Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

Centre Génie Industriel et Informatique (G2I)

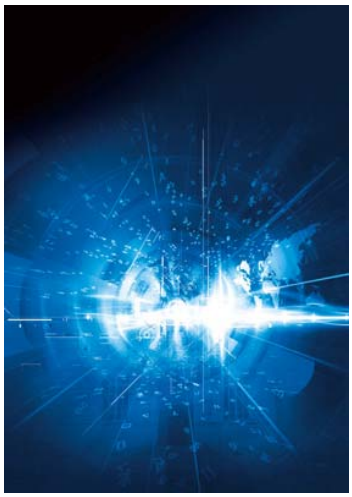
RECHERCHE D'INFORMATION DANS LES DOCUMENTS XML

X . TANNIER

Juin 2006

RAPPORT DE RECHERCHE

2006-400-007



MINISTÈRE DE L'ÉCONOMIE
DES FINANCES ET DE L'INDUSTRIE

Les rapports de recherche
du Centre G2I de l'ENSM-SE
sont disponibles en format PDF
sur le site Web de l'Ecole

G2I research reports
are available in PDF format
on the site Web of ENSM-SE

www.emse.fr

Centre G2I
Génie Industriel et Informatique

Division for
Industrial Engineering and Computer Sciences
(G2I)

Par courier :

By mail:

Ecole Nationale Supérieure des Mines de Saint-Etienne
Centre G2I
158, Cours Fauriel
42023 SAINT-ETIENNE CEDEX 2
France

Table des matières

Index

Note

Ce document est la version longue d'un chapitre d'état de l'art d'une thèse concernant l'extraction et la recherche d'information en langage naturel dans les documents structurés. Nous avons pensé qu'il pouvait être utile en lui-même aux personnes désireuses de se familiariser avec les problématiques de la recherche d'information dans les documents semi-structurés de type XML.

Cet état de l'art n'est donc en aucun cas une description exhaustive des applications des techniques de recherche d'information, mais une introduction à certaines problématiques, choisies au départ pour correspondre avec le sujet général de la thèse présentée.

Chapitre 1

Les documents semi-structurés

1.1 Le langage XML

XML

XML (eXtensible Markup Language [164]) est un standard mis en place par le *World Wide Web Consortium* (W3C [157]). *"Il définit une syntaxe générique utilisée pour marquer les données avec un balisage simple et lisible par les humains."* [1] [58].

La structure est représentée en XML par des *éléments*, contenant des *attributs*, du texte ou d'autres éléments. Les éléments ne peuvent pas se chevaucher. Le choix du nom des éléments structurants et des attributs, ainsi que l'organisation des éléments entre eux, est laissé à la libre volonté de l'auteur. C'est pourquoi on dit que le langage XML est *générique*.

Un exemple de document XML, représentant des informations sur des films présents dans une cinémathèque, est donné à la figure 1.1. L'organisation particulière d'un document XML (notamment l'imbrication des éléments sans possibilité de chevauchement) permet de représenter celui-ci sous forme arborescente, comme le montre l'exemple de la figure 1.2.

Nous pouvons illustrer par cet exemple les bases de la terminologie XML :

- "cinémathèque", "film", "titre", etc. sont des *noms* ou des *types de balises* ;
- `<film>` et `</film>` sont des *balises* (respectivement balises de début et de fin d'élément) ; *balise*
- Les balises de début et de fin ainsi que leur contenu (texte et éléments insérés entre les balises) constituent un *élément*, aussi appelé *nœud* ou *sous-arbre* ; *élément*
- `id="1"` est un attribut (id) ayant la valeur "1". *attribut*
- Les parties purement textuelles (*"Woody Allen"*, *"1993"*, etc.) sont des *éléments textuels*.
- L'élément `acteurs` est le *parent* des éléments `acteur`, qui sont donc ses *enfants*. On dit également qu'un parent *contient* un enfant. Par extension, l'élément `film` est l'*ancêtre* des éléments `acteur` (et `titre`, etc.) qui sont ses *descendants*. *hiérarchie*
- L'élément `cinémathèque` est l'élément *racine*, il est l'ancêtre de tous les autres éléments. Par la même analogie avec les arbres, on dit que les éléments situés au plus bas de l'arborescence (`titre`, `acteur`, etc.) sont des *feuilles*. *racine* *feuilles*

Par ailleurs, les liens hiérarchiques (parenté, attributs nécessaires, etc.) sont décrits par une *DTD* (Document Type Definition) ou par un *Schema* [162]. La DTD ou le Schema définissent un "sous-langage" restreignant, imposant avec plus ou moins de contraintes d'utiliser une organisation prédéfinie. Les Schemas permettent de décrire beaucoup plus de caractéristiques que les DTD (notamment concernant les types

[1] "It defines a generic syntax used to mark up data with simple, human-readable tags." [58, p.3]

```
<cinémathèque>
    <film id="1">
        <titre>Meurtre Mystérieux à Manhattan</titre>
        <réalisateur>Woody Allen</réalisateur>
        <année>1993</année>
        <durée unité="min">108</durée>
        <affiche>./affiches/allen_mmm.jpg</affiche>
        <site>http ://www.woodyallen.com/</site>
        <acteurs>
            <acteur>Woody Allen</acteur>
            <acteur>Diane Keaton</acteur>
            <acteur>Anjelica Huston</acteur>
        </acteurs>
    </film>
    <film id="2">
        ...
    </film>
    ...
</cinémathèque>
```

FIG. 1.1 – Exemple de document XML représenté sous forme textuelle.

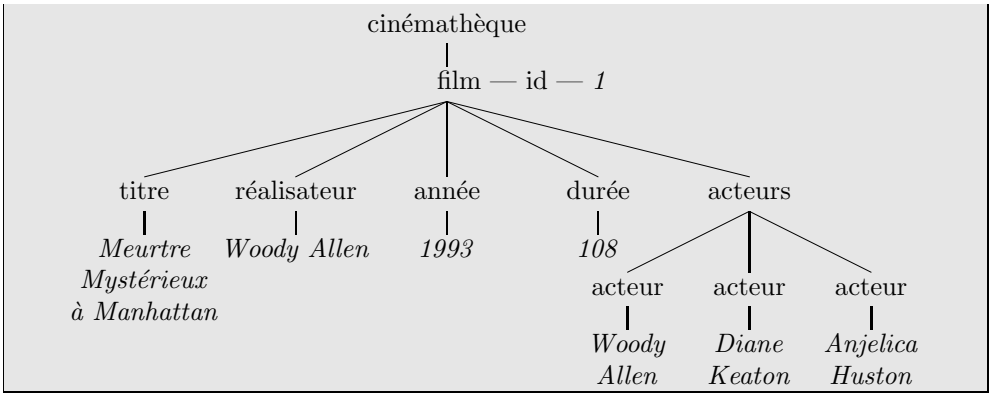


FIG. 1.2 – Exemple de document XML représenté sous forme arborescente.

de données utilisées dans le contenu). La DTD est abordée de façon plus précise en annexe 2.5.4.5.

Ainsi, les formats DocBook [168], la TEI [65] ou XHTML [163], mais aussi des langages de traitement du XML comme XSLT [159] ou XQueryX [165] obéissent à des DTD et/ou des Schemas spécifiques, ce sont des sous-langages de XML.

On voit dans les exemples que si la syntaxe XML est relativement intuitive, en revanche la représentation textuelle d'un document peut vite devenir très verbeuse, et donc assez difficile à lire pour un humain. Des langages de transformation permettant d'obtenir une apparence plus agréable sont donc utilisés (XSLT, CSS).

Il est également possible de naviguer et de sélectionner des informations à travers les documents XML grâce aux langages XPath [158], XSLT [159] ou XQuery [166].

Plus de détails sur XML peuvent être trouvés à l'annexe 2.5.4.5.

1.2 Du document plat au document structuré

Structure *: Manière dont les parties d'un ensemble concret ou abstrait sont arrangées entre elles.*
Dictionnaire Larousse.

marquage, balisage

Coombs et al. [25] distinguent six sortes de marquage, ou balisage :

- La *ponctuation* : virgules, points de toutes sortes, tirets divers. Cette forme de marquage est familière pour tous, au moins dans le monde occidental, relativement stable, mais certaines de ses caractéristiques sont variables selon les langues (espaces autour des signes notamment), les usages ou les habitudes.
- La *présentation* : les auteurs utilisent différents moyens de rendre leurs documents plus clairs. Les passages à la ligne, espacements divers, coupures de pages, énumérations, notes, numérotation, ou même les étiquettes explicites du type "Chapitre 1." entrent dans cette catégorie.
- Le *marquage procédural* : il s'agit d'instructions remplaçant le marquage de présentation, dans le cas où le document est destiné à être formaté par un système. Exemple : `'.sk 3'` pour sauter trois lignes (*"skip 3 lines"*), ou `'.in -10'` pour une indentation négative de dix colonnes.
- Le *marquage descriptif* permet à l'auteur d'identifier le rôle (la sémantique) d'une partie de texte (paragraphe, citation, note). Dans la plupart des documents lus par un être humain, ce marquage est remplacé par des conventions de présentation.
- Le *marquage référentiel* représente des entités externes au document. Une référence à une telle entité sera remplacée par le contenu de celle-ci pendant le formatage du texte.
- Le *méta-marquage* facilite l'interprétation des autres types de marquage, notamment par une description des procédures qu'ils nécessitent.

document plat

La ponctuation, ainsi que la présentation dans une moindre mesure, sont des marquages effectués par l'auteur de tout document écrit, numérique ou non. Le lecteur humain en a besoin pour apporter du sens au texte qu'il lit. Dans la suite de cette thèse, on appellera *document plat* tout texte ne comportant que ces deux types de marquage. On dira d'un tel document qu'il est *non structuré*, même si la ponctuation ou les passages à la ligne sont des éléments structurants (séparation en phrases, en paragraphes, etc.).

On considèrera qu'il existe véritablement une structure lorsque celle-ci est indiquée par des indices conventionnels formels et dénués d'ambiguïté[1].

document structuré

A l'opposé des documents plats, les documents dits *structurés* possèdent une structure régulière prépondérante à base de marquage descriptif. Par exemple, la structure d'une base de données relationnelle est représentée par des tables comportant plusieurs colonnes et plusieurs lignes, ainsi que par des relations entre les champs des différentes tables. L'ordre des éléments n'a généralement pas d'importance. On ne parle alors plus de *texte* mais de *données* ; ces données n'ont habituellement aucune signification intrinsèque, c'est-à-dire qu'il est impossible de les considérer sans examiner la structure dans laquelle elles sont inscrites.

document semi-structuré

Le document *semi-structuré* est un pont entre les données structurées et non structurées [150]. Le langage XML, qui permet de produire des documents à la fois structurés et semi-structurés, est devenu un mode de représentation standard dans le domaine des

[1] On peut remarquer notamment que le point peut signifier une fin de phrase, mais également apparaître dans un sigle ou le nom abrégé d'une personne. La décomposition automatique d'un texte en phrases n'est, de ce fait, pas un problème trivial. Par cette ambiguïté, le point n'est pas considéré comme structurant au sens où nous l'entendons ici.

documents électroniques. D'une part, par un marquage inséré dans le texte, il apporte des éléments sémantiques supplémentaires concernant la structure, le fond ou la forme du document. D'autre part, par le caractère flexible, irrégulier ou incomplet de sa structure [1], ainsi que par son format (textuel), il permet de stocker et d'échanger des données beaucoup plus aisément que les bases de données.

Ainsi, tous les exemples donnés dans ce chapitre témoignent de cette flexibilité. Il suffit par exemple d'écrire une ligne de texte pour ajouter un acteur à un film (figure 1.1) ou un mot-clé à une référence bibliographique (figure 1.4). Si la DTD le permet, il est possible de supprimer la ligne concernant l'éditeur dans la figure 1.3 si cette information est inconnue.

La notion de documents semi-structurés ne recouvre pas une seule réalité. *"Le terme "document" en XML constitue [...] un terme technique, qui ne correspond pas nécessairement à la notion classique d'un document "narratif", c'est-à-dire à un ensemble de données textuelles organisées et mises en forme à l'attention d'un lecteur. Il s'applique également à toute structure de données à vocation d'échange interapplications."* [18]

Comme l'indiquent notamment *Fuhr et Großjohann* [39], il est possible de distinguer deux approches (ou vues) différentes du XML, correspondant à deux besoins différents, et à deux manières d'aborder la recherche d'information :

orienté document

- *l'approche orientée document* considère le document sous sa forme traditionnelle, c'est-à-dire un texte dont la finalité principale reste la lecture par un être humain. Dans cette vue le marquage (ou balisage) sert essentiellement à fournir des informations concernant la structure (chapitres, sections...) et/ou la forme (caractères gras, italiques...) du texte. Des exemples de cette approche sont les manuels, les livres (voir figure 1.3), les articles ou les pages Web statiques.

orienté données

- *l'approche orientée données*, plus proche des bases de données, est utilisée pour représenter et échanger des ensembles structurés de données, comme des horaires de vols, des catalogues, des bibliographies, etc. Cette vue est illustrée par un exemple à la figure 1.4.

En pratique il n'existe pas de frontière nette entre ces deux approches. Ainsi, les premiers éléments du document de la figure 1.3, qui informent sur le titre, l'auteur, l'éditeur (et pourquoi pas d'autres métadonnées*), peuvent être considérés comme étant orientés données, puisque le contenu sans la structure devient beaucoup moins informatif.

1.3 La sémantique du balisage

1.3.1 Généralités

L'ajout d'une structure de type XML aux documents permet de prendre certaines libertés avec le texte lui-même. Notamment, l'ordre *physique* des éléments du fichier XML peut différer de l'ordre réel, *logique* [147], du texte lu par l'humain après mise en forme.

C'est la sémantique particulière de certaines balises, attribuée par l'être humain qui a construit (ou structuré) le document, qui autorise ces manipulations tout en conservant le sens du texte initial.

A l'opposé, pour les processeurs automatiques de documents XML, les balises sont toutes équivalentes, et surtout toutes totalement vides de sens. La perte d'information entre la création d'un document XML et son analyse est donc manifeste. Pour cette raison, des initiatives telles que l'élaboration des Schemas XML [162] et le Web Sémantique [160], dirigées par le World Wide Web Consortium (W3C [157]), visent à

```
<livre type="roman">
    <titre>Le tour du monde en 80 jours</titre>
    <auteur>Jules Verne</auteur>
    <éditeur>Hatzel</éditeur>
    <chapitre n="I">
        <titre_chapitre>
            Dans lequel Phileas Fogg et Passepartout s'acceptent réciproquement,
            l'un comme maître, l'autre comme domestique
        </titre_chapitre>
        En l'année 1872, la maison portant le numéro 7 de Saville-row, Burlington
        Gardens – maison dans laquelle Sheridan mourut en 1814 – était habitée
        par Phileas Fogg, esq., l'un des membres les plus singuliers et les plus
        remarqués du Reform-Club de Londres, bien qu'il semblât prendre à tâche
        de ne rien faire qui pût attirer l'attention.
        ...
    </chapitre>
</livre>
```

FIG. 1.3 – Représentation en XML d'un roman. Ici le texte reste l'information essentielle du document, la structure est secondaire.

représenter formellement la sémantique structurelle des documents.

D'autre part, d'autres travaux portent sur une simple catégorisation des balises pour apporter un surplus de sémantique aux documents [143, 115]. En 1981, bien avant l'invention du langage XML, *Goldfarb* [50] faisait déjà la distinction entre les approches "procédurale" (instructions de formatage) et "descriptive" (composants logiques, comme les paragraphes, les chapitres ou les titres). Cette distinction est devenue consensuelle avec SGML [51] puis XML.

En HTML, qui concerne essentiellement la forme du document, la dichotomie entre balises "logiques" et balises "physiques" sépare le rôle sémantique joué par un élément d'une stricte description de son apparence.

Par ailleurs, la TEI (Text Encoding Initiative [65]) a mis en place une intéressante répartition entre différents niveaux d'inclusion : "chunk", "phrase-level" et "inter-level". La définition de ces niveaux est purement syntaxique et aucune distinction sémantique n'est faite.

D'autres réflexions ont été menées plus récemment [114, 105, 115, 141], introduisant de nombreux nouveaux concepts, parmi lesquels nous pouvons citer une discrimination entre deux axes (comprenant d'une part les domaines logique et renditionnel, et d'autre part impératif ou indicatif [114]) ainsi que les marquages proleptiques et métaleptiques [105].

Nous allons nous attarder sur une division des balises particulièrement appropriée au domaine de la recherche d'information, division que nous utiliserons par la suite. Elle a été proposée par *Lini et al.* [85], dans le but d'identifier différentes catégories qu'il serait important de distinguer dans le cadre de la recherche dans des documents XML. L'idée de départ était de permettre des traitements différents pendant une recherche de motif (de séquence de caractères) [85, 24].

```
@InProceedings{Boudon99,
    title = {Le 18 Brumaire dans l'histoire},
    author = {Jacques-Olivier Boudon and Yves Bruley},
    booktitle = {Actes du colloque "Brumaire. La prise de pouvoir de
Bonaparte"},
    year = {1999},
    address = {Paris, France},
    pages = {161-173},
    keywords = {Napoléon Bonaparte, Coup d'Etat du 18 Brumaire}
    }
```

a. Représentation en BibTeX [82]

```
<bibitem type="InProceedings" key="Boudon99">
    <title>Le 18 Brumaire dans l'histoire</title>
    <author>Jacques-Olivier Boudon and Yves Bruley</author>
    <booktitle>Actes du colloque "Brumaire. La prise de pouvoir de
    Bonaparte"</booktitle>
    <year>1999</year>
    <address>Paris, France</address>
    <pages>161-173</pages>
    <keywords>
        <item>Napoléon Bonaparte</item>
        <item>Coup d'Etat du 18 Brumaire</item>
    </keywords>
</bibitem>
```

b. Représentation en XML

FIG. 1.4 – Représentation d'un élément bibliographique sous un format BibTeX (a.) ou XML (b.). On parle dans ce cas-là de document semi-structuré, car la structure n'est pas régulière ; des champs peuvent être omis s'ils sont inconnus ou non pertinents (adresse, pages...), d'autres ont une certaine structure interne (par exemple, dans le champ `author`, les noms obéissent à une certaine grammaire et sont séparés par le mot réservé `'and'`). BibTeX permet aussi d'effectuer des références croisées. Les mots-clés (`keywords`) de la version XML contiennent un nombre de champs `item` indéterminé. Toutes ces caractéristiques rendraient inutilement complexe la description de ces informations par des bases de données [1]. Cet élément bibliographique est néanmoins *orienté données*, puisque la structure a autant d'importance que le contenu textuel.

1.3.2 Balises dures, balises de saut, balises transparentes

Les trois différentes classes proposées par *Lini et al.* [85] sont les suivantes :

hard tags

– les *balises "dures"* (*"hard" tags*) sont les plus fréquentes. Elles interrompent la "linéarité" d'un texte et contribuent généralement à la structuraton du document. Des exemples de balises dures sont les titres, les chapitres, les paragraphes.

