCARD(A, A) = 1$
- $JACCARD(A, B) = 0$ if $A \cap B = 0$
- Renvoie un nombre entre 0 et 1.

Similarité entre ensembles : indice de Jaccard

Distance de Jaccard

$$d(A, B) = 1 - JACCARD(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$



- Intersection = 3, union = 8
- Similarité = $\frac{3}{8}$
- Distance = $\frac{5}{8}$

Similarité entre ensembles : indice de Jaccard

Exemple

- Coefficient de Jaccard pour
 - ▶ Requête : “ides of March”
 - ▶ Document “Caesar died in March”
 - ▶ $\text{JACCARD}(q, d) = 1/6$

Limitations de l'indice de Jaccard

- Ne considère pas la fréquence des termes.
- Les termes rares sont plus informatifs que les termes fréquents : ne prend pas en compte cette information

Idée 2 : Modèle Vectoriel

Principe

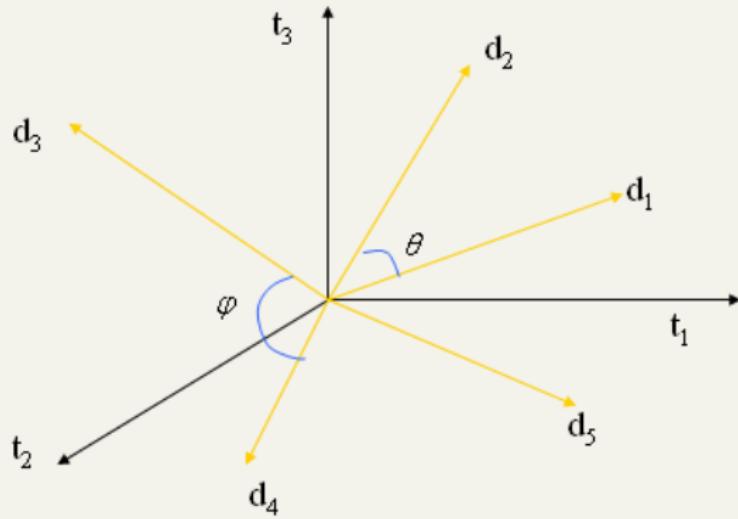
Représentation des documents et des requêtes comme des vecteurs

- Chaque document peut être vu comme un vecteur de valeurs. On a une composante par terme
- Ensemble vectoriel :
 - ▶ Les termes = axes (n termes) - espace euclidien à n dimensions.
 - ▶ Documents sont des vecteurs dans cet espace
 - ▶ $\vec{d}_i = (w_{i,1}, \dots, w_{i,n})$
 - ▶ $w_{i,j}$ poids pour le terme j dans le document i
 - ▶ La longueur du vecteur est proportionnelle au poids des termes.
- Mesure de **similarité** : plus deux représentations contiennent les mêmes éléments, plus la probabilité qu'elles représentent la même information est élevée.
- La pertinence du document correspond au degré de similarité entre le vecteur de la requête et celui du document.

Modèle Vectoriel

Idée

Les documents qui sont proches dans l'espace vectoriel traitent de sujets similaires



Modèle Vectoriel

Quelle mesure de similarité ?

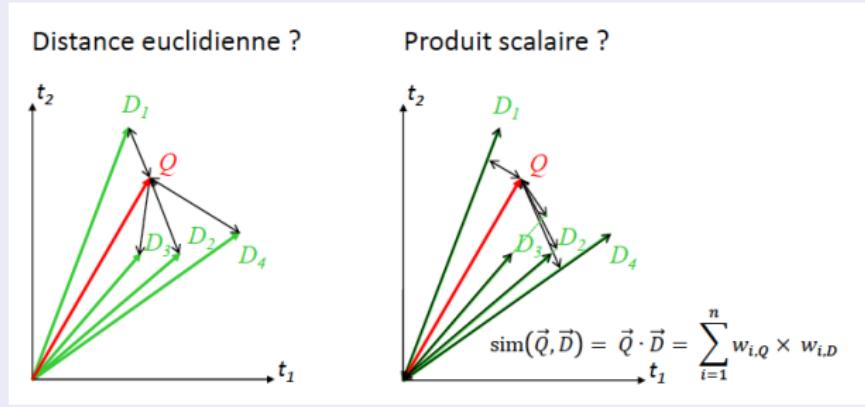


FIGURE: D'après X. Tannier

- Degré de similarité : corrélation entre \vec{q} et \vec{d} .
- Une bonne idée ?

Modèle Vectoriel

Mesure de similarité

Solution : travailler avec l'angle des vecteurs (produit scalaire avec normalisation de la longueur des vecteurs)

$$sim(q, d_j) = \cos(\Theta) = \frac{\vec{d}_j \bullet \vec{q}}{|\vec{d}_j| \times |\vec{q}|}$$

Tant que $w_{ij} > 0$ et $w_{iq} > 0$, $0 \leq sim(q, d_j) \leq 1$

$$sim(q, d_j) = \cos(\Theta) = \frac{\sum_{i=1}^n w_{i,j} w_{i,q}}{\sqrt{(\sum_{i=1}^n w_{i,j}^2)} \sqrt{(\sum_{i=1}^n w_{i,q}^2)}}$$

- On normalise par la longueur du document
 - Document long : termes plus fréquents, plus de termes
- Les documents sont classés par valeur décroissante de cosinus.