(1.1)
```
...
<titre>Text Retrieval System</titre>
<auteur>Michele Paoli</auteur>
...
```

– Les *balises "transparentes"* (*"soft" tags*) identifient des parties significatives du texte, comme les textes cités, les effets de forme ou les corrections, mais sont "transparentes" lorsqu'on lit le texte. *soft tags*

(1.2) Les ministres de l'<gras>Union Européenne</gras> se sont réunis ...

– Les *balises de "saut"* (*"jump" tags*) sont utilisées pour représenter des éléments particuliers comme les notes de marges, les références bibliographiques ou des définitions. Elles sont détachées du texte les entourant, comme la `'note'` suivante : *jump tags*

(1.3) La mort de Mozart, dans l'indifférence générale<note>Le corps du compositeur sera jeté à la fosse commune, sans même une croix.</note>, l'empêcha de terminer son requiem.

N.B. : Des éléments comme les tableaux ou les listes, qui ne sont pas abordés par les auteurs, semblent particuliers. Les tableaux peuvent être considérés en première approximation comme des éléments "durs", mais leurs spécificités demandent une attention particulière [33]. Quant aux listes, qui possèdent également des caractéristiques propres [2], nous verrons qu'il est possible de les traiter de plusieurs façons différentes.

Par ailleurs, les formules mathématiques, chimiques ou autres peuvent être utilisées dans le texte (éléments transparents) ou dans un contexte à part (éléments durs).

Chapitre 2

Recherche d'information et structure

La recherche d'information automatisée est une discipline qui met en jeu le stockage et la représentation de l'information d'une part, l'analyse et la satisfaction d'un besoin d'information d'autre part.

Plusieurs disciplines évoluent dans cette problématique, elles se chevauchent souvent, mais restent distinctes à de nombreux points de vue. C'est pourquoi la première section de ce chapitre est consacrée à décrire ce qu'est la recherche d'information (RI), et ce qu'elle n'est pas. Nous analyserons également les raisons pour lesquelles une discipline relativement récente, la RI dans les documents semi-structurés de type XML, introduit une problématique spécifique (section 2.2). Les concepts de base étant définis, la suite du chapitre sera consacrée à décrire le processus de recherche dans les grandes lignes. Nous passerons en revue les étapes d'indexation (section 2.3) et d'interrogation (section 2.4), en insistant pour chacune d'elles sur les caractéristiques propres aux documents semi-structurés. Enfin nous nous attarderons tout particulièrement sur la question de l'évaluation des systèmes de RI (section 2.5) en présentant notamment la campagne INEX qui s'est spécialisée dans la recherche XML (section 2.5.4).

2.1 Introduction

> Information (INFORM) *: Elément de connaissance susceptible d'être codé pour être conservé, traité ou communiqué.*
> Dictionnaire Larousse.

La recherche d'information (RI), dans le cas général, comporte deux acteurs principaux :

- Un être humain, qui a un besoin d'*information* à un moment donné[1] ;
- Un stock de *données*, fixé au préalable.

données vs. *info*

"Les données sont reçues, stockées et retrouvées par un endosystème. Les données sont impersonnelles ; elles sont disponibles pour tout utilisateur du système. L'information, en revanche, est un ensemble de données qui correspond à un besoin particulier. Le concept d'information a des composantes personnelles et temporelles absentes du concept de donnée." [2] [78]

[1] Ou qui est dans un *"état anormal de connaissance"* ("Anomalous state of knowledge") [11]

[2] "*Data* are received, stored, and retrieved by an information endosystem. The data are impersonal ; they are equally available to any users of the system. *Information*, in contrast, is a set of data that have been matched to a particular information need. That is, the concept of information has both personal

Les besoins d'information des utilisateurs sont divers, et cette diversité influe sur le mode de recherche de cette information. Il est possible de distinguer quatre types génériques de besoins d'information, et donc de stratégies de recherche [95, chap. 13] :

- *La recherche d'un élément connu* : l'utilisateur sait exactement quels éléments il recherche. Il sait reconnaître les éléments désirés s'il les voit. Par exemple, il recherche une citation bibliographique précise et possède assez d'information pour l'identifier sans ambiguïté.
- *La recherche d'une information spécifique* : l'utilisateur recherche une information spécifique mais ignore sous quelle forme elle se présente. Cette information peut d'ailleurs être présente à plusieurs reprises. Exemple : A quelle date le président Kennedy a-t-il été assassiné ? La réponse sera trouvée ou non, mais ne peut être partielle.
- *La recherche d'une information générale* : l'utilisateur recherche une information sur un sujet en général. Il existe de nombreuses façons de décrire le sujet. Il est possible que l'information pertinente ne soit pas reconnue, et cette information peut ne satisfaire l'utilisateur que de façon partielle [10].
- *L'exploration* : le but n'est pas de répondre à une question en particulier, mais de parcourir l'ensemble des données pour découvrir quels types d'informations concernant un sujet ou un domaine sont présents.

Ces différentes stratégies de recherche, en particulier les trois premières, sont le sujet d'étude de domaines scientifiques distincts : la recherche d'un élément connu a nécessité la mise en place de langages de requête structurés tels que SQL [36] pour les bases de données ou XQuery [166] pour XML. La recherche d'une information spécifique est plutôt du domaine des systèmes de question-réponse [156]. Enfin le troisième type de besoin est l'apanage de la recherche d'information.

RI

La RI est donc le processus consistant à chercher une réponse appropriée au besoin de l'être humain dans la masse de données disponible. Pour satisfaire ce besoin, il ne suffit pas d'apporter à l'utilisateur un document traitant du sujet demandé ; il faut également que le type du document (en termes de longueur, de complexité, de niveau de langage, de portées géographique et temporelle [54]) soit approprié.

Nous allons bien sûr nous intéresser exclusivement à la recherche d'information assistée par ordinateur [127, 8], ce qui ajoute un acteur : le système de recherche d'information (SRI). Celui-ci fait l'interface entre les deux acteurs précédemment cités.

requête

L'utilisateur exprime son besoin à la machine au moyen d'une requête, et le système a pour mission de retrouver dans la masse d'information disponible les documents répondant à la requête. Etant donnée la quantité importante de données potentiellement retrouvée, les systèmes modernes effectuent un classement des documents de manière à placer en tête de liste ceux qui sont jugés les plus pertinents. La figure 2.1 illustre ce

modèle de RI

processus de façon très générale. Une phase préalable d'analyse des documents, indépendante du besoin, est nécessaire. Elle correspond en pratique au processus d'*indexation*, elle utilise un *modèle de documents* (cf. section 2.3). La formalisation de la requête, ainsi que la recherche et le classement des documents (à l'aide d'une *fonction d'appariement*) représentent la base de la phase d'*interrogation* (cf. section 2.4). Le résultat est dans la plupart des cas un ensemble ordonné de références à des documents de la collection (celui jugé le plus pertinent étant bien sûr placé en première position). La description théorique de toutes ces étapes est appelée *modèle de recherche d'information*.

De nombreuses extensions, comme le retour (ou rétroaction) de pertinence (*relevance feedback*), la reformulation de la requête, l'évaluation, peuvent venir enrichir ce schéma universel [127, 9, 56, 106, 131].

and time-dependant components that are not present in the concept of data." [78, page 8]

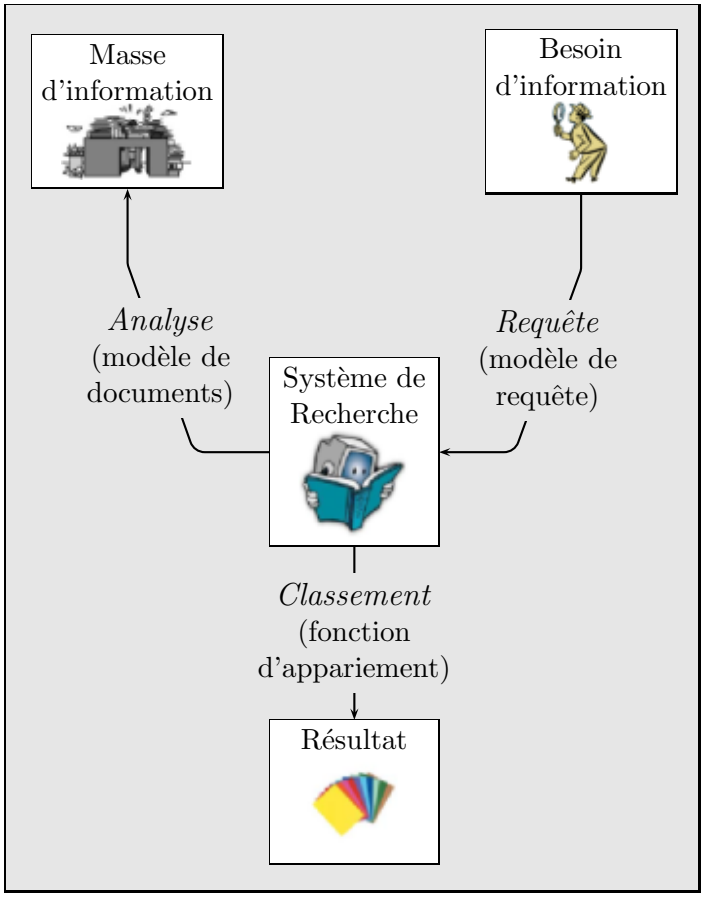


FIG. 2.1 – Principe général de la recherche d'information.

Il est important de noter d'ors et déjà que le besoin d'information de l'utilisateur est parfois vague et toujours subjectif. Deux personnes décriront un même besoin de deux façons différentes, et même une seule personne le décrira différemment selon son humeur. De plus une formulation peut être comprise de plusieurs façons par des personnes différentes. La prise en compte de ces aspects psychologiques conduit aux constats suivants :

- La perte d'information entre la réalité du besoin d'information et son expression peut être importante.
- La *pertinence* d'un document pour une requête est une notion variable et très complexe à définir.
- Il ne peut pas exister de système de recherche d'information parfait.
- L'évaluation d'un système dépasse les aspects habituels de performance informatique, nécessite la mise en place de métriques particulières et l'intervention de juges humains, eux-mêmes subjectifs et soumis à des variations entre les individus d'une part, et pour un même individu d'autre part.

Nous reviendrons sur ces différents points tout au long de ce chapitre.

Comme nous l'avons dit, chaque type de besoins d'information se concrétise par une discipline de recherche spécifique. Ainsi, ce que nous appelons ici "recherche d'information" se distingue d'autres domaines proches mais mettant en œuvre des techniques sensiblement différentes. Notamment :

- La recherche de données qui consiste, dans le cadre de la recherche dans un texte,

à déterminer quels documents contiennent les mots-clés demandés par l'utilisateur [8]. Dans le domaine des bases de données, elle permet de retrouver des données dans un espace structuré. Dans tous les cas l'évaluation de la réponse est binaire (la donnée correspond ou ne correspond pas à ce qui est cherché).

extraction d'information

– L'extraction d'information [30] qui isole des fragments d'information, les classe et les organise dans un format fixe. Elle peut servir au remplissage automatique de bases de données ou à un processus d'indexation, préalable à la recherche d'information.

question/ réponse

– Les systèmes question/réponse (*Question Answering*) [46, 156] qui donnent une réponse aussi concise que possible à une question précise posée en langage naturel*. Par exemple, la réponse à la question *"Qui était président de la France en 1985 ?"* doit être *"François Mitterrand"*, et non des documents traitant ce sujet.

résumé

– Le résumé automatique [97] qui a pour but de prendre un ou plusieurs documents pré-sélectionnés en entrée et d'en composer un résumé.

2.2 Les spécificités de la recherche d'information semi-structurée

La recherche de l'information dans des corpus semi-structurés (écrits en XML dans notre cas) avec une approche orientée document* comporte des caractéristiques spécifiques qui la distinguent fortement de la recherche d'information classique. Par certains aspects, notamment la nécessité d'utiliser un langage de requêtes structuré, elle se rapproche des interrogations de bases de données. Pourtant les enjeux restent bel et bien ceux de la recherche d'information.

2.2.1 L'unité d'information pertinente

unité d'information

Dans la discipline de la recherche d'information, se pose le problème de l'*unité d'information* adéquate. En réponse à une requête, la majorité des systèmes retourne un ensemble de *documents*. Outre qu'il est parfois difficile d'établir ce qu'est un document numérique [110], le choix de cette unité n'est pas une garantie de la satisfaction de l'utilisateur. Il peut en effet sembler plus pertinent, dans certains cas, de ne retourner qu'une portion du document, voire un ensemble de textes issus de documents différents, pour éviter que l'information utile ne soit trop dispersée [100]. De plus, la division des documents en plus petites unités de sens peut constituer une aide au processus de recherche [61]. Ces considérations ne datent pas de l'apparition des langages semi-structurés, et le sujet préoccupe depuis longtemps les tenants de la *recherche de passages* (*passage retrieval* [124]).

doxel

Dans le cadre de la recherche d'information dans des documents XML, une nouvelle unité d'information potentielle se présente : l'élément XML, aussi appelé *doxel* [107] (pour *"document element"* en Anglais, et par analogie avec le pixel, l'unité graphique élémentaire), ou *nœud* (sous-arbre) dans la représentation arborescente du document. Il est également possible de s'intéresser à des ensembles de doxels, consécutifs ou pas dans les documents initiaux, ceux-ci composant eux-mêmes un document (semi-)structuré [106]. Les autres unités, comme le document entier, la phrase, le paragraphe ou des unités sémantiques particulières, sont toujours à prendre en considération.

Par ailleurs, la pertinence du choix d'une certaine unité d'information en réponse à une requête peut se mesurer selon deux dimensions [20, 80]. La première correspond à la notion de présence de l'information demandée dans l'information apportée ; la seconde détermine si l'unité d'information retournée a une taille appropriée par rapport à la

quantité et à la disposition des informations pertinentes qu'elle contient. Ces deux dimensions sont reprises dans les mesures de *degré de pertinence* et de *couverture*, puis d'*exhaustivité* et de *spécificité* mises en place pour l'évaluation de la recherche d'information dans les documents XML (cf. section 2.5).

auto-explicativité

On considère généralement que la partie de document renvoyée par un système se doit d'être *auto-explicative*, c'est-à-dire de ne pas dépendre du reste du document pour être comprise. Pourtant cette règle dictée par une vision de laboratoire de la recherche d'information [75] nous semble très discutable. Tout d'abord parce qu'aucun document complet n'est indépendant du reste de la connaissance (voir les quelque 250 références externes que comporte ce mémoire). Ensuite parce qu'un document est généralement écrit pour être lu en entier ; il est conçu comme un tout, l'humain et le langage étant ainsi faits qu'il semble difficilement concevable qu'un paragraphe bien écrit ne fasse en aucune manière référence au reste (introduction préalable du contexte, recours à des références pronominales, à des paraphrases, etc.). Une application stricte du principe d'auto-explicativité conduirait à une perte bien trop importante de *spécificité*. Par ailleurs l'exposition des résultats qui semble satisfaire le plus l'utilisateur est celle consistant à mettre en évidence l'élément retourné tout en le présentant dans son contexte (le document entier), avec notamment un accès à la table des matières de ce document [45, 83].

2.2.2 Recherche sur contenu et structure

L'utilisateur cherchant une information dans un corpus qu'il sait structuré peut formuler ses requêtes de trois façons différentes :

contenu seul

– Il peut choisir d'ignorer la structure et de ne formuler sa requête qu'en fonction du *contenu*, parce qu'il n'a pas d'idée précise de l'endroit où se trouve la réponse à sa question, ou parce que son besoin d'information est trop vague, ou encore parce qu'il ne connaît pas précisément la structure des documents.

contenu/ structure

– Il peut "aider" le moteur de recherche en précisant des contraintes structurelles dans sa requête.
– Il peut également ne se préoccuper que de la structure. Cet aspect correspond à une interrogation de type "base de données" (listings par exemple) dans laquelle la recherche d'information n'a pas sa place, et nous ne l'aborderons pas.

Le format habituel d'une requête, dans le cadre de la recherche d'information dans des documents plats*, est une simple liste de mots-clés, ordonnée ou non, avec éventuellement des opérateurs (booléens ou autres). Le besoin d'exprimer des contraintes structurelles impose de mettre en place des langages de requêtes possédant un formalisme plus complexe. Les possibilités des systèmes de recherche documentaire "classique" restent indispensables, mais des fonctionnalités propres aux manipulations de bases de données doivent être ajoutées. Ainsi *Sacks-Davis et al.* [120] distinguent huit classes de requêtes, de la recherche par mots-clés uniquement jusqu'aux accès de type bases de données, en passant par les requêtes dans des sous-parties de documents ou dans différents types de documents. *Bonifati et Ceri* [12] comparent des modèles de RI en utilisant dix groupes de caractéristiques, parmi lesquelles le modèle de données, les possibilités de sélection, de filtrage, de jointures, de restructuration, d'opérations sur les ensembles, de mise à jour des données, mais aussi l'intégration avec les spécificités de XML. *Baeza-Yates et Navarro* [7] ajoutent la possibilité de chevauchement structurel dans les réponses et la notion d'*expressivité*, et *Sauvagnat* [132, 131] celle d'*informativité*.

Un compromis est souvent nécessaire entre les fonctionnalités avancées, la facilité d'utilisation du langage et la complexité informatique des algorithmes mis en place.

2.2.3 Interprétation de la requête

Si l'utilisation d'un langage de requêtes structuré semble rapprocher la recherche d'information semi-structurée de l'interrogation des bases de données, il est important de souligner que les enjeux restent très différents.

Dans le document XML tel que nous le considérons, le contenu textuel est l'information prépondérante. La recherche d'information est un instrument appliqué à des documents dont la fonction première est d'être tout simplement lus par des humains (après une mise en forme appropriée). A l'opposé, une base de données n'existe que pour être interrogée par un utilisateur ou un logiciel (en SQL généralement). Cet utilisateur connaît dans la plupart des cas la structure de la base qu'il interroge et il doit impérativement fournir des indications structurelles dans sa requête. De plus son besoin d'information est précis, il recherche des données répondant à des contraintes de contenu et de structure *strictes*.

L'interprétation de la requête SQL est donc stricte. En conséquence, une requête correcte conduira inévitablement à la réponse attendue, et tout autre requête produira des résultats inutiles. Si la notion de requête "meilleure" qu'une autre existe, c'est en termes de *performances* (rapidité de traitement par tel ou tel système de gestion), mais en aucun cas au niveau de la *qualité* des enregistrements retournés.

En revanche, en recherche d'information (structurée comme traditionnelle), l'utilisateur n'a généralement pas une idée précise du type d'information qui le satisferait le plus (taille, forme et même contenu). Sa requête n'est donc plus le reflet fidèle du résultat souhaité ; elle est plutôt considérée comme une "aide" apportée au système pour lui permettre de répondre efficacement au besoin d'information. C'est pourquoi il n'existe pas de réponse "parfaite" à une requête, et il est toujours possible d'améliorer sa requête pour obtenir des résultats plus proches de son attente. De plus, des requêtes très différentes peuvent conduire à des résultats de pertinence comparable.

Ainsi un résultat qualitativement satisfaisant ne dépend, dans le cadre des bases de données, que de la composition de la requête ; en RI, en revanche, la clé d'un bon résultat est partagée entre la requête et le système qui la traite. Le moteur de recherche a donc la "liberté" d'interpréter la requête de façon beaucoup plus souple. Ce principe s'applique bien sûr au contenu textuel[1] mais aussi à la structure. Une contrainte structurelle peut n'être considérée que comme une indication non stricte. On peut par exemple imaginer qu'un utilisateur qui demande un paragraphe expliquant un concept donné puisse être satisfait par une figure décrivant ce concept. En revanche un long article sera moins pertinent, puisque la requête concernait un élément relativement court.

Ces subtilités spécifiques à la recherche d'information semi-structurée sont très importantes et justifient en grande partie le travail présenté par la suite.

2.2.4 Bilan

La tableau 2.1 récapitule les enjeux de la recherche d'information dans les documents semi-structurés. Il ne s'agit ici que des aspects concernant le principe même de la recherche, et non des techniques à mettre en œuvre pour gérer les problèmes engendrés par la présence d'une structure flexible (indexation, évaluation, etc.). Par ailleurs nous n'abordons que les documents dont le contenu est textuel. Pourtant, nous avons vu (section 1.1) que les documents XML pouvaient intégrer des références à des éléments multimédia, voire ces éléments eux-mêmes, qui peuvent bien entendu être pris en compte

[1] Les systèmes peuvent retourner des documents ne contenant pas tous les termes de la requête, mais aussi utiliser des techniques d'extension de requête, de racinisation, etc. [146].

	Documents plats	*Documents semi-structurés*	*Documents structurés*
Contenu	texte seulement	texte + structure	structure + données
Besoin d'information	**général**		précis
Unité de recherche	document/segment	**élément (doxel)**	(tuple)
Demande	texte seulement	contenu et/ou structure	
Requête	mots-clés	**langages de requêtes structurés**	
Interprétation	**vague**		stricte

TAB. 2.1 – Récapitulatif de quelques caractéristiques des documents plats, semi-structurés et structurés dans le cadre de la recherche d'information.

en recherche d'information [153].

Des observations menées auprès d'utilisateurs "réels" (non experts) en 2005 [83] ont montré que ces fonctionnalités spécifiques, en particulier la variabilité de la taille de l'unité de recherche, apportaient une véritable satisfaction en comparaison des moteurs de recherche classiques connus (de type Google[1] ou Yahoo![2]). La condition de cette satisfaction est une présentation appropriée des résultats, tenant compte du chevauchement éventuel des réponses et du contexte des éléments retournés.

2.3 Indexation

Il n'est pas envisageable d'effectuer une recherche directement dans la collection de documents à chaque nouvelle requête. Notamment de nombreux traitements de cette collection, préalables à la recherche, peuvent être effectués une seule fois (ou au moins à chaque fois que la collection change) : détermination de la pondération des mots, élimination des mots "vides" (c'est-à-dire très fréquents et/ou sans portée sémantique, tels que les mots de liaison [37, 134]), lemmatisation*, étiquetages syntaxiques ou sémantiques, reconnaissance des termes complexes, etc.

indexation

L'indexation consiste à *"identifier l'information contenue dans tout texte et [à] la représenter au moyen d'un ensemble d'entités appelé index pour faciliter la comparaison entre la représentation d'un document et d'une requête."* [118, p. 12]. Cette représentation est formalisée par un *modèle de documents*. Il s'agit principalement, dans le cadre de la recherche d'information traditionnelle (documents plats), de collecter des mots-clés permettant un ciblage sémantique du document. Ce processus peut être effectué de façon manuelle, automatique ou semi-automatique (un expert ajustant un index composé automatiquement). Si le processus manuel permet de répérer de façon plus précise les mots-clés décrivant un document, il est très coûteux en temps, subjectif et pose des problèmes de consistence [148] (dus notamment aux variations sémantiques [146] – et aux présomptions des indexeurs concernant les connaissances et les buts du lecteur [64]). Pour contrer ce problème on peut choisir les mots-clés dans un ensemble de mots prédéfini (langage *contrôlé*), mais le risque est alors de perdre en *spécificité*[3].

modèle de document

De *Salton* [122] à *Savoy* [135], les études sur l'efficacité relative des indexations manuelle et automatique n'ont pas conduit à une préférence claire.

[1] http://www.google.com.

[2] http://search.yahoo.com.

[3] Ainsi, une requête concernant les bergers allemands sera mal traitée si le seul mot autorisé par l'ensemble contraint de termes est *"chien"*.

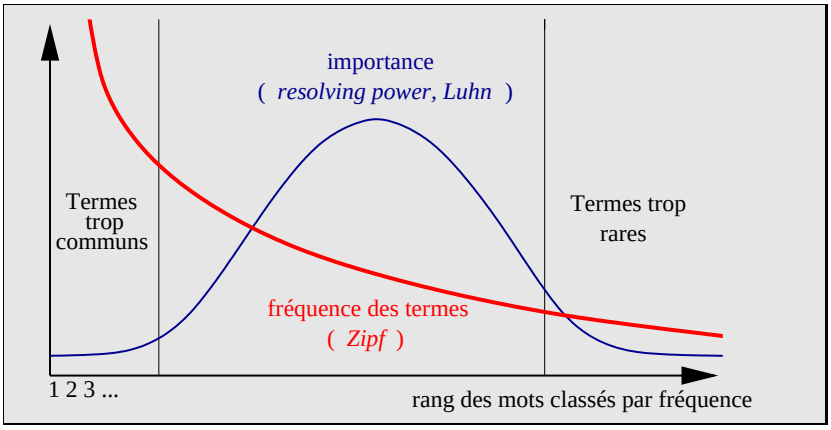


FIG. 2.2 – Fréquence d'apparition et importance des mots dans un texte.

Nous abordons ici rapidement l'indexation automatique à travers la pondération des termes collectés et la construction de l'index[1].

2.3.1 Pondération des termes

La pondération des termes permet d'une part de déterminer l'importance relative des termes de la requête (leur pouvoir discriminant[2]), et d'autre part de quantifier leur importance dans la description sémantique du contenu d'un document[3].

tf - idf

Les techniques courantes de pondération des termes sont basées sur les notions de fréquence des termes dans un document (*tf* pour *term frequency*, mesure représentant l'importance locale d'un terme – fréquence relative) et de fréquence de ces termes dans l'ensemble des documents de la collection étudiée (on utilise l'inverse de cette mesure, simulant l'importance globale d'un terme, soit *idf* pour *inverse document frequency* – fréquence absolue) [125], notions elles-mêmes fondées sur des constatations linguistiques de *Zipf* [174] et *Luhn* [87]. La loi de Zipf stipule notamment que si l'on classe les termes d'un document faisant du sens et rédigé dans une langue naturelle du plus fréquent au plus rare, alors le rang d'un terme dans ce classement est inversement proportionnel à sa fréquence. Luhn a utilisé cette loi pour en déduire une relation entre la fréquence des termes et leur pouvoir discriminant, c'est-à-dire leur importance dans la participation au sens d'un texte (figure 2.2).

La mesure de l'*idf* part du principe que *"le nombre de documents pertinents à une requête est généralement faible [en comparaison du nombre total de documents], et donc les termes apparaissant fréquemment doivent nécessairement apparaître dans beaucoup de documents non pertinents. En revanche, les termes peu fréquents ont une plus grande probabilité d'apparaître dans des documents pertinents [et donc sont plus importants]."* [4] [68]

[1] Pour plus de détails, *Roussey et al.* [119] proposent un tour d'horizon complet de cette problématique.

[2] Par exemple, dans une requête portant sur l'*indexation des documents*, le mot *"indexation"* a plus d'importance discriminative que le mot *"document"*.

[3] On peut imaginer notamment qu'un terme apparaissant à de nombreuses reprises dans un document le décrira de façon plus judicieuse qu'un terme apparaissant rarement.

[4] "The number of documents relevant to a query is generally small, and thus any frequently occurring terms must necessarily occur in many irrelevant documents ; infrequently occurring query terms, conversely, have a greater probability of occurring in relevant documents and should thus be considered

Le produit de ces deux mesures [67], utilisées avec des formules diverses, sert donc à indiquer l'importance d'un terme dans un document. Un terme répété de nombreuses fois dans un petit nombre de documents est probablement très significatif quant au sujet de ces documents.

Une amélioration des poids peut être effectuée en utilisant la technique de retour de pertinence (*relevance feedback* [127, p. 236]), en favorisant les termes présents dans les documents qui s'avèrent pertinents.

Notons que l'utilisation brute de ces métriques est une simplification forte mais courante consistant à considérer que le poids d'un terme dans un document est indépendant du poids des autres termes du document [8, p. 25].

Dans le cadre de la recherche d'information dans les documents XML, les unités d'information sont imbriquées les unes dans les autres. Les mesures aux niveaux local et global vues ci-dessus nécessitent d'être adaptées, d'autant plus que la loi de Zipf n'est plus obligatoirement respectée dans les documents possédant une structure [133]. L'importance d'un terme dans un élément dépend fortement de l'importance de ce terme dans les sous-éléments qui le composent [86], et même des éléments situés autour [74]. Ceci a provoqué de nombreuses propositions conservant la mesure *tf.idf* [21, 136, 79] ou l'abandonnant [4, 23, 29, 74], avec notamment la définition de l'*inverse element frequency (ief)* [170, 53] pour généraliser la notion d'*idf*, en considérant les nœuds* d'un certain type plutôt que les documents.

ief

2.3.2 Indexer le contenu et la structure

Nous ne nous attarderons pas ici sur les diverses méthodes utilisées pour créer un index (fichiers inversés, tableaux ou arbres de suffixes, fichiers de signatures [8, chap. 8]). Dans tous les cas le principe consiste à faire correspondre aux termes de la collection (parfois modifiés d'une façon ou d'une autre) un certain nombre d'informations utiles au processus de recherche. Dans le cas d'une collection de multiples documents à indexer, l'information minimale à faire figurer est bien entendu l'identifiant des documents contenant le terme. Par la suite on peut ajouter les positions de ce terme dans chaque document, ceci permettant de mettre en place des algorithmes tenant compte de la proximité des termes concernés par la requête [173]. On peut également ajouter des informations d'ordre linguistique, comme la catégorie grammaticale du mot, jusqu'à aller vers des types d'indexation bien plus complexes comme l'utilisation de graphes conceptuels ou l'indexation sémantique [101].

En ce qui concerne les documents structurés (et en particulier XML), le schéma d'indexation dépend des capacités voulues pour le langage de requêtes qui fera l'interface entre l'utilisateur humain et le système. On peut bien sûr se contenter d'ignorer le balisage et de considérer l'élément racine comme un document plat [120], ou de considérer les balises comme des termes classiques de l'index [89] (ainsi dans l'exemple 1.3 de la page 7, la balise `<auteur>` est indexée en tant que terme *"auteur"*). Mais il est impossible dans ce cas-là de mener une recherche véritablement structurée.

Dès lors que l'on souhaite effectuer une indexation structurée d'un document XML, la question des liens hiérarchiques entre les unités d'information se pose. A quel point l'information présente dans un élément est-elle répercutée dans ses ancêtres* ou ses descendants* ? Ce problème se présentera à chaque étape de la recherche d'information. Au niveau de l'indexation, si beaucoup considèrent les éléments comme des *unités disjointes* sans connection, d'autres [29, 136, 138] propagent certains termes ou tous les

unités disjointes

as being of greater potential importance when searching a database." [68, p. 307] Les termes trop rares ont moins d'importance car leur probabilité d'apparition est trop faible.

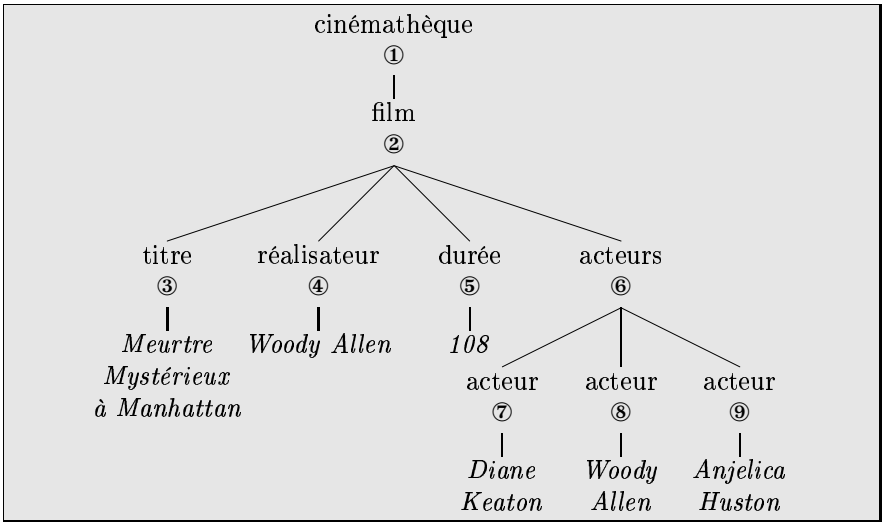


FIG. 2.3 – Exemple de document XML représenté sous forme arborescente, avec numérotation des nœuds.

sous-arbres imbriqués

termes d'un élément donné vers ses ancêtres (technique dite des *sous-arbres imbriqués*). Ainsi on peut considérer que dans l'exemple de la figure 2.3 l'élément `film` contient tous les mots de ses descendants. Nos exemples suivants tiennent compte de cette présence ou non de propagation.

L'indexation de la structure passe par le choix du mode de représentation des éléments structurants. Parmi les possibilités présentées par *Luk et al.* [89], on trouve :

- *L'indexation par les champs*, où chaque terme est associé au nom de la balise dans laquelle il apparaît (voir table 2.2). Cette méthode permet de savoir dans quel contexte l'information textuelle est énoncée. Il est également possible d'indexer selon les *chemins*, c'est-à-dire d'indiquer le chemin d'accès logique plutôt que le nom de la balise [23] (`cinémathèque/film/titre` au lieu de `titre`). Ceci permet de faciliter et d'accélérer l'accès dans certains cas, mais apporte peu d'information supplémentaire (sauf dans le cas où beaucoup de balises de même nom sont présentes à divers niveaux de la structure).
- *L'indexation par les segments* qui divise les documents structurés en régions. Les modèles qui y sont associés [121, 22] sont très efficaces, mais antérieurs à l'apparition du format XML, et ils en supportent mal toutes les possibilités.
- *L'indexation par les arbres*, où un identifiant unique est attribué à chaque nœud (élément) du graphe représentant le document XML [84]. Les termes sont donc associés à cet identifiant, ce qui permet de localiser de façon précise l'endroit où ces termes sont apparus et de retrouver les relations hiérarchiques entre les éléments [137]. L'identifiant unique peut également être, tout simplement, le chemin d'accès (XPath absolu, avec le numéro des éléments) de l'élément [171] (voir table 2.3).

Il est également possible de créer un index différent par type d'élément [93] ou de mixer plusieurs types d'index [139, 59].

Par ailleurs, au niveau de l'indexation purement structurelle (dans laquelle les termes n'ont plus leur place), et dans le cas où les nœuds sont identifiés de façon unique, des informations concernant les relations avec d'autres nœuds peuvent être ajoutées pour faciliter certaines opérations de navigation : une référence au nœud pa-

Terme	*Sans propagation*	*Avec propagation*
Meurtre	titre	cinémathèque, film, titre
Allen	réalisateur, acteur	cinémathèque, réalisateur, film, titre, acteurs, acteur
108	durée	cinémathèque, film, durée

TAB. 2.2 – Exemple d'indexation basée sur les noms des balises (associée au document de la figure 2.3).

Terme	*Sans propagation*	*Avec propagation*
Meurtre	③ (/cinémathèque[1]/film[1] /titre[1])	① (/cinémathèque[1]) ② (/cinémathèque[1]/film[1]) ③ (/cinémathèque[1]/film[1]/titre[1])
Allen	④ (/cinémathèque[1]/film[1] /réalisateur[1])	① (/cinémathèque[1]) ② (/cinémathèque[1]/film[1]) ④ (/cinémathèque[1]/film[1]/réalisateur[1]) ⑦ (/cinémathèque[1]/film[1]/acteurs[1]) ⑧ (/cinémathèque[1]/film[1]/acteurs[1]/acteur[2])
...	...	...

TAB. 2.3 – Exemple d'indexation basée sur les arbres (associée au document de la figure 2.3).

rent [170], au parent et aux enfants [149], au dernier (*rightmost*) descendant [136]. *Grust* [55] propose d'attribuer deux identifiants (numérotation par deux parcours en largeur d'abord et en profondeur d'abord) et de référencer le parent et le nom de la balise. *Hayashi et al.* [60] composent sept index "plats" représentant le titre, l'auteur, le sujet, les mots-clés, etc.

2.4 Interrogation

L'interrogation est, contrairement à l'indexation qui n'est effectuée qu'une seule fois, un processus qui se déclenche à chaque nouvelle requête. Elle consiste dans sa forme générale en trois phases qui sont la formulation de la demande de l'utilisateur, la recherche dans la collection de documents et la présentation des résultats. Comme nous l'avons vu dans l'introduction (voir figure 2.1), le cadre théorique de l'interrogation est essentiellement composé d'un *modèle de requête* et d'une *fonction d'appariement* (ou *fonction de correspondance*) :

modèle de requête

fonction d'appariement

- Le *modèle de requête* décrit comment le besoin de l'utilisateur doit être retranscrit pour pouvoir être traité par le système, et comment ce système va procéder à l'analyse de ce besoin.
- La *fonction d'appariement* détermine la pertinence d'un document par rapport à une requête, et permet éventuellement de classer les documents par ordre de pertinence supposée.

Ces notions sont communes à la RI traditionnelle (*i.e.* concernant des documents plats*), et à la RI dans les documents semi-structurés. Ces principes de bases doivent pourtant être adaptés aux caractéristiques propres des documents XML, notamment le changement de l'unité d'information considérée et la nécessité d'interroger la structure (voir section 2.2). Ceci impose de proposer aux utilisateurs de nouvelles manières de formuler leurs besoins d'information, et de traiter ces requêtes avec des techniques

spécifiques.

2.4.1 Modèles traditionnels

Les modèles de recherche d'information dans les documents plats sont légion et les plus classiques ont été amplement décrits [78, chap. 4][8, chap. 2]. Nous donnons ici un aperçu rapide des trois grandes classes de modèles :

modèle booléen

– Les modèles *ensemblistes*, en particulier le modèle *booléen*, dans lequel les documents sont vus comme un ensemble de termes associés à une variable booléenne, celle-ci étant positionnée à "vrai" pour un document si le terme y est présent. Les requêtes sont des suites de termes séparés par des opérateurs booléens, et les documents pour lesquels cette expression est vérifiée sont retournés à l'utilisateur (table 2.4). Ce modèle est dit *exact* car il ne renvoie aucun document répondant partiellement à la demande et ne permet pas de classer les résultats. Cependant il a été étendu [126] pour traiter les poids des termes et autoriser un classement.