Modèle Vectoriel

Autres mesures de similarité

- Dice

$$RSV(q, d_j) = \frac{2 \sum_{i=1}^n w_{i,j} w_{i,q}}{\sum_{i=1}^n w_{i,j} + \sum_{i=1}^n w_{i,q}}$$

- Jaccard

$$RSV(q, d_j) = \frac{\sum_{i=1}^n w_{i,j} w_{i,q}}{\sum_{i=1}^n w_{i,j} + \sum_{i=1}^n w_{i,q} - \sum_{i=1}^n w_{i,q} w_{i,j}}$$

- Overlap

$$RSV(q, d_j) = \frac{\sum_{i=1}^n w_{i,j} w_{i,q}}{\min(\sum_{i=1}^n w_{i,j}, \sum_{i=1}^n w_{i,q})}$$

Modèle Vectoriel

Principe de la requête

- La requête est considérée comme un document court
- On retourne la liste des documents classés en fonction de la proximité de leur vecteur avec le vecteur de la requête

$$sim(q, dj) = \cos(\Theta) = \frac{\sum_{i=1}^n w_{i,j} w_{i,q}}{\sqrt{(\sum_{i=1}^n w_{i,j}^2)} \sqrt{(\sum_{i=1}^n w_{i,q}^2)}}$$

Modèle Vectoriel

Algorithme de recherche avec la mesure cosinus

Entrées : La collection \mathcal{C} de N documents et son index inversé associé, un tableau contenant les facteurs de normalisation associés aux documents n_{d_j} et une requête constituée de K termes $q = (t_{1,q}, \dots, t_{K,q})$

Algorithm 4: VECTORIAL-SEARCH(q , \mathcal{C} , Index of \mathcal{C})

```

1   $n_q \leftarrow 0$ 
2  for  $j \in [1, \dots, N]$  do
3       $s[j] \leftarrow 0$ 
4  for  $i \in [1, \dots, K]$  do
5       $w_{t_{i,q},q} \leftarrow p_{tf_{t_{i,q},q}} \times p_{df_{t_{i,q},q}}$ 
6       $n_q \leftarrow n_q + (w_{t_{i,q},q})^2$ 
7       $L \leftarrow$  posting List of  $t_{i,q}$ 
8      for  $d_j \in L$  do
9           $w_{t_{i,q},d_j} \leftarrow n_{d_j} \times p_{tf_{t_{i,q},d_j}} \times p_{df_{t_{i,q},d_j}}$ 
10          $s[j] \leftarrow s[j] + w_{t_{i,q},d_j} \times w_{t_{i,q},d_j}$ 
11 for  $j \in [1, \dots, N]$  do
12     if  $s[j] \neq 0$  then
13          $s[j] \leftarrow \frac{s[j]}{\sqrt{n_{d_j}} \times \sqrt{n_{d_j}}}$ 
14 return The documents with the higher level score  $s[.]$ 

```

Modèle Vectoriel : Bilan

- La requête est représentée comme un vecteur (quelle pondération ?)
- Un document est représenté comme un vecteur pondéré.
- On calcule la similarité entre chaque vecteur document et le vecteur requête.
- On ordonne les résultats dans l'ordre décroissant des scores obtenus.
- On fournit les k premiers résultats à l'utilisateur.

Modèle Vectoriel

Avantages

- Recherche avec classement
- Pondération des termes selon leur importance et donc meilleures performances
- Mise en correspondance partielle
- La fonction d'appariement permet de trier les documents.

Inconvénients

- Hypothèse forte : les termes sont indépendants
- Pondération plus intuitive que formelle
- Le langage de requête est plus simple mais moins expressif.

Le modèle le plus populaire en RI

Plan

- 1 Contexte
- 2 Introduction
- 3 Historique des SRI
- 4 Indexation
 - Des documents aux termes du dictionnaire
 - Lois de base en RI
 - Représentation d'un document et de la collection - Index inversé
- 5 Modèles classiques de recherche d'Information
 - Modèle booléen
 - Traitement des requêtes booléennes
 - Optimisation de requêtes
 - Modèle vectoriel
- 6 Bilan

Quelques outils

- Lucene : <https://lucene.apache.org/core/>, Moteur d'indexation et de recherche en java. Supporté par la fondation Apache.
- Lemur - Indri : <http://www.lemurproject.org/>, C++, api Java, PHP. Université du Massachusetts et Carnegie Mellon.
- Wumpus : <http://www.wumpus-search.org/>, C++. Université de Waterloo.
- Terrier : <http://terrier.org/>, Java, Université de Glasgow
- Xapian : <http://xapian.org/>, C++,
- ...

Bilan de ce cours

- Plusieurs concepts fondateurs de la RI ont été introduits.
- Indexation et index inversé.
- Modèle sac de mots et pondération tf-idf.
- Deux modèles de RI :
 - ▶ Modèle booléen.
 - ▶ Modèle vectoriel.

Suite au prochain épisode

Application pratique ! A vous de jouer !