$t_1 \in \mathcal{D}$	$t_2 \in \mathcal{D}$	$t_3 \in \mathcal{D}$	**pertinence de $\mathcal{D}$**
0	x	x	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

TAB. 2.4 – Pertinence d'un document $\mathcal{D}$ pour la requête $q = t_1 \wedge (t_2 \vee t_3)$ avec le modèle booléen (où t_1, t_2 et t_3 sont des termes quelconques).

modèle vectoriel

– Les modèles *algébriques*, notamment le modèle *vectoriel* [123], dans lequel documents et requêtes sont représentés par un vecteur dont les coordonnées sont calculées dans un espace euclidien à n dimensions (n étant le nombre de termes de l'index). La longueur de la projection d'un vecteur selon une dimension est proportionnelle au poids du terme qu'elle représente (figure 2.4). La pertinence d'un document par rapport à la requête est évaluée par le degré de similarité entre le vecteur du document et celui de la requête, cette similarité étant calculée par exemple par un produit scalaire, une mesure de distance métrique ou, plus couramment, la mesure du cosinus de l'angle entre les deux vecteurs. Ce modèle permet des requêtes moins expressives (sans opérateurs) que le modèle booléen, mais son efficacité est démontrée.

modèle probabiliste

– Les modèles *probabilistes* (initiés par *Maron et Kuhns* [92]) abordent la notion de probabilité de pertinence et de non-pertinence d'un document par rapport à une requête, à travers le "principe de classement probabiliste" (*Probability Ranking Principle – PRP*) [116].

Des modèles "alternatifs" de chacune de ces trois grandes classes sont décrits par *Beaza-Yates et Ribeiro-Neto* [8, chap. 2].

2.4.2 Les langages de requêtes

Ce qui se conçoit bien s'énonce clairement
Et les mots pour le dire arrivent aisément.
Boileau, *L'art poétique, Chant I.*

L'apparition des premiers langages de requêtes permettant de prendre en compte la structure des documents est antérieure à la création du langage XML. Certains ajoutaient des opérateurs logiques ou des prédicats à des requêtes "plates" pour spécifier

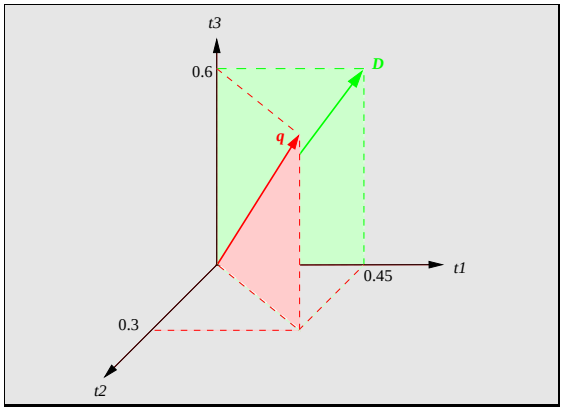


FIG. 2.4 – Représentation vectorielle d'une requête q contenant les trois termes t_1, t_2 et t_3, de poids repectifs 0.45, 0.3 et 0.6, et d'un document $\mathcal{D}$ ne contenant que les termes t_1 et t_3.

$$(\mathcal{I}(\text{"}\texttt{[play}\text{"}) \diamond \mathcal{I}(\text{"}\texttt{play]"}\text{)}) \rhd (\mathcal{I}(\text{"}\texttt{birnam}\text{"}) \diamond \mathcal{I}(\text{"}\texttt{dunsinane"}\text{}))$$

FIG. 2.5 – Représentation avec le langage de *Clarke et al.* [22] de la requête : *Trouvez les pièces contenant "Birnam" suivi de "Dunsinane"*. Les crochets `'['` et `']'` représentent le début et la fin d'un élément structurel (ici la pièce de théâtre – *play*). L'opérateur $\diamond$ signifie "suivi de", tandis que $\rhd$ exprime l'inclusion.

des contraintes structurelles (inclusion, succession, opérations) [22, 102] (figure 2.5). Les plus nombreux, issus de la communauté des bases de données, ont répondu aux particularités de la RI structurée par des langages de requêtes s'approchant de SQL, par le format et les fonctionnalités (sélection d'éléments, mais aussi jointures, aggrégats, construction d'un résultat structuré, mise à jour des documents... – notamment UnQL [14] et Lorel [94])[1].

L'arrivée en 1997 du langage XML, très vite pressenti comme un standard important pour le stockage et l'échange de données, engendre un essor des recherches sur le sujet. Celles-ci sont à l'origine de nombreux nouveaux langages de requêtes, notamment à travers le *Query Language Workshop* dès 1998 [111], où sont proposés entre autres XML-QL [31] (figure 2.6), XQL [117] et XML-GL [16], et où de nombreux articles de positionnement face à la nouvelle problématique sont présentés [96] ; puis avec le premier atelier de la conférence ACM SIGIR consacré à XML en 2000 [6], où sont présentés par exemple XIRQL [38], une amélioration de XQL, et Tequyla [3] ainsi qu'un nouvel état de l'art [88]. Lorel est également adapté à XML [52], Quilt est créé pour interroger à la fois des documents XML et des bases de données [19] et les standards du World Wide Web Consortium (W3C [157]) se dessinent avec XPath [158], XSL [159] et XQuery [166] (figure 2.7 ; une adaptation de XQuery pour la recherche documentaire est également à l'étude [167]) [2].

Beaucoup de ces langages ne traitent que des requêtes exactes (présence/absence d'un élément structurel ou d'une chaîne de caractères), mais peuvent, pour certains, être étendus avec des prédicats plus "flous". De plus ils sont en général extrêmement com-

[1] Un tour d'horizon de l'état des recherches à cette époque a été proposé par *Abiteboul* [1] et *Florescu et al.* [35].

[2] Des comparaisons de ces différents langages ont été effectuées par *Fernandez et al.* [34] et par *Bonifati et Ceri* [12].

```
WHERE
    <film>
        <titre>$t</titre>
        <réalisateur>Woody Allen</réalisateur>
        <année>$a</année>
    </film>
IN "cinematheque.xml"
CONSTRUCT
    <woody>
        <film>$t</film><sortie>$a</sortie>
    </woody>
```

FIG. 2.6 – Représentation avec XML-QL [31] de la requête : *Trouvez les films de Woody Allen et leur date de sortie dans le document* ***cinematheque.xml*** (voir la structure utilisée dans la figure 1.2).

```
let $doc := (doc("cinematheque.xml"))
for $f in $doc/cinémathèque/film[./réalisateur = "Woody Allen"]
return
<woody>
     <film>$f/titre/text()</film><sortie>$f/année/text()</sortie>
</woody>
```

FIG. 2.7 – Représentation avec XQuery [166] de la requête : *Trouvez les films de Woody Allen et leur date de sortie dans le document* ***cinematheque.xml*** (voir la structure utilisée dans la figure 1.2).

plexes, ceci étant dû au nombre important de fonctionnalités possibles. Pour contrer cette difficulté, des aides graphiques à la construction d'une requête ont été proposées [17, 136], ainsi que des langages simplifiés, basés uniquement sur l'aspect recherche d'information (sans possibilité par exemple de construction de la sortie, d'agrégation, de mise à jour des données, mais avec de véritables fonctions de recherche textuelle) [15, 130]. C'est le cas de NEXI [151], qui est dérivé du langage XPath [158]. L'importance de ces deux langages pour le travail que nous décrirons par la suite nous amène à focaliser notre attention sur eux.

2.4.2.1 XPath

XPath [158] est un langage permettant de sélectionner un ensemble d'éléments XML (ou nœuds) répondant à certaines contraintes structurelles ou textuelles, ces contraintes étant représentées par des chemins (de même type que ceux qui permettent de se déplacer dans une arborescence de fichiers informatiques) et des filtres. Il est possible notamment de caractériser les nœuds par leur position (absolue ou relative), leur type, leur contenu. XPath 1.0 distingue les nombres, les chaînes de caractères, les booléens.

Navigation. La navigation dans l'arborescence du document XML peut se faire selon différents axes. Parmi ceux-ci :

- *"child::"* représente les enfants* du nœud* contextuel (abréviation : chaîne vide) ;
- *"descendant-or-self::"* représente les descendants* du nœud contextuel ou ce nœud lui-même (abréviation : "/") ;

```
//film[contains(./réalisateur/text() = "Woody Allen")]
```

FIG. 2.8 – Représentation avec XPath [166] de la requête : *Trouvez les films de Woody Allen.* Notons que la reconstruction de la sortie étant impossible, on ne peut pas recomposer un élément nouveau contenant deux nœuds de provenances diverses, comme dans les exemples précédents (figures 2.6 et 2.7).

- *"parent::"* représente le parent* du nœud contextuel (abréviation : "..") ;
- *"ancestor::"* représente les ancêtres* du nœud contextuel ;
- *"following-sibling::"* représente les nœuds suivant le nœud contextuel et ayant le même parent que celui-ci ;
- *"attribute::"* représente les attributs* du nœud contextuel (abréviation : "@") ;
- *"self::"* représente le nœud contextuel lui-même (abréviation : ".") ;
- ...

Ainsi :

- `/child::para`, ou `/para` sélectionne tous les éléments `para` enfants de l'élément actif ;
- `/descendant-or-self::node()/child::para`, ou `//para` sélectionne tous les éléments `para` descendants de l'élément actif ;
- `parent::node()/child::para`, ou `../para` sélectionne tous les éléments `para` ayant le même parent que l'élément actif ;
- `.//liste/item` sélectionne tous les éléments `item` enfants d'un élément `liste`, lui-même descendant de l'élément actif ;
- `.//section/*` sélectionne tous les éléments, quels que soient leurs types, qui sont enfants d'un élément `section`, lui-même descendant de l'élément actif ;

Filtres. Il est possible de filtrer par :

- La position : `para[5]` sélectionne le cinquième enfant du nœud contextuel ayant le type `para` ;
- Le contenu structurel : `chapitre[titre]` sélectionne les éléments `chapitre` ayant au moins un enfant de type `titre` ;
- Le contenu structurel et textuel : `chapitre[titre="introduction"]` sélectionne les éléments `chapitre` dont un enfant de type `titre` a la valeur `"introduction"`.

Prédicats. Il est également possible d'utiliser des prédicats, dont certains sont prédéfinis dans le langage. Par exemple, le prédicat 'contains' :
`chapitre[contains(./titre, "Napoléon") and contains(./titre, "Wagram")]`
sélectionne les chapitres dont le titre contient les mots `"Napoléon"` et `"Wagram"`.

2.4.2.2 NEXI

La campagne d'évaluation INEX (cf. section 2.5.4) a choisi le langage XPath en 2003 pour permettre aux utilisateurs d'exprimer leurs besoins d'informations. Il s'est avéré que ce formalisme était trop complexe à utiliser, tandis que beaucoup de ses fonctionnalités étaient superflues. Ainsi 63 % des requêtes proposées étaient syntaxiquement ou sémantiquement incorrectes, et jusqu'à 12 tours de corrections ont été nécessaires pour parvenir à un ensemble satisfaisant de requêtes. En 2004, le langage NEXI (Narrowed

NEXI

12. Les sections (`sec`) qui parlent de moteurs de recherche sur Internet (*"Internet search engine"*) dans des articles publiés en 2001 ou 2002 :
`//article[.//pdt//yr = 2001 or .//pdt//yr = 2002]`
`//bdy//sec[about(., internet search engine)]`

70. Les résumés (`abs`) traitant de recherche d'information (*"information retrieval"*) et de bibliothèques numériques (*"digital libraries"*) :
`//article[about(.//fm//abs, +"information retrieval" "digital libraries")]`

72. Les éléments de corps d'article (`bdy`) à propos des systèmes de prédictions météorologiques, et écrits par des auteurs affiliés aux Etats-Unis :
`//article[about(.//fm//au//aff, United States of America)]//bdy//*[about(., weather forecasting systems)]`

FIG. 2.9 – Exemples de requêtes NEXI, extraites des ensembles de requêtes utilisés à INEX (requêtes 12, 70 et 72). Dans chaque cas la partie cible a été soulignée.

Extended XPath I [151]) a donc été proposé en remplacement. Grâce à ce nouveau formalisme, le taux d'erreur est tombé à 12 %.

NEXI est un sous-langage de XPath. Il le restreint en grande partie :

- Seul l'axe *descendant* est conservé ;
- Le filtrage selon le contenu est reservé aux données numériques (ex. : `//article[.//année = 1984]`) ;
- Le filtrage selon la position ainsi que selon la structure seule sont impossibles ;
- Les prédicats prédéfinis sont interdits.

Un prédicat binaire vague `about(P, C)` est cependant ajouté. Il est considéré vrai si `P` *parle de* `C`. Dans la partie `C`, NEXI accepte une liste de mots-clés. Un mot-clé peut être :

- Un simple terme ;
- Un groupe de mots, entouré alors par des guillemets ;
- Un de ces deux éléments précédé par les opérateurs `'+'` (signifiant une importance particulière du mot-clé) ou `'-'` (signifiant que le mot ne doit pas apparaître).

NEXI est donc conçu exclusivement pour la tâche de recherche d'information "basique" qui consiste à retourner des éléments satisfaisant un besoin d'information, et ce sans construire de sortie nouvelle et sans opérer de jointure ni d'agrégation. Il est présenté comme le "langage de requête le plus simple qui puisse jamais fonctionner" (*"The Simplest Query Language That Could Possibly Work"* [103]).

Des exemples de requêtes sont donnés à la figure 2.9, ainsi que dans la section 2.5.4.2. Notons qu'une requête NEXI possède deux parties distinctes :

cible
- la *cible*, l'élément qui doit être retourné à l'utilisateur, ainsi que les indications concernant son contenu ;

support
- le *support*, les autres contraintes structurelles et textuelles, permettant de mieux retrouver la cible.

Ainsi, dans la requête *"Trouvez les films dont Woody Allen est à la fois le réalisateur*

et l'acteur principal", le film est la cible tandis que le réalisateur et l'acteur sont le support. Dans les requêtes de la figure 2.9, la cible a été soulignée. Que ce soit pour la cible ou le support, l'élément peut être de type `'*'`, c'est-à-dire de type indifférent.

2.4.3 Le traitement des requêtes

La majorité des approches orientées RI présentées dans la littérature sont des adaptations des modèles traditionnels. Ces adaptations visent essentiellement à tenir compte de certaines spécificités des documents XML, à savoir les tailles très différentes que peuvent avoir les éléments, le fait que ces éléments peuvent être imbriqués les uns dans les uns (et donc non indépendants), ainsi que, plus rarement, les informations données par la structure elle-même.

Nous n'allons pas détailler les différentes propositions faites pour traiter les requêtes, mais plutôt donner un rapide tour d'horizon des caractéristiques mises en jeu. Nous présenterons ensuite le système GPX, du Queensland University of Technology, non parce que nous le jugeons représentatif, mais parce que nous avons nous-même utilisé ce système dans une partie de nos recherches.

2.4.3.1 Extension des modèles traditionnels

Les différents types de modèles (vectoriel, probabiliste, de langage, etc.) sont étendus de diverses façons pour tenir compte de la structure ou, le plus souvent, des impacts de la présence d'une structure sur les caractéristiques des unités d'information. On ajoute ainsi des paramètres supplémentaires pour ajuster les formules classiques : le nombre d'enfants d'un élément est important [45], ainsi que son type (à travers des informations sur son importance sémantique [45, 32], la fréquence de ce type d'élément dans la collection [136, 53], l'importance d'un terme dans les autres éléments du même type [53] ou tout simplement des index séparés par types d'éléments [93]).

Lorsque les éléments sont considérés comme des unités disjointes (cf. section 2.3), les scores de pertinence sont *propagés* des nœuds feuilles vers leurs ancêtres en diminuant progressivement l'influence des termes pour tenir compte du besoin de spécificité[1]. Les scores sont calculés par diverses formules (maximum des poids [4], mesure d'entropie [29], somme pondérée [48, 63], facteur d'augmentation [39]). *Lalmas* [80] utilise la théorie de Dempster-Shafer qui permet de combiner les *croyances* de plusieurs doxels par rapport à une requête.

L'information structurelle en elle-même subit rarement un traitement spécifique. Souvent les éléments retrouvés par la recherche textuelle sont simplement filtrés [138, 149] (ou pondérés) en fonction des contraintes structurelles de la requête. Pourtant certains utilisent des représentations structurées de la requête et des documents et tentent de calculer une similarité entre ces deux "arbres". Ainsi *Fuzzy XPath* [13] adapte XPath pour permettre d'ordonner un ensemble de chemins. D'autres adaptent les techniques de *"tree matching"* pour comparer les nœuds [136] ou définissent une distance entre elements [172, 109].

De nombreux modèles récents sont décrits dans les actes résultant des campagnes INEX [40, 41, 43, 42] ainsi que ceux des ateliers consacrés au sujet à la conférence SIGIR [5].

[1] Cette propagation permet de rendre compte du fait que du texte contenu par un nœud feuille est également contenu par ses ancêtres. Le score d'un élément doit donc participer au score de son parent (voir l'explication du système GPX à la section suivante).

2.4.3.2 Le système GPX

Le système GPX de *Shlomo Geva* (Queensland University of Technology) [49, 47, 48] est basé sur une indexation par fichiers inversés héritée de la RI plate et sur deux formules heuristiques de propagation des poids des termes. Le langage de requêtes qui lui est associé est NEXI (cf. section 2.4.2.2).

Indexation. Le schéma d'indexation considère les éléments comme des unités disjointes et conserve le chemin absolu des doxels contenant les termes (indexation par les arbres, cf. section 2.3). Chaque terme présent dans un document est associé à l'identifiant de ce document, à son chemin absolu (XPath) et à sa position dans le contenu textuel de l'élément (tableau 2.5).

TAB. 2.5 – Fichier inversé du système GPX.

Document	XPath	Position
e1303.xml	article[1]/bdy[1]/sec[6]/p[6]	23
e1303.xml	article[1]/bdy[1]/sec[7]/p[1]	12
e2404.xml	article[1]/bdy[1]/sec[2]/p[1]/ref[1]	1
f4576.xml	article[1]/bm[1]/bib[1]/bibl[1]/bb[13]/pp[1]	3
f4576.xml	article[1]/bm[1]/bib[1]/bibl[1]/bb[14]/pp[1]	2
g5742.xml	article[1]/fm[1]/abs[1]	7

En pratique, pour éviter les redondances (des termes et des chemins), l'index est stocké dans une base de données relationnelle comprenant quatre tables (trois tables *termes*, *chemins* et *documents* reliées ensemble par une table *liste*).

Traitement des requêtes. Le système considère que les nœuds feuilles* contiennent l'essentiel de l'information textuelle. Un score est calculé pour ces nœuds, puis les valeurs sont propagées vers le haut jusqu'à la racine*.

Le score de pertinence des feuilles est calculé avec la formule (2.1), que ce soit pour les éléments concernant le support* ou la cible* de la requête.

$$L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i} \tag{2.1}$$

Ici n est le nombre de termes de la requête présents dans l'élément, K est un petit entier ($K = 5$ dans le système) servant à favoriser les éléments contenant plusieurs termes de la requête. La somme est calculée sur l'ensemble des termes, où t_i est la fréquence du $i^{ème}$ terme de la requête dans la feuille et f_i la fréquence de ce même terme dans l'ensemble de la collection. Ainsi les termes rares contribuent plus au score que les termes communs.

Le score des éléments feuilles est alors utilisé pour calculer celui de leurs ancêtres*. Il s'agit de propager la pertinence des éléments (l'*exhaustivité*) tout en favorisant la *spécificité*. Ceci est assuré par la formule (2.2).

$$R = D(n) \sum_{i=1}^{n} L_i \tag{2.2}$$

Avec :

– n = le nombre d'enfants* jugés pertinents de l'élément (avec $L > 0$)

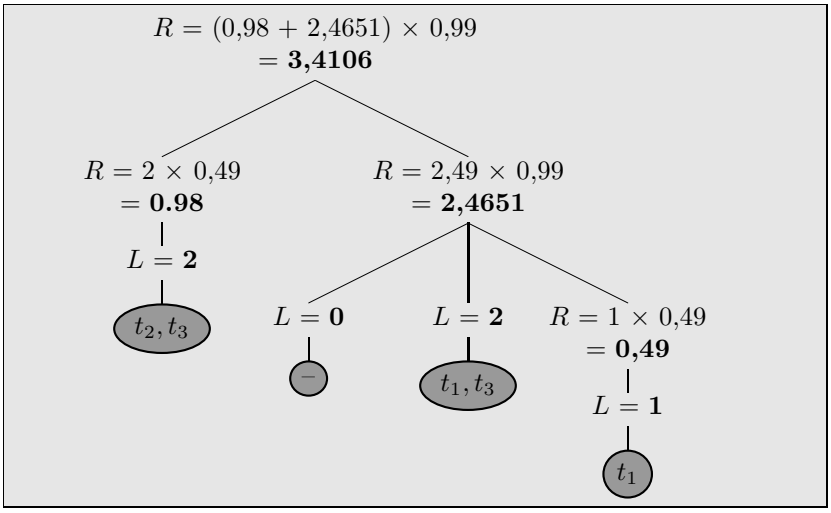


FIG. 2.10 – Système GPX. Propagation des scores des nœuds feuilles vers les nœuds supérieurs, pour la requête concernant les termes t_1, t_2 et t_3. On considère pour simplifier que chacun de ces termes a un poids de 1.

– $D(n) = \begin{cases} 0.49 & \text{si } n = 1 \\ 0.99 & \text{sinon} \end{cases}$
– L_i = le score du $i^{ème}$ enfant

Ainsi la fonction $D(n)$ assure que le score propagé décline fortement lorsqu'un élément n'a qu'un seul enfant pertinent (en effet dans ce cas il vaut mieux sélectionner cet enfant directement). En revanche un nœud comprenant plusieurs enfants ayant reçu un score non nul sera mieux classé que ses enfants (voir figure 2.10).

Les scores des parties support et ceux de la partie cible sont ensuite additionnés. Par exemple, pour la requête :

```
//A[about(.//B,C)]//X[about(.//Y,Z)]
```

le score du support `//A//B` (avec la condition `C`) sera ajouté à celui de la cible `//A//X//Y` (condition `Z`).

Les contraintes structurelles font l'objet d'un simple filtrage (vérification du type de la balise) si l'on souhaite les appliquer de façon stricte.

Le système GPX a participé aux différentes campagnes d'INEX avec de très bons résultats[47, 48].

2.5 Evaluation

Pour évaluer et comparer les systèmes de recherche d'information, les critères de performance (ou d'efficience, selon un anglicisme usité) habituels en informatique, c'est-à-dire les mesures du temps de réponse et de l'espace disque occupé, ne sont plus suffisants (même si les applications des modèles se doivent de considérer ces aspects). En plus de la performance, d'autres indicateurs sont l'*environnement* (notamment l'ergonomie du système en termes de facilité d'utilisation et de présentation des résultats) mais surtout les *résultats* (*outcome*) [95, chap. 16]. En effet, il n'est pas possible de formuler un jugement qualitatif binaire sur la sortie du système (sortie correcte ou sortie incorrecte). Dans ce qui suit, nous considérons que le critère unique d'évaluation

des systèmes de recherche d'information est leur *efficacité*, c'est-à-dire leur capacité à trouver des documents répondant au besoin de l'utilisateur [69].

2.5.1 L'évaluation ou la dictature de l'humain

Pertinent *: Qui se rapporte exactement à ce dont il est question ; approprié*
Dictionnaire Larousse.

Comme nous l'avons déjà noté, la plupart des paramètres en jeu dans le processus de recherche d'information sont des paramètres humains. Notamment c'est un humain qui pose une question et reçoit (et juge) la réponse. Un être humain est, en l'occurrence, caractérisé par :

- sa subjectivité : pour un même besoin d'information, deux humains formuleront des requêtes (très) dissemblables ; ils estimeront la qualité des réponses retournées de deux façons tout aussi différentes. Les variables entrant en jeu sont notamment le passé des personnes, leur caractère, leur connaissance initiale du sujet, leur niveau de langue, leur niveau d'exigence.
- sa versatilité : pour un même besoin d'information, un seul humain formulera des requêtes (très) dissemblables à divers moments de sa journée ; il estimera la qualité des réponses retournées de plusieurs façons tout aussi différentes. Il s'agit là de variations d'humeur, d'éveil, de pensées parasites ou même d'évolution de la connaissance du sujet entre deux instants.

Déjà ardue, la tâche apparaît donc insurmontable quand on se souvient que les documents dans lesquels l'information est recherchée sont eux-mêmes le fruit d'humains subjectifs et versatiles, et qu'à toutes ces notions s'ajoutent les phénomènes linguistiques de variations morphologiques, lexicales, sémantiques, syntaxiques.

Pourtant seuls des humains peuvent formuler des requêtes (c'est la finalité de la discipline), seuls des humains peuvent écrire les documents et seuls des humains peuvent juger les réponses. Si ce n'était pas le cas les systèmes de recherche d'information n'auraient plus besoin d'être perfectionnés.

Toutes ces réflexions mènent à la conclusion qu'il n'existe pas un seul ensemble possible de réponses à la question posée, et même qu'il n'existe pas de bonne réponse dans l'absolu.

pertinence

En outre, la notion de *pertinence* d'un document ou d'une portion de document, qui est au cœur du principe de l'évaluation des systèmes de RI, est relativement floue. Souvent décrite comme un synonyme d'*utilité* ou de *satisfaction* de l'utilisateur [66], elle est elle-même un phénomène social et cognitif complexe [26] et n'a pas de définition consensuelle. Des formalisations ont été proposées [26, 169], ainsi que des études sur les variables affectant les jugements humains sur la pertinence [113], ou les diverses facettes qu'elle peut prendre [128, 129, 98, 99], mais le fait est que la pertinence est un concept insaisissable car, comme son inventeur, subjectif et versatile.

En l'absence de possibilité d'attribution d'une valeur binaire par une seule personne à un moment donné, elle serait dans l'idéal le résultat d'une *"tendance consensuelle, centrale"* de plusieurs juges humains [9, chap. 4], ce qui est complexe à obtenir en pratique.

A ces nombreux problèmes vient s'ajouter celui de la taille constamment croissante de la masse de documents mise en jeu. La collection d'INEX 2005 compte 760 méga-octets[1], celle d'INEX 2006 environ 10 giga-octets (100 avec les images), et certaines

[1] Voir la section 2.5.4.

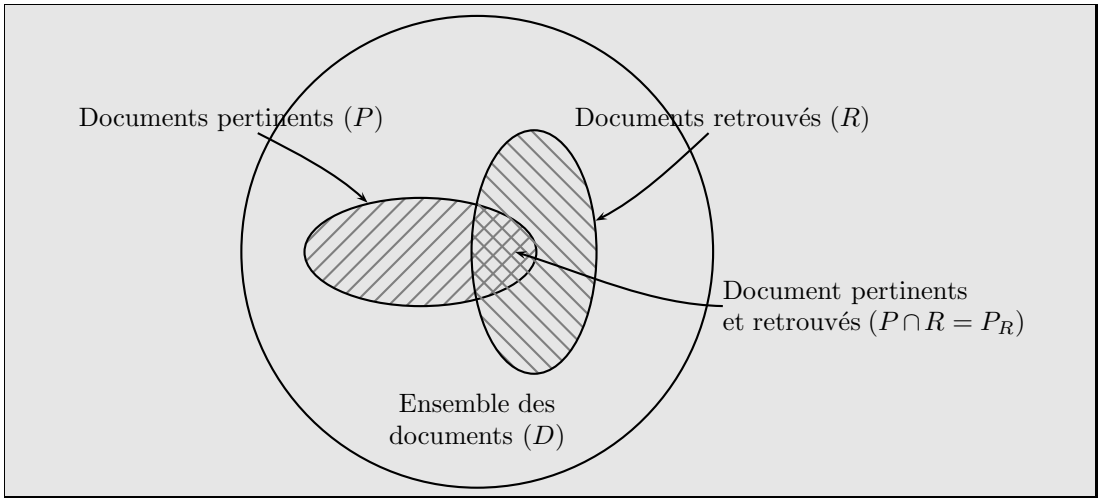


FIG. 2.11 – Répartition des documents face à une requête.

collections utilisées par TREC [154] dépassent la dizaine de giga-octets, soit l'équivalent du texte de plusieurs encyclopédies. Il est donc depuis longtemps hors de question qu'un expert humain cherche dans l'ensemble de la collection les réponses pertinentes à une requête donnée. Pourtant les "bonnes" réponses doivent être connues si l'on veut évaluer correctement les systèmes. Pour pallier (de façon très imparfaite) ce problème insoluble la méthode du *pooling* a été mise en place [142, 155], qui consiste en une fusion "intelligente" des résultats des participants en un ensemble de documents qui devra être jugé. Les documents non jugés sont considérés non pertinents. Pourtant *Zobel* [175] a estimé qu'au mieux 50 à 70 % des documents pertinents étaient détectés par le pooling.

pooling

Autant dire qu'une évaluation objective et impartiale des systèmes de RI est une tâche perdue d'avance, et que toutes les méthodologies, toutes les procédures, toutes les métriques mises en place ne seront jamais que des pis-aller[1].

2.5.2 Mesures d'évaluation des systèmes classiques

Une mesure d'évaluation appropriée doit estimer la faculté des systèmes à trouver des documents pertinents, sans pour autant favoriser ceux qui retournent plus de documents (et donc potentiellement plus de documents pertinents) [145]. Il est donc nécessaire de considérer aussi les documents non pertinents dans une métrique.

Les mesures combinées de *rappel* et de *précision* [76] sont les mesures les plus simples et les plus utilisées qui tiennent compte de ces deux impératifs (retrouver des documents pertinents, ne pas retrouver de documents non pertinents).

Soient, pour une requête donnée et un système donné (voir figure 2.11) :

- P l'ensemble des documents de la collection qui sont pertinents par rapport à cette requête.
- R l'ensemble des documents retrouvés par le système pour cette requête.
- $P_R = P \cap R$, l'ensemble des documents pertinents retrouvés par le système.

[1] *Swanson* [144] dresse une liste de neuf "postulats d'impuissance" (*"Postulates of Impotence"*) dans un article tristement intitulé : "La recherche d'information et le futur d'une illusion" (*"Historical Note : Information Retrieval and the Future of an Illusion"*). Seuls trois "postulats de fertilité" (*"Postulates of Fertility"*) viennent contrebalancer ce noir tableau.

précision

La précision mesure le taux de documents pertinents parmi tous les documents retrouvés par le système :

$$précision = \frac{|P_R|}{|R|}$$

rappel

Le rappel mesure le taux de documents pertinents retrouvés par le système parmi l'ensemble des documents pertinents de la collection :

$$rappel = \frac{|P_R|}{|P|}$$

Ces deux valeurs sont indissociables et concurrentes, puisque souvent l'amélioration de l'une d'elle se fait au détriment de l'autre. Une caractéristique intéressante est d'ailleurs l'évolution de la précision en fonction du rappel (précision quand 10 % des documents pertinents ont été retrouvés, puis 20 %, etc.). Ces informations sont regroupées dans la *courbe de rappel-précision*.

Universellement utilisés, précision et rappel posent cependant quelques problèmes. Parmi eux :

- La mesure du rappel nécessite de connaître en théorie l'ensemble des documents pertinents par rapport à une requête dans toute la collection. Nous avons vu que cela était impossible.
- Le rappel n'est pas une mesure qui reflète la satisfaction de l'utilisateur moyen. Celui-ci est rarement intéressé par la connaissance de l'ensemble des documents pertinents, mais souhaite plutôt trouver "vite" (donc bien classés) quelques documents pertinents (voire un seul).
- La précision ne favorise pas les systèmes qui placent les documents pertinents dans les premiers rangs.
- Précision et rappel nécessitent un ordre *total* dans la liste de documents retournés, ce qui n'est souvent pas le cas.
- La précision et le rappel sont indissociables et reflètent des aspects différents des documents retournés. Une mesure unique serait la bienvenue.
- L'interactivité proposée par de nombreux systèmes n'est pas prise en compte par ces mesures.

Pour pallier ces inconvénients, de nombreuses métriques héritées de la précision et du rappel ont été proposées, comme la *précision moyenne*, la *R-précision*, la *moyenne harmonique*, la *mesure E* [152], mais aussi d'autres mesures plus orientées vers l'utilisateur, au nombre desquelles la *couverture* et la *nouveauté* [78], la *"expected search length"* [26], la *probabilité de pertinence* et la *précision attendue* [112]. Ces mesures (et d'autres) sont décrites avec plus de détails dans divers supports [8, chap. 3][78, chap. 8].

Nous allons nous intéresser plus précisément aux mesures d'évaluation appliquées pour les systèmes de RI dans les documents XML.

2.5.3 Mesures d'évaluation des systèmes de RI structurée

L'évaluation des systèmes de recherche d'information structurée pose de nouvelles difficultés en termes de métriques et de comportements humains :

- On considère que l'unité d'information est l'élément XML, ou *doxel*. L'évaluation doit donc porter sur chaque élément. Dans la collection INEX 2005, à laquelle nous ferons souvent référence (cf. section 2.5.4), chaque document contient en moyenne 1500 doxels. Même si le volume de données reste inchangé entre un contenu plat et le même contenu structuré, le nombre d'éléments qu'il est potentiellement

nécessaire de considérer augmente de façon non négligeable la charge de travail des juges humains.

- Comme nous l'avons déjà remarqué, deux dimensions sont nécessaires à l'estimation de la pertinence d'un élément : la première dimension permet de déterminer si (ou *à quel point*) le doxel contient des informations intéressantes au regard de la requête, la seconde si (ou *à quel point*) sa taille est appropriée, c'est-à-dire s'il aurait pu être ou non possible, en choisissant un élément descendant ou ancêtre, de concentrer de manière plus précise l'information pertinente.
- Il est normal de considérer que si un élément de petite taille est pertinent, alors ses ancêtres le sont aussi (même s'ils sont moins *spécifiques*). Pourtant le retour par un système de plusieurs éléments se chevauchant (c'est-à-dire, les uns étant ancêtres des autres) pose un problème. Si l'on intègre de nombreux éléments imbriqués dans la *base de rappel* (la liste de tous les doxels pertinents), cela encourage les concepteurs des systèmes à renvoyer des informations très redondantes. Ceci est contre-productif vis-à-vis de l'utilisateur (qui ne sera pas satisfait de visualiser la même information plusieurs fois en parcourant la liste des résultats[1]), mais aussi pour la comparaison entre les systèmes [72].

chevauchement

- Enfin, nous avons déjà abordé le problème du caractère auto-explicatif des éléments retournés (2.2.1). Il semble impossible de s'assurer qu'un élément permet de répondre à la requête de façon indépendante du reste du document. L'entorse à cette règle doit tout de même être limitée, pour ne pas nuire à la lisibilité des résultats.

Les métriques d'évaluation sont loin d'être stabilisées et consensuelles pour la RI structurée. Leur évolution est indissociable de la campagne INEX. Toutes les propositions ont été faites dans ce cadre, et les métriques retenues chaque année pour l'évaluation "officielle" ont eu de fait un succès supérieur. De plus, beaucoup des formules proposées dépendent fortement des tâches spécifiques à INEX d'une part et du mode de jugement de l'exhaustivité et de la spécificité d'autre part. C'est pourquoi, après cette introduction de la problématique générale, nous abordons les métriques au sein de la section consacrée à INEX (2.5.4.4).

2.5.4 La campagne d'évaluation INEX

INEX (INitiative for the Evaluation of XML retrieval [90] [2]) est une campagne d'évaluation visant, à la manière de TREC (Text REtrieval Conference [154] [3]) ou CLEF (Cross Language Evaluation Forum [104] [4]) pour les documents plats, à permettre une évaluation et une comparaison aussi rigoureuses que possible des systèmes de recherche d'information dans les collections XML orientées documents.

INEX

Ainsi INEX fournit un ensemble de documents, un ensemble de requêtes et des *jugements de pertinence*, c'est-à-dire les estimations humaines des éléments pertinents concernant chaque requête. Nous allons consacrer quelques pages à décrire le fonctionnement global d'INEX, qui est devenu l'organe central et le promoteur majeur des progrès en recherche d'information XML.

[1] Cependant cette difficulté n'est pas nouvelle, puisque des informations rigoureusement identiques (des citations par exemple) peuvent apparaître dans des documents différents. Elle a toujours été contournée (y compris à INEX) en demandant aux évaluateurs de considérer les documents indépendamment les uns des autres.

[2] http://inex.is.informatik.uni-duisburg.de/2005/.

[3] http://trec.nist.gov.

[4] http://www.clef-campaign.org.

2.5.4.1 Les documents utilisés par INEX en 2005

La collection regroupait lors de la campagne de 2005 (quatrième édition) des articles scientifiques issus de journaux fournis par l'IEEE Computer Society, écrits en XML avec une DTD unique. 16819 articles balaient ainsi de nombreux domaines de l'informatique entre 1995 et 2004. Des extraits d'un article sont fournis à la figure 2.12.

Dans un article type, des méta-données concernant la publication (auteur, titre, journal mais aussi numéro ISBN, numéros de pages, prix, etc.) sont d'abord fournies, puis vient le texte lui-même, séparé en divers types de sections (`'sec'`), sous-sections (`'ss1'`, `'ss2'`), paragraphes (`'p'`), figures (`'fig'`), avec des balises distinguant notamment les références à des figures, à la bibliographie, à des notes de base de page (`'ref'`), ainsi que des indications de forme, comme la mise en caractères italiques (`'it'`) ou gras (`'b'`). Puis les annexes sont essentiellement composées des éléments bibliographiques cités dans le corps du document et des indications concernant les auteurs.

La DTD comprend 192 balises différentes et un article est composé en moyenne d'environ 1500 éléments.

2.5.4.2 Les requêtes

topic

Une requête est la représentation la plus fidèle possible d'un besoin d'information. Dans INEX ces requêtes (appelées aussi *topics*) sont proposées par les participants, donc experts du domaine de la recherche d'information et possédant des connaissances diverses en informatique (domaine de la collection), ce qui présente un biais non négligeable (d'autant plus que les participants sont également les juges – cf. section 2.5.4.3). Cependant il est demandé que ces requêtes *"reflètent les besoins réels de systèmes opérationnels"*[1].

De façon générale, il est possible de distinguer deux types de requêtes :

contenu seul

- Celles concernant le contenu seul, pour les utilisateurs qui ignorent (ou qui souhaitent ignorer) la structure des documents.
  Exemple : *Les algorithmes de codage pour la compression de texte ou d'index.*[2]

contenu/ structure

- Celles concernant le contenu et la structure, incorporant donc des contraintes structurelles.
  Exemple : *Nous recherchons des résumés de documents concernant le data mining écrits par Jiawei Han.*[3]

Ceci a conduit à un format de requêtes dont des exemples sont donnés dans les figures 2.13, 2.14 et 2.15, et dont les champs importants sont [140] :

- *description*, une courte description du besoin d'information, rédigée en Anglais.
- *narrative*, une explication détaillée du besoin d'information, rédigée en Anglais, expliquant les raisons de la requête, ce qui sera jugé pertinent et éventuellement ce qui ne le sera pas.
- *title*, une liste de mots-clés concernant le besoin d'information.
- *castitle*, un titre écrit en NEXI (cf. section 2.4.2.2).

Il est important de préciser que la *description* doit contenir *"les mêmes termes et les mêmes conditions structurelles que ceux apparaissant dans les parties* title *et* castitle*"*[4].

[1] "Topics should reflect the real need of operational systems." [140]
[2] "Any type of coding algorithm for text and index compression." (Topic 162, INEX 2004).
[3] "We are looking for the abstracts of the documents about data mining and written by Jiawei Han." (Topic 132, INEX 2004).
[4] "The description [...] should contain the same terms and the same structural requirements that appear in the <title> and in the <castitle>". [140]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE article SYSTEM "/usr/projects/inex/2005/inex/dtd/xmlarticle.dtd">
<article>
 <fno>A1067</fno>
 <doi>10.1041/A1067s-1998</doi>
 <fm>
  <hdr>
   <hdr1>
    <ti>IEEE ANNALS OF THE HISTORY OF COMPUTING</ti>
    <crt><issn>1058-6180</issn>/98/$10.00
          <cci><onm>&copy; 1998 IEEE</onm></cci>
    </crt>
   </hdr1>
   <hdr2>
    <obi><volno>Vol. 20</volno>,<issno>No. 1</issno></obi>
    <pdt><mo>JANUARY-MARCH</mo>
      <yr>1998</yr></pdt>
    <pp>pp. 67-76</pp>
   </hdr2>
  </hdr>
  <tig>
   <atl>Calculators</atl>
   <pn>pp. 67-76</pn>
  </tig>
 </fm>
 <bdy>
  <sec>
   <st>THE CALCUMETER</st>
   <ip1><b>by Robert Otnes</b></ip1>
   <ip1>
      The Calcumeter adder appeared shortly after 1900, prospered for a while,
      and then apparently stopped production by 1920. Before its demise, as many
      as 100,000 adders may have been made. The Calcumeter is an interesting,
      well-made, small adding machine that is sought by collectors.
   </ip1>
   <lc>
    <li>
     <p>
       Walsh was the inventor. He held the two patents on the machine in his name.
       Early machines are labeled Morse and Walsh, whereas later ones are labeled
       only Morse, perhaps indicating that Walsh dropped out of the business. His
       name is mentioned in the 1905 <it>Trenton City Directory</it> (TCD)
       advertisement for the Calcumeter, but it was omitted in 1906.
     </p>
    </li>
    <li>
     <p>
       Morse apparently managed the business. At least one of the Calcumeter ads
       has his picture. The city directories mention him as being business manager
       for the State Department of Public Instruction in 1919; this is at the end
       of the era of the Calcumeter.
     </p>
    </li>
```

FIG. 2.12 – Exemple de document XML de la collection INEX (extraits, 1/2)

```
    </lc>
    <ss1>
      <st>The Machine</st>
      <ip1>
        Calcumeters are shown in <ref rid="a10671" type="fig">Figs. 1</ref> and
        <ref rid="a10672" type="fig">2</ref>.<ref rid="a10671" type="fig">Fig. 1
        </ref> shows a relatively early machine without the reset mechanism. This
        machine is marked <scp>THE CALCUMETER</scp>.
        <ref rid="a10672" type="fig">Fig. 2</ref> shows a later example that
        includes a reset mechanism. Note that this machine is marked <scp>THE
        STANDARD DESK CALCUMETER</scp> rather than <scp>THE
        CALCUMETER</scp>. Both machines are about the same size.
      </ip1>
    </ss1>
  </sec>
 </bdy>
 <bm>
  <ack>
    <h>Acknowledgment</h>
    <ip1>
      I would like to thank Rolf Ziegler, my colleague in the Section on IBM
      History of the IBM Club B&ouml;blingen.
    </ip1>
  </ack>
  <bib>
    <bibl>
      <h>References</h>
      <bb id="biba10671">
        <au>
          <fnm>M.T.</fnm><snm>Olivier</snm>
        </au>
        <atl>"Machines a Calculer du Docteur Roth,"</atl>
        <ti>Bulletin De La Societe D'Encouragement Pour L'Industrie Nationale,</ti>
        <pdt><mo>Sept.</mo><yr>1843,</yr></pdt>
        <loc><cty>Paris.</cty></loc>
      </bb>
      <bb id="biba10672">
        <au>
          <fnm>E.</fnm><snm>Martin</snm>
        </au>
        <ti>Die Rechenmaschinen und ihre Entwicklungsgeschichte,</ti>
        <obi>1. Band, 1924,</obi><pp>p. 63</pp>
        <pdt><yr>1992.</yr></pdt>
      </bb>
    </bibl>
  </bib>
 </bm>
</article>
```

FIG. 2.12 – Exemple de document XML de la collection INEX (extraits, 2/2)

```
<inex_topic topic_id="205" query_type="CO+S" ct_no="12">
    <InitialTopicStatement>McLuhan</InitialTopicStatement>
    <title>marshall mcluhan</title>
    <castitle>//bdy//*[about(., "Marshall McLuhan")]</castitle>
    <description>
        Find information about the relevance of Marshall McLuhan's ideas for current
        digital technologies.
    </description>
    <narrative>
        I am writing an essay on the influence of new media icon Marshall McLuhan
        on digital technologies. I'm seeking information describing how McLuhan's views
        have influenced current digital technologies. To be relevant, a retrieved item should
        discuss some aspect of Marshall McLuhan's visionary ideas or famous one-liners
        in the context of current digital technologies. Retrieved elements that merely cite
        some of McLuhan's work are non-relevant, as are elements that discuss ideas not
        originating from McLuhan.
    </narrative>
</inex_topic>
```

FIG. 2.13 – Exemple de requête CO+S (Topic 205 d'INEX 2005).

En 2005, les organisateurs ont souhaité étudier l'impact des contraintes structurelles sur l'efficacité de la recherche. Ils ont donc ajouté la possibilité de donner des indices concernant la structure des résultats souhaités, y compris dans les requêtes sur le contenu seulement. Cette réflexion a mené aux deux types de requêtes suivants :

- *Contenu seulement et structure* (*"Content-Only + Structure"* – *CO+S*), dont le nom étrange est dû à l'évolution historique de la tâche. Il qualifie les requêtes se rapportant au contenu textuel seul (*title* et *description*), mais pour lesquels l'utilisateur peut malgré tout "aider" le système par des indications structurelles (dans le *castitle* – voir un exemple figure 2.13). *CO + S*
- *Contenu et Structure* (*"Content and Structure"* – *CAS*) permet aux utilisateurs connaissant la structure des documents de formuler des contraintes concernant les types d'éléments souhaités (voir un exemple figure 2.14). *CAS*

De plus, dans le but d'évaluer de façon précise l'action de chaque contrainte structurelle, chaque requête comportant plusieurs références à la structure (résumé + auteur dans le topic 132 sur le data mining ci-dessus, deux paragraphes dans l'exemple de la figure 2.14) doit être accompagnée d'une sous-requête par contrainte (une des sous-requêtes correspondant au topic 256 de la figure 2.14 est reproduite à la figure 2.15).

2.5.4.3 Les jugements de pertinence

Les jugements concernant la pertinence des éléments renvoyés par les systèmes sont effectués par les participants eux-mêmes, à l'aide d'une interface mise à disposition en ligne [108, 81]. Les différents niveaux et degrés de pertinence, ainsi que la forme prise par l'interface d'évaluation ont changé chaque année depuis les débuts d'INEX [57, 73, 91]. Nous décrivons ici la méthodologie mise en œuvre en 2005. Deux dimensions ont été utilisées [44] :

- l'*exhaustivité**, qui décrit à quel point l'élément traite du sujet de la requête ; *exhaustivité*
- la *spécificité**, qui décrit à quel point l'élément se focalise sur le sujet de la requête. *spécificité*

```
<inex_topic topic_id="256" query_type="CAS" ct_no="61">
    <InitialTopicStatement>To Find Information regarding data embedding using
    watermarking.</InitialTopicStatement>
    <title></title>
    <castitle>
        //article[about(.//p,"data embedding")] //p[about(.,watermarking)]
    </castitle>
    <description>
        We are looking for paragraphs describing watermarking in articles which describe
        data embedding.
    </description>
    <narrative>
        In today's world the issue of data security is highly significant. One such tech-
        nique to ensure data security is steganography where data is embedded in various
        media files like images, sound files, video files and so on. A commonly used data
        embedding technique is Watermarking where data can be efficiently hidden in
        a file without the changes being visible to a common person. Therefore I am
        interested in retrieving information regarding Watermarking technique for data
        embedding.Any information regarding data embedding that does not expalin wa-
        termarking is irrelevant. The purpose of this retrieval is to make a presentaion on
        watermarking.
    </narrative>
</inex_topic>
```

FIG. 2.14 – Exemple de requête CAS (Topic 256 d'INEX 2005).

```
<inex_topic topic_id="256" query_type="CAS" ct_no="61">
    <InitialTopicStatement>To Find Information regarding data
    embedding.</InitialTopicStatement>
    <title>embedding data</title>
    <castitle>//article//p[about(.,embedding data)]</castitle>
    <parent>
        //article[about(.//p,embedding data)]//p[about(.,watermarking)]
    </parent>
    <description>
        We are looking for paragraphs in articles which contain embedding data
    </description>
    <narrative>
        I am interested in retrieving information regarding techniques for embedding data.
        Any information that does not explain data embedding is irrelevant
    </narrative>
</inex_topic>
```

FIG. 2.15 – Exemple de requête CAS ayant pour parent une autre requête (Topic 272 d'INEX 2005, dépendant du Topic 256 de la figure 2.14).

Ces dimensions sont presque indépendantes l'une de l'autre. Par exemple, une question peut être traitée de façon satisfaisante dans un élément (exhaustivité forte) mais rester malgré tout un sujet marginal de cet élément (spécificité basse).

L'*exhaustivité* est jugée par l'humain sur une échelle à quatre niveaux [81] :

- Pas exhaustif (*not exhaustive*) : aucune correspondance entre la requête et l'élément ;
- Partiellement exhaustif (*partially exhaustive*) : l'élément ne traite que certains aspects de la requête ;

- Très exhautif (*highly exhaustive*) : l'élément traite tous ou presque tous les aspects de la requête ;
- Trop petit (*too small*) : l'élément peut être considéré comme pertinent à certains égards, mais la partie pertinente est trop petite pour être considérée.

La *spécificité* est depuis 2005 considérée comme une dimension continue représentée par une valeur réelle comprise entre 0 et 1.

En pratique, la tâche du juge humain consiste, pour une requête donnée et un document proposé, à surligner le contenu qu'il juge pertinent, puis à attribuer un niveau d'exhaustivité aux doxels contenant du texte surligné. La spécificité est alors automatiquement calculée comme un ratio entre la longueur de texte surligné dans chaque élément et la longueur totale du texte [71]. De plus certaines règles automatiques spécifient des contraintes intuitives concernant l'exhaustivité. En particulier les ancêtres d'un élément doivent avoir une exhaustivité égale ou supérieure à cet élément.

2.5.4.4 Les métriques d'évaluation

Comme nous l'avons déjà expliqué (section 2.5.3), un axe important des recherches autour d'INEX consiste en la définition de métriques d'évaluation adaptées aux spécificités des documents XML.

Valeur de la pertinence. L'ensemble des mesures proposées utilise une valeur unique combinant exhausitivité et spécificité. Cette valeur unique est calculée par des fonctions différentes selon la métrique.

Parmi ces fonctions, l'agrégation *stricte* est la plus stable :

$$f_{strict}(s,e) = \begin{cases} 1 & \text{si } e = e_{max} \text{ et } s = s_{max} \\ 0 & \text{sinon} \end{cases} \tag{2.3}$$

s et e sont les valeurs de spécificité et d'exhaustivité pour un élément, et s_{max} et e_{max} sont les valeurs attribuées à la spécificité et l'exhaustivité maximale (3 en 2004, respectivement 1 et 2 en 2005). La pertinence est ainsi convertie en une valeur binaire.

L'agrégation *généralisée* propose une échelle de valeurs entre 0 et 1. Par exemple, en 2005, où s était un réel entre 0 et 1 et e un entier égal à 0, 1 ou 2 (2 correspond à "très exhaustif"), l'agrégation généralisée était calculée par le simple produit des deux valeurs :

$$f_{gen}(s,e) = s \times e \tag{2.4}$$

Les années précédentes cette fonction était une adaptation de celle-ci à des valeurs discontinues de s. D'autres fonctions binaires autorisant différents degrés pour e ou s ont également abouti à autant de mesures d'évaluation en 2004 [70]. Par ailleurs, *Kazai et al.* [72] proposent une fonction donnant plus de poids à la spécificité, et *Hiemstra et Mihajlović* [62] une fonction binaire $f_{liberal}$ bien plus "généreuse" que f_{strict}.

N.B. : Désormais nous appellerons un élément *pertinent* un élément qui a un score non nul avec la fonction d'agrégation utilisée.

Métriques. Nous ne nous étendons pas sur l'historique des métriques utilisées au cours des premières années d'INEX, et nous nous focalisons sur celles utilisées en 2005,

à savoir le *gain cumulé étendu normalisé* (*nxCG*) et la courbe d'*effort-précision* par rapport au *gain-rappel*.

La mesure du gain cumulé étendu (*xCG*) mesure le gain, c'est-à-dire le score de pertinence, calculé par une des fonctions présentées ci-dessus, et accumulé au fur à et mesure de la progression dans la liste ordonnée d'éléments retournés.

$$xCG(i) = \sum_{j=1}^{i} xG(j) \tag{2.5}$$

où $xG(j)$ est le score obtenu pour l'élément classé en $j^{ème}$ position par le système évalué.

nxCG

On compare ce gain cumulé avec celui qu'aurait dû atteindre le système s'il avait produit une liste triée optimale[1], pour obtenir le gain cumulé étendu normalisé :

$$nxCG(i) = \frac{xCG(i)}{xCI(i)} \tag{2.6}$$

où $xCI(i)$ est le gain cumulé idéal. On peut comparer cette mesure au rappel* pour les documents plats.

La métrique complémentaire, l'*effort-précision*, représente l'effort (en nombre de liens à visiter) qu'un utilisateur doit fournir pour parvenir à un gain donné r :

$$ep(r) = \frac{e_{ideal}}{e_{run}} \tag{2.7}$$

où e_{run} (resp. e_{ideal}) est le rang auquel le gain r est atteint par le système (resp. par la liste optimale).

Parallèlement, le gain-rappel est le rapport entre le gain obtenu en une position i de la liste et le gain total auquel on peut parvenir :

$$gr(i) = \frac{xCG(i)}{xCI(n)} \tag{2.8}$$

ep/gr

où n est le nombre total d'éléments pertinents. Ainsi on obtient la courbe *ep/gr* en calculant l'effort-précision à différents points de *gain-rappel* (comme pour la courbe rappel/précision, à 10 %, puis 20 %, etc.).

MAep

Enfin, la mesure MAep (Mean Average Effort Precision) est la moyenne des valeurs d'effort-précision pour chaque rang auquel un nouvel élément pertinent est renvoyé par le système évalué.

2.5.4.5 Les différentes tâches

La tâche *ad-hoc*. La tâche principale de la campagne INEX est la tâche *ad-hoc*. Il s'agit de trouver les éléments XML contenant les informations pertinentes quant aux besoins d'information exprimés par les titres des requêtes, et d'ordonner ces éléments dans l'ordre supposé de leur pertinence. En 2005, plusieurs aspects de cette tâche ont été explorés, dans la but d'étudier l'influence de l'utilisation de la structure (dans les requêtes) sur l'efficacité de la recherche.

[1] C'est-à-dire une liste contenant tous les éléments pertinents au début, par ordre décroissant de pertinence. Le problème du chevauchement a conduit à d'autres définitions de cette liste optimale [72].

En effet, outre la distinction CO+S / CAS déjà exprimée, les organisateurs ont séparé la partie CAS en quatre sous-parties, qui distinguent le caractère strict ou souple de l'interprétation des contraintes structurelles dans la cible* et le support* : SSCAS correspond à une interprétation stricte des deux éléments, SVCAS à une interprétation stricte de la cible et souple du support, et VSCAS et VVCAS représentent les deux autres combinaisons.

La tâche CO, elle, propose trois variantes : *Focussed*, dans laquelle le système doit trouver le *meilleur* élément dans un chemin donné (le chevauchement est donc interdit) ; *Thorough* autorise les chevauchements, mais bien entendu les meilleurs éléments doivent être mieux classés ; enfin, la stratégie *Fetch and Browse* consiste à renvoyer un article (un document) entier et à lister à la suite les éléments les plus pertinents de cet article.

Autres tâches. Des tâches annexes ont été ajoutées au fil des ans à la tâche *ad-hoc* pour arriver au nombre de six en 2005 :

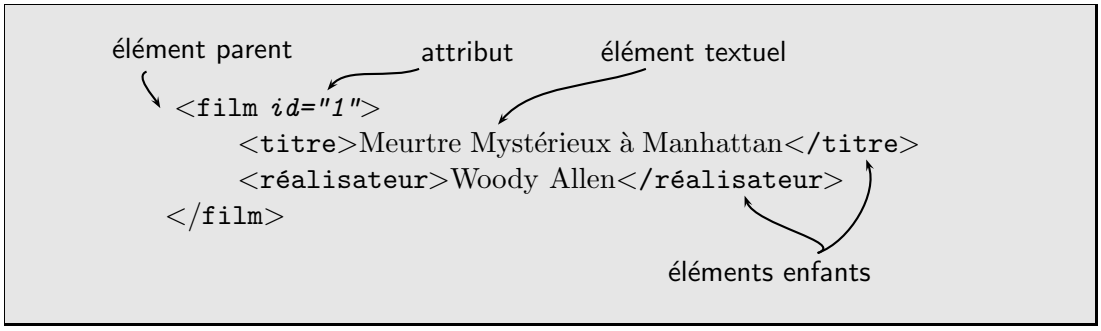
- *Relevance Feedback*, ou comment utiliser les résultats d'un moteur pour améliorer la requête.
- *Heterogenous Collections*, ou comment permettre aux utilisateurs de faire des recherches sur plusieurs collections différentes (aux DTD différentes). Pour cela d'autres ensembles de documents ont été mis à la disposition des participants.
- *Interactive*, ou comment rendre la recherche dans les collections XML accessibles à tous les types d'utilisateurs, en termes d'interface de requête et de présentation des résultats.
- *Document Mining*, ou comment appliquer les techniques de classification et de clusterisation aux collections XML.
- *Multimedia*, ou comment utiliser les contenus non textuels pour la recherche d'information XML.
- *Natural Language Processing*.

Annexes

Le langage XML

Les bases de XML

Un document XML peut se représenter sous la forme d'une arborescence d'éléments. Cette arborescence comporte une racine (unique), qui englobe l'ensemble des autre éléments, ainsi que des branches et des feuilles, qui forment la structure même du document. Ils peuvent contenir du texte, ou bien d'autres éléments (ses "enfants") :



Les autres notions de la terminologie XML sont données au chapitre 1.

commentaires XML

Il est possible d'ajouter des commentaires, qui commencent par `'<!--'` et se terminent par `'-->'`. Ils peuvent être placés à n'importe quel endroit tant qu'ils se trouvent à l'extérieur d'une autre balise.

Certains caractères ayant un sens précis en XML, on a recours à des entités prédéfinies pour les remplacer lorsqu'on en a besoin. Par exemple `'&'` remplace le caractère `'&'`, `'<'` le signe `'<'`, `'>'` le signe `'>'`, etc. D'autres entités peuvent être définies par l'utilisateur. Un autre moyen d'échapper aux caractères prédéfinis est de les insérer dans des sections CDATA :

<exemple><![CDATA[Une section CDATA peut contenir des < et des >.]]>
</exemple>

Il est aussi possible de définir des règles plus strictes indiquant quelles sont les séquences et imbrications de balises autorisées. Cela se fait à l'aide d'une DTD ou d'un Schéma.

La DTD

DTD

La DTD est une description générique d'une classe de documents XML. Elle impose des contraintes concernant le nom des balises à employer et les relations que celles-ci possèdent entre elles : inclusion, ordre, possibilité d'absence, etc.

On peut dire par abus de langage que deux documents obéissant à la même DTD ont la même structure. Cependant il faut garder à l'esprit qu'ils peuvent présenter des

différences sensibles dans leur structuration. Ainsi, pour un DTD représentant des articles scientifiques, comme celles de la collection INEX 2005, un document peut contenir une bibliographie ou un résumé, tandis que l'autre non ; l'un peut contenir trois sections dans le corps du texte, l'autre quatre, etc.

On dit d'un document respectant une DTD donnée qu'il est *valide* par rapport à cette DTD. Un exemple de DTD correspondant au document XML montré à la figure 1.1 (page 4).

```
<!ELEMENT cinémathèque (film*)>
<!ELEMENT film (titre, réalisateur, année?, durée?, affiche?, site?,
acteurs?)>
<!ATTLIST film id CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT réalisateur (#PCDATA)>
<!ELEMENT année (#PCDATA)>
<!ELEMENT durée (#PCDATA)>
<!ATTLIST durée unité (min|h) #REQUIRED>
<!ELEMENT affiche (#PCDATA)>
<!ELEMENT site (#PCDATA)>
<!ELEMENT acteurs (acteur+)>
<!ELEMENT acteur (#PCDATA)>
```

FIG. 16 – Exemple de DTD

XLink

XLink

XLink [161] permet les interactions entre documents XML. Il rend possible les liens simples, mais aussi multiples (plusieurs cibles, internes, externes, etc.), ce qui le distingue des liens HTML, et les accompagne éventuellement d'annotations. Il est également bi-directionnel, c'est-à-dire qu'il garde la trace du document source lorsqu'un lien est déclenché.

Enfin, n'importe quel élément peut devenir un lien et, grâce à XPointer, on peut indexer des positions arbitraires d'un document.

Un exemple de lien étendu est donné à la figure 17.

```
<element         xmlns:xlink="http://www.w3.org/1999/xlink/namespace/"
xlink:type="extended">
     <xlink:locator href="Source" role="role1"></xlink:locator>
     <xlink:locator href="Target" role="role2"></xlink:locator>
     <xlink:arc               from="role1" to="role2" show="embed" ac-
     tuate="auto"></xlink:arc>
     <xlink:title>The link title</xlink:title>
     <xlink:title xml:lang="fr">Description du lien</xlink:title>
</element>
```

FIG. 17 – Exemple de lien XLink.

Glossaire

XML

ancêtre : Un élément ou nœud XML est ancêtre d'un autre s'il contient celui-ci, quel que soit le nombre d'éléments ou nœuds situés entre eux.

descendant : Un élément ou nœud XML est descendant d'un autre s'il est contenu par celui-ci, quel que soit le nombre d'éléments ou nœuds situés entre eux.

document (vue XML orientée-) : approche des documents XML qui considère le document comme un texte traditionnel comportant un marquage structurel. Cette vue s'oppose à la vue orientée données*. Voir la section 1.2.

document plat : Document ne comportant aucune marque formelle (balisage) indiquant la structure ou la présentation du texte. Voir la section 1.2

données (vue XML orientée-) : Approche des documents XML qui considère le document comme une structure contenant des données, ayant une fonction comparable à celle d'une base de données. Cette vue s'oppose à la vue orientée document*. Voir la section 1.2.

doxel : Elément XML.

enfant : Un élément ou nœud XML est l'enfant d'un autre s'il est contenu par celui-ci de façon directe, sans élément ou nœud situé entre eux.

feuille : Dans un document XML, un nœud (ou élément) *feuille* est un nœud qui se situe au bout de l'arbre, qui ne possède pas d'enfant non textuel.

granularité : La notion de granularité définit la taille du plus petit élément, au-delà duquel l'information n'est plus découpée par du balisage.

nœud : Element XML, dans la représentation arborescente.

parent : Un élément ou nœud XML est l'enfant d'un autre s'il contient celui-ci de façon directe, sans élément ou nœud situé entre eux.

racine : Dans un document XML, un nœud (ou élément) *racine* est le nœud ancêtre de tous les autres, celui qui contient l'ensemble du document.

Recherche d'information

cible : Dans une requête structurée, la cible est la partie de la requête qui spécifie l'élément qui doit être retourné à l'utilisateur (par opposition au support*). Voir la section 2.4.2.2.

exhaustivité : En recherche d'information semi-structurée, l'exhaustivité décrit à quel point l'élément traite du sujet de la requête (voir section 2.5.4.3).

langage naturel (requête) : Se dit d'une requête exprimée par une phrase complète ou un groupe de mots en langage naturel, par opposition à une requête dans un langage formel.

lemmatisation : Opération consistant à extraire la forme canonique d'un mot (son *lemme*), ainsi qu'éventuellement d'autres informations morphologiques. Exemple : *"voitures"* est au féminin pluriel, et a pour lemme *voiture*.

poids : Importance d'un mot dans un énoncé ou dans un document. Voir la section 2.3.1.

précision : taux de documents pertinents parmi tous les documents retrouvés par le système (voir page 31).

rappel : taux de documents pertinents retrouvés par le système parmi l'ensemble des documents pertinents de la collection (voir page 31).

spécificité : En recherche d'information semi-structurée, la spécificité décrit à quel point l'élément se focalise sur le sujet de la requête (et pas sur d'autres sujets – voir section 2.5.4.3).

support : Dans une requête structurée, le support est la partie de la requête qui concerne les éléments qui ne doivent pas être retournés à l'utilisateur, ceux qui ne servent que d'aide au système pour satisfaire le besoin exprimé (par opposition à la cible*). Voir la section 2.4.2.2.

Bibliographie

[1] S. Abiteboul. Querying Semi-Structured Data. In *Proceedings of the International Conference on Database Theory (ICDT)*, pages 1–18, Delphi, Greece, 1997. 6, 8, 23

[2] S. Aït-Mokhtar, V. Lux, and E. Banik. Linguistic Parsing of Lists in Structured Documents. In *Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML, NLPXML-2003)*, Budapest, Hungary, Apr. 2003. 9

[3] A. Albano, D. Colazzo, G. Ghelli, P. Manghi, and C. Sartiani. A Type System for Querying XML Documents. In Baeza-Yates et al. [6]. 23

[4] V. N. Anh and A. Moffat. Compression and an IR Approach to XML Retrieval. In Fuhr et al. [40], pages 99–104. 19, 27

[5] R. Baeza-Yates, N. Fuhr, and Y. S. Maarek, editors. *ACM SIGIR 2002 Workshop On XML and Information Retrieval*, Tampere, Finland, Aug. 2002. ACM Press, New York City, NY, USA. 27, 50

[6] R. Baeza-Yates, N. Fuhr, R. Sacks-Davis, and R. Wilkinson, editors. *ACM SIGIR 2000 Workshop On XML and Information Retrieval*, Athens, Greece, July 2000. ACM Press, New York City, NY, USA. 23, 49, 51, 52, 54

[7] R. Baeza-Yates and G. Navarro. Integrating contents and structure in text retrieval. *ACM SIGMOD Record*, 25(1):67–79, Mar. 1996. 15

[8] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York City, NY, USA and Addison Wesley Longman Publishing Co. Inc., 1999. 12, 14, 19, 22, 32

[9] R. K. Belew. *Finding Out About, A Cognitive Perspective on Search Engine Technology and the WWW*. Cambridge University Press, 2000. 12, 30

[10] R. K. Belew and J. Hatton. RAVE Reviews: Acquiring Relevance Assessment from Multiple Users. In *Machine Learning in Information Access. AAAI Spring Symposia*, 1996. 12

[11] N. J. Belkin, R. N. Oddy, and H. M. Brooks. ASK for Information Retrieval: Part I. Background and Theory. *Journal of Documentation*, 38(2):61–71, 1982. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.299-304). 11

[12] A. Bonifati and S. Ceri. Comparative Analysis of Five XML Query Languages. *SIGMOD Record*, 29(1):68–79, 2000. 15, 23

[13] D. Braga, A. Campi, E. Damiani, P. Lanzi, and G. Pasi. FXPath: Flexible Querying of XML Documents. In *Proceedings of the EUROFUSE Workshop on Information Systems*, 2002. 27

[14] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proceedings of the 1996 ACM*

SIGMOD international conference on Management of data, pages 505–516, Montreal, Quebec, Canada, 1996. ACM Press, New York City, NY, USA. 23

[15] D. Carmel, N. Efraty, G. M. Landau, Y. S. Meerek, and Y. Mass. An extension of the Vector Space Model for querying XML documents via XML fragments. In Baeza-Yates et al. [5], pages 151–158. 24

[16] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: A Graphical Language for Querying and Reshaping XML Documents. In QL'98 [111]. 23

[17] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: a Graphical Language for Querying and Restructuring XML Documents. In *Proceedings of the 8th International WWW Conference, WWW8*, Toronto, Canada, May 1999. International World Wide Web Conference Committee (IW3C2). 24

[18] F. Chahuneau. XML. In *Documents Numériques – Gestion de Contenu*. Techniques de l'Ingénieur, 2001. (H7-148). 6

[19] D. Chamberlin, J. Robie, and D. Florescu. Quilt: An XML Query Language for Heterogeneous Data Sources. In *Proceedings of WebDB 2000 Conference*, Lecture Notes in Computer Science. Springer-Verlag, New York City, NY, USA, 2000. 23

[20] Y. Chiaramella, P. Mulhem, and F. Fourel. A model for multimedia information retrieval. Technical Report Basic Research Action FERMI 8134, CLIPS-IMAG, 1996. 14

[21] T. T. Chinenyanga and N. Kushmerick. An expressive and efficient language for XML information retrieval. *Journal of American Society for Information Science and Technology (JASIST), Special Topic Issue on XML and Information Retrieval*, 53(6):438–453, 2002. 19

[22] C. L. A. Clarke, G. V. Cormarck, and F. J. Burkowski. An Algebra for Structured Text Search and A Framework for its Implementation. *The Computer Journal*, 38(1):43–56, 1994. 20, 23

[23] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In J. C. Freytag, P. C. Lockemann, S. Abiteboul, M. J. Carey, P. G. Selinger, and A. Heuer, editors, *Proceedings of 29th International Conference on Very Large Data Bases*, Berlin, Germany, Sept. 2003. Morgan Kaufmann. 19, 20

[24] D. Colazzo, C. Sartiani, A. Albano, G. Ghelli, P. Manghi, L. Lini, and M. Paoli. A Typed Text Retrieval Query Language for XML Documents. *Journal of American Society for Information Science and Technology (JASIST), Special Topic Issue on XML and Information Retrieval*, 53(6):467–488, Apr. 2002. 7

[25] J. H. Coombs, A. H. Renear, and S. J. DeRose. Markup Systems and the Future of Scholarly Text Processing. *Communications of the ACM*, 30(11):933–947, 1987. 5

[26] W. Cooper. A Definition of Relevance for Information Retrieval. *Information Storage and Retrieval*, 7(1):19–37, 1971. 30, 32

[27] *Actes de la 1ère Conférence en Recherche d'Information et Applications, CORIA'04*, Toulouse, France, Mar. 2004. IRIT, Toulouse. 55, 56

[28] *Actes de la 2ème conférence en Recherche d'Information et Applications (CORIA'05)*, Grenoble, France, Mar. 2005. 56, 57

[29] H. Cui, J.-R. Wen, and T.-S. Chua. Hierarchical Indexing and Flexible Element Retrieval for Structured Document. In F. Sebastiani, editor, *Advances in Information Retrieval, Proceedings of the 25th European Conference on IR Research, ECIR 2003*, volume 2633 of *Lecture Notes in Computer Science*, pages 73–87, Pisa, Italy, Apr. 2003. Springer-Verlag, New York City, NY, USA. 19, 27

[30] H. Cunningham. Information Extraction, Automatic. In *Encyclopedia of Language and Linguistics*. Elsevier, second edition, 2005. 14

[31] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: A Query Language for XML. In QL'98 [111]. 23, 24

[32] P. Dopichaj. The University of Kaiserslautern at INEX 2005. In Fuhr et al. [42], page ? ? 27

[33] S. Douglas, M. Hurst, and D. Quinn. Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 535–546, Las Vegas, Nevada, USA, 1995. 9

[34] XML Query Languages: Experiences and Exemplars, 1999. http://www.w3.org/1999/09/ql/docs/xquery.html. 23

[35] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the World-Wide Web: a survey. *ACM SIGMOD Record*, 27(3):59–74, 1998. 23

[36] I. O. for Standardisation (ISO). Information Technology – Database Language SQL. Standard No. ISO/IEC 9075:1999. 12

[37] C. Fox. A stop list for general text. *ACM SIGIR Forum*, 24(1-2):19–21, 1989. 17

[38] N. Fuhr and K. Großjohann. XIRQL – An Extension of XQL for Information Retrieval. In Baeza-Yates et al. [6]. 23

[39] N. Fuhr and K. Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In W. Croft, D. Harper, D. Kraft, and J. Zobel, editors, *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180, New York City, NY, USA, Sept. 2001. ACM Press, New York City, NY, USA. 6, 27

[40] N. Fuhr, N. Gövert, G. Kazai, and M.Lalmas, editors. *The First Workshop of the Initiative for the Evaluation of XML retrieval (INEX)*, Schloss Dagstuhl, International Conference And Research Center For Computer Science, Germany, Dec. 2002. 27, 49, 52, 59

[41] N. Fuhr, M. Lalmas, and S. Malik, editors. *Proceedings of the second Workshop of the Initiative for the Evaluation of XML retrieval (INEX), December 15–17, 2003*, Schloss Dagstuhl, Germany, 2004. 27, 51, 52, 53, 54, 55, 56, 57

[42] N. Fuhr, M. Lalmas, S. Malik, and G. Kazai, editors. *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005)*, volume 3493 of *Lecture Notes in Computer Science*, Schloss Dagstuhl, Germany, November 28-30, 2005, 2006. Springer-Verlag, New York City, NY, USA. 27, 51, 52, 54, 55, 57, 58

[43] N. Fuhr, M. Lalmas, S. Malik, and Z. Szlàvik, editors. *Advances in XML Information Retrieval. Third Workshop of the Initiative for the Evaluation of XML retrieval (INEX)*, volume 3493 of *Lecture Notes in Computer Science*, Schloss Dagstuhl, Germany, December 6-8, 2004, 2005. Springer-Verlag, New York City, NY, USA. 27, 52, 54, 58, 59

[44] N. Fuhr, S. Malik, and M. Lalmas. Overview of the Initiative for the Evaluation of XML retrieval (INEX) 2003. In Fuhr et al. [41], pages 1–11. 37

[45] M. Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkinson. Structured answers for a large structured document collection. In Korfhage et al. [77], pages 204–213. 15, 27

[46] E. Gaussier, C. Jacquemin, and P. Zweigenbaum. Traitement automatique des langues et recherche d'information. In E. Gaussier and M.-H. Stéphanini, editors,

Assistance intelligente à la recherche d'informations. Hermes Sciences Publications, Lavoisier, Paris, 2003. 14

[47] S. Geva. GPX - Gardens Point XML Information Retrieval at INEX 2004. In Fuhr et al. [43], pages 211–223. 28, 29

[48] S. Geva. GPX - Gardens Point XML IR at INEX 2005. In Fuhr et al. [42], page ? ? 27, 28, 29

[49] S. Geva and M. Leo-Spork. XPath Inverted File for Information Retrieval. In Fuhr et al. [41], pages 110–117. 28

[50] C. F. Goldfarb. Introduction to Generalized Markup. In *Proceedings of the ACM SIGPLAN-SIGOA Symposium on Text Manipulation*, pages 68–73, New York City, NY, USA, 1981. ACM Press, New York City, NY, USA. 7

[51] C. F. Goldfarb. *The SGML Handbook*. Oxford University Press, 1991. 7

[52] R. Goldman, J. McHugh, and J. Widom. From semistructured data to XML: Migrating the Lore data model and query language. In *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99)*, Philadelphia, Pennsylvania, USA, June 1999. 23

[53] T. Grabs and H.-J. Schek. ETH Zürich at INEX: Flexible Information Retrieval from XML with PowerDB-XML. In Fuhr et al. [40]. 19, 27

[54] L. Gravano. Characterizing Web Resources for Improved Search. In *Proceedings of the First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries*, Zurich, Switzerland, Dec. 2000. 12

[55] T. Grust. Accelerating XPath location steps. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 109–120, Madison, Wisconsin, USA, June 2002. ACM Press, New York City, NY, USA. 21

[56] M. Géry. *Indexation et interrogation de chemins de lecture en contexte pour la Recherche d'Information Structurée sur le Web*. PhD thesis, Université Joseph Fourier, Grenoble, France, Oct. 2002. 12

[57] N. Gövert and G. Kazai. Overview of the Initiative for the Evaluation of XML retrieval (INEX) 2002. In Fuhr et al. [40], pages 1–17. 37

[58] E. R. Harold and W. S. Means. *XML in a Nutshell*. O'Reilly, second edition, 2002. 3

[59] M. Hassler and A. Bouchachia. Searching XML Documents – Preliminary Work. In Fuhr et al. [42], page ? ? 20

[60] Y. Hayashi, J. Tomita, and G. Kikui. Searching text-rich XML documents with Relevance ranking. In Baeza-Yates et al. [6]. 21

[61] M. Hearst. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64, Mar. 1997. 14

[62] D. Hiemstra and V. Mihajlović. The simplest evaluation measures for XML information retrieval that could possibly work. In A. Trotman, M. Lalmas, and N. Fuhr, editors, *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 43–46, Glasgow, UK, Aug. 2005. 39

[63] G. Hubert. XML retrieval based on direct contribution of query component. In Fuhr et al. [42], page ? ? 27

[64] J. Hutchins. The concept of 'aboutness' in subject indexing. In *Aslib Proceedings*, volume 30, pages 172–181, May 1978. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.93-97). 17

[65] N. Ide and J. Veronis, editors. *The Text Encoding Initiative: Background and Context*. Kluwer Academic Publisher, 1995. 4, 7

[66] A. Imafouo and X. Tannier. Retrieval Status Values in Information Retrieval Evaluation. In M. Consens and G. Navarro, editors, *String Processing and Information Retrieval: 12th International Conference, SPIRE 2005*, volume 3772 of *Lecture Notes in Computer Science*, pages 222–227, Buenos Aires, Argentina, Nov. 2005. Springer-Verlag, New York City, NY, USA. 30

[67] K. S. Jones. A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation*, 28(1):11–20, 1972. 19

[68] K. S. Jones and P. Willett, editors. *Readings in Information Retrieval*. Morgan Kaufmann Publishers, 1997. 18, 19

[69] Y. Kagolovsky and J. Moehr. Current Status of the Evaluation in Information Retrieval. *Journal of Medical Systems*, 27(5):409–424, Oct. 2003. 30

[70] G. Kazai. Report of the INEX 2003 Metrics working group. In Fuhr et al. [41], pages 184–190. 39

[71] G. Kazai and M. Lalmas. INEX 2005 Evaluation Metrics, 2005. http://inex.is.informatik.uni-duisburg.de/2005/inex-2005-metricsv4.pdf. 39

[72] G. Kazai, M. Lalmas, and A. P. de Vries. The overlap problem in content-oriented XML retrieval evaluation. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 72–79, Sheffield, United Kingdom, 2004. ACM Press, New York City, NY, USA. 33, 39, 40

[73] G. Kazai, M. Lalmas, and B. Piwowarski. INEX'03 Relevance Assessment Guide. In Fuhr et al. [41], pages 203–209. 37

[74] G. Kazai, M. Lalmas, and T. Roelleke. Focussed Structured Document Retrieval. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE 2002)*, Lisbon, Portugal, Sept. 2002. 19

[75] J. Kekäläinen and K. Järvelin. Evaluating Information Retrieval Systems under the Challenges of Interaction and Multi-Dimensional Dynamic Relevance. In H. Bruce, R. Fidel, P. Ingwersen, and P. Vakkari, editors, *Proceedings of the CoLIS 4 Conference*, pages 253–270, Seattle, Washington, USA, July 2002. 15

[76] A. Kent, M. Berry, F. Luehrs, and J. Perry. Machine literature searching: VIII. Operational criteria for designing information retrieval systems. *American Documentation*, 6(2):83–101, 1955. 31

[77] R. Korfhage, E. M. Rasmussen, and P. Willett, editors. *16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, Pennsylvania, USA, June 1993. ACM Press, New York City, NY, USA. 51, 56

[78] R. R. Korfhage. *Information Storage and Retrieval*. John Wiley & Sons, Inc., 1997. 11, 12, 22, 32

[79] E. Kotsakis. Structured information retrieval in XML documents. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 663–667, Madrid, Spain, 2002. ACM Press, New York City, NY, USA. 19

[80] M. Lalmas. Dempster-Shafer's theory of evidence applied to structured documents: modelling uncertainty. In *Proceedings of the 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 110–118, Philadelphia, Pennsylvania, USA, Aug. 1997. ACM Press, New York City, NY, USA. 14, 27

[81] M. Lalmas and B. Piwowarski. INEX 2005 Relevance Assessment Guide, 2005. http://inex.is.informatik.uni-duisburg.de/2005/internal/pdf/Relevance_Assessment2005.pdf. 37, 38

[82] L. Lamport. *Latex*. Addison-Wesley, 1994. 8

[83] B. Larsen, S. Malik, and A. Tombros. The Interactive Track at INEX 2005. In Fuhr et al. [42]. 15, 17

[84] Y. K. Lee, S.-J. Yoo, K. Yoon, and P. B. Berra. Index structures for structured documents. In *Proceedings of the first ACM international conference on Digital libraries*, pages 91–99, Bethesda, Maryland, USA, 1996. ACM Press, New York City, NY, USA. 20

[85] L. Lini, D. Lombardini, M. Paoli, D. Colazzo, and C. Sartiani. XTReSy: A Text Retrieval System for XML documents. In D. Buzzetti, H. Short, and G. Pancalddella, editors, *Augmenting Comprehension: Digital Tools for the History of Ideas*. Office for Humanities Communication Publications, King's College, London, 2001. 7, 8

[86] S. Liu, Q. Zou, and W. W. Chu. Configurable indexing and ranking for XML information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 88–95, Sheffield, United Kingdom, 2004. ACM Press, New York City, NY, USA. 19

[87] H. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):307–317, 1957. 18

[88] R. Luk, A. Chan, T. Dillon, and H. Leong. A survey of Search Engines for XML Documents. In Baeza-Yates et al. [6]. 23

[89] R. W. Luk, H. Leong, T. S. Dillon, A. T. Chan, W. B. Croft, and J. Allan. A survey in indexing and searching XML documents. *Journal of American Society for Information Science and Technology (JASIST)*, 53(6):415–437, 2002. 19, 20

[90] S. Malik, G. Kazai, M. Lalmas, and N. Fuhr. Overview of INEX 2005. In Fuhr et al. [42]. 33

[91] S. Malik, M. Lalmas, and N. Fuhr. Overview of INEX 2004. In Fuhr et al. [43], pages 1–15. 37

[92] M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, 7(3):216–244, July 1960. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.39-46). 22

[93] Y. Mass and M. Mandelbrod. Retrieving the most relevant XML Components. In Fuhr et al. [41], pages 53–58. 20, 27

[94] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A Database Management System for Semistructured Data. *SIGMOD Record*, 26(3):54–66, Sept. 1997. 23

[95] C. T. Meadow, B. R. Boyce, and D. H. Kraft. *Text Information Retrieval Systems*. Academic Press, New York City, NY, USA, San Diego, second edition, 2000. 12, 29

[96] G. Mecca, P. Merialdo, and P. Atzeni. Do we really need a new query language for XML ? In QL'98 [111]. 23

[97] J. Minel. Le résumé automatique de textes : solutions et perspectives. *Traitement automatique des langues*, 45(1), 2004. 14

[98] S. Mizzaro. Relevance, the whole (hi)story. *Journal of American Society for Information Science*, 48(9):810–832, 1997. 30

[99] S. Mizzaro. How many relevances in information retrieval ? *Interacting with Computers*, 10(3):303–320, 1998. 30

[100] A. Moffat, R. Sacks-Davis, and R. Wilkinson. Retrieval of Partial Documents. In D. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC)*, pages 181–190, Gaithersburg, MD, 1994. National Institute of Standards and Technology, Department of Commerce, National Institute of Standards and Technology. NIST Special Publication 500-215. 14

[101] F. Moreau and P. Sébillot. Contributions des techniques du traitement automatique des langues à la recherche d'information. Technical Report 1690, IRISA, France, Feb. 2005. 19

[102] G. Navarro. A Language for Queries on Structure and Contents of Textual Databases. Master's thesis, Dept. of Computer Science, University of Chile, Apr. 1995. 23

[103] R. A. O'Keefe and A. Trotman. The Simplest Query Language That Could Possibly Work. In Fuhr et al. [41], pages 167–174. 26

[104] C. Peters. What happened in CLEF 2005. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria, Sept. 2005. 33

[105] W. Piez. Beyond the "descriptive vs. procedural" distinction. *Markup Languages: Theory and Practice*, 3(2):141–172, Dec. 2001. 7

[106] B. Piwowarski. *Techniques d'apprentissage pour le traitement d'informations structurées : application à la recherche d'information.* PhD thesis, Université Pierre et Marie Curie, Paris, France, July 2003. 12, 14

[107] B. Piwowarski and P. Gallinari. Expected Ratio of Relevant Units: A Measure for Structured Information Retrieval. In Fuhr et al. [41], pages 158–166. 14

[108] B. Piwowarski and M. Lalmas. Interface pour l'évaluation de systèmes de recherche sur des documents XML. In CORIA 2004 [27], pages 109–120. 37

[109] E. Popovici, G. Ménier, and P.-F. Marteau. SIRIUS: A Lightweight XML Indexing and Approximate Search System at INEX 2005. In Fuhr et al. [42], page ? ? 27

[110] R. T. Pédauque. Document : forme, signe et médium, les re-formulations du numérique. Working Paper, July 2003. http://archivesic.ccsd.cnrs.fr/documents/archives0/00/00/05/11/index_fr.html. 14

[111] *Proceedings of the Query Languages Workshop (QL'98)*, 1998. 23, 50, 51, 54, 56

[112] V. V. Raghavan, P. Bollmann, and G. S. Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3):205–229, July 1989. 32

[113] A. M. Rees and D. G. Schulz. A field experimental approach to the study of relevance assessments in relation to document searching. Technical Report NSF Contract No. C-423, Center for Documentation and Communication Research, School of Library Science, Case Western Reserve University, Cleveland, Ohio, USA, 1967. 30

[114] A. Renear. The descriptive/procedural distinction is flawed. *Markup Languages: Theory and Practice*, 2(4):411–420, 2000. 7

[115] A. Renear, D. Dubin, C. M. Sperberg-McQueen, and C. Huitfeldt. Towards a Semantics for XML Markup. In *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 119–126, McLean, Virginia, USA, 2002. ACM Press, New York City, NY, USA. 7

[116] S. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.281-286). 22

[117] J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL). In QL'98 [111]. 23

[118] C. Roussey. *Une méthode d'indexation sémantique adaptée aux corpus multilingues*. PhD thesis, Institut National des Sciences Appliquées (INSA) de Lyon, Dec. 2001. 17

[119] C. Roussey, S. Calabretto, and J.-M. Pinon. Etat de l'art en indexation et recherche d'information. *Document Numérique, numéro spécial Gestion des documents et gestion des connaissances*, 3(3-4):121–150, Dec. 1999. 18

[120] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel. Database systems for structured documents. In *Proceedings of the International Symposium on Advanced Database Technologies and Their Integration (ADTI)*, Nara, Japan, 1994. 15, 19

[121] A. Salminen and F. W. Tompa. PAT expressions: an algebra for text search. In F. Kiefer, G. Kiss, and J. Pajzs, editors, *Proceedings of the 2nd International Conference on Computational Lexicography, COMPLEX '92*, pages 309–332. Budapest: Linguistics Institute, Hungarian Academy of Sciences, 1992. 20

[122] G. Salton. A Comparison Between Manual and Automatic Indexing Methods. *Journal of American Documentation*, 20(1):61–71, 1969. 17

[123] G. Salton. *The SMART retrieval system ; experiments in automatic document processing*. Prentice-Hall, 1971. 22

[124] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In Korfhage et al. [77], pages 49–58. 14

[125] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.323-328). 18

[126] G. Salton, A. A. Fow, and H. Wu. Extended Boolean Information Retrieval. *Communications of the ACM*, 26(11):1022–1036, Dec. 1983. 22

[127] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, 1983. 12, 19

[128] T. Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of American Society for Information Science*, 26(6):321–434, 1975. 30

[129] T. Saracevic. Evaluation of Evaluation in Information Retrieval. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 138–145, Seattle, Washington, USA, July 1995. ACM Press, New York City, NY, USA. 30

[130] K. Sauvagnat. XFIRM : Un Modèle Flexible de Recherche d'Information pour le stockage et l'interrogation de documents XML. In CORIA 2004 [27], pages 121–142. 24

[131] K. Sauvagnat. *Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés*. PhD thesis, Université Paul Sabatier, Toulouse, France, June 2005. 12, 15

[132] K. Sauvagnat and M. Boughanem. A la recherche de nœuds informatifs dans des corpus de documents XML: où pourquoi on a toujours besoin de plus petit que soi... In CORIA 2005 [28], pages 119–134. 15

[133] K. Sauvagnat, G. Hubert, M. Boughanem, and J. Mothe. IRIT at INEX 2003. In Fuhr et al. [41], pages 142–148. 19

[134] J. Savoy. A stemming procedure and stopword list for general French corpora. *Journal of American Society for Information Science*, 50(10):944–952, 1999. 17

[135] J. Savoy. Indexation manuelle et automatique : une évaluation comparative basée sur un corpus en langue française. In CORIA 2005 [28], pages 9–23. 17

[136] T. Schlieder and H. Meuss. Querying and Ranking XML Documents. *Journal of American Society for Information Science and Tecnology (JASIST), Special Topic Issue on XML and Information Retrieval*, 53(6):489–503, Apr. 2002. 19, 21, 24, 27

[137] D. Shin, H. Jang, and H. Jin. BUS: an effective indexing and retrieval scheme in structured documents. In *Proceedings of the third ACM conference on Digital libraries*, pages 235–243, Pittsburgh, Pennsylvania, USA, 1998. ACM Press, New York City, NY, USA. 20

[138] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. An Element-based Approach to XML Retrieval. In Fuhr et al. [41], pages 19–26. 19, 27

[139] B. Sigurbjörnsson, J. Kamps, and M. de Rijke. University of Amsterdam at INEX 2005: AdHoc Track. In Fuhr et al. [42], page ? ? 20

[140] B. Sigurbjörnsson, A. Trotman, S. Geva, M. Lalmas, B. Larsen, and S. Malik. INEX 2005 Guidelines for Topic Development, 2005. http://inex.is.informatik.uni-duisburg.de/2005/internal/pdf/TD05.pdf. 34

[141] I. Song and P. S. Bayerl. Semantics of XML documents. Internal Working Paper, Apr. 2003. 7

[142] K. Sparck Jones and C. van Rijsbergen. Report on the need for and provision of an "ideal" information retrieval test collection. Technical report, Computer Laboratory, University of Cambridge, 1975. British Library Research and Development Report 5266. 31

[143] C. M. Sperberg-Mcqueen, C. Huitfeldt, and A. Renear. Meaning and interpretation of markup. *Markup Languages: Theory and Practice*, 2(3):215–234, Aug. 2000. 7

[144] D. R. Swanson. Historical Note: Information Retrieval and the Future of an Illusion. *Journal of the American Society for Information Science*, 39(2):92–98, 1988. Repr. in: Sparck Jones, Karen and Willett, Peter (eds.) Readings in information retrieval (1997, p.555-561). 31

[145] J. Swets. Effectiveness of information retrieval methods. *American Documentation*, 20(1):72–89, 1969. 31

[146] X. Tannier. Traitement automatique du langage naturel pour l'extraction et la recherche d'information. Technical report, Ecole Nationale Supérieure des Mines de Saint-Etienne, July 2006. 16, 17

[147] X. Tannier, J.-J. Girardot, and M. Mathieu. Classifying XML Tags through "Reading Contexts". In P. R. King, editor, *Proceedings of the 2005 ACM Symposium on Document Engineering*, pages 143–145, Bristol, United Kingdom, Nov. 2005. ACM Press, New York City, NY, USA. 6

[148] D. Tarr and H. Borko. Factors influencing inter-indexer consistency. In *Proceedings of the ASIS 37th Annual Meeting*, volume 11, pages 50–55, 1974. 17

[149] A. Theobald and G. Weikum. The Index-based XXL Search Engine for Querying XML Data with Relevance Ranking. In *Proceedings of the 8th International Conference on Extending Database Technology (EDBT)*, pages 477–495, Prague, Czech Republic, Mar. 2002. 21, 27

[150] B. Trippe. Do XML Editors Matter ?, Oct. 2001. http://www.transformmag.com/db_area/archs/2001/10/tfm0110xm.shtml. 5

[151] A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In Fuhr et al. [43], pages 16–40. 24, 26

[152] C. van Rijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979. 32

[153] R. van Zwol, G. Kazai, and M. Lalmas. The Multimedia Track at INEX 2005: Overview. In Fuhr et al. [42]. 17

[154] E. Voorhees. Overview of TREC 2004. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings of the Thirteenth Text REtrieval Conference (TREC)*, Nov. 2004. NIST Special Publication 500-261. 31, 33

[155] E. Voorhees and D. Harman. Overview of the Sixth Text REtrieval Conference (TREC-6). In *Proceedings of the Sixth Text REtrieval Conference (TREC)*, Nov. 1997. NIST Special Publication 500-240. 31

[156] E. Voorhees and D. M. Tice. The TREC-8 question answering track evaluation. In *Proceedings of the Eighth Text REtrieval Conference (TREC)*, Nov. 1999. NIST Special Publication 500-246. 12, 14

[157] World Wide Web Consortium (W3C). http://www.w3.org/. 3, 6, 23

[158] XML Path Language (XPath). World Wide Web Consortium (W3C) Recommandation, 1999. http://www.w3.org/TR/1999/REC-xpath-19991116. 4, 23, 24

[159] XSL Transformation (XSL). World Wide Web Consortium (W3C) Recommandation, 1999. http://www.w3.org/TR/1999/REC-xslt-19991116. 4, 23

[160] Semantic Web. World Wide Web Consortium (W3C)., 2001. http://www.w3.org/2001/sw/. 6

[161] XML Linking Language (XLink), 2001. http://www.w3.org/TR/xlink/. 44

[162] XML Schema (W3C), 2001. http://www.w3.org/XML/Schema/. 3, 6

[163] XHTML 1.0 The Extensible HyperText Markup Language, 2002. http://www.w3.org/TR/xhtml1/. 4

[164] Extensible Markup Language (XML). World Wide Web Consortium (W3C) Recommandation, 2004. http://www.w3.org/TR/2004/REC-xml-20040204/. 3

[165] XML Syntax for XQuery 1.0 (XQueryX), 2005. http://www.w3.org/TR/xqueryx/. 4

[166] XQuery 1.0: An XML Query Language. World Wide Web Consortium (W3C) Working Draft, 2005. http://www.w3.org/TR/2005/WD-xquery-20050404/. 4, 12, 23, 24, 25

[167] XQuery 1.0 and XPath 2.0 Full-Text Use Cases. World Wide Web Consortium (W3C) Working Draft, 2006. http://www.w3.org/TR/xmlquery-full-text-use-cases/. 23

[168] N. Walsh and L. Muellner. *DocBook: The Definitive Guide*. O'Reilly, first edition, Oct. 1999. 4

[169] P. Wilson. Situational Relevance. *Information Storage and Retrieval*, 9(8):457–471, 1973. 30

[170] J. E. Wolff, H. Florke, and A. B. Cremers. Searching and Browsing Collections of Structural Information. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries (ADL)*, pages 141–150, Washington, D.C., USA, May 2000. IEEE Computer Society. 19, 21

[171] A. Woodley and S. Geva. NLPX at INEX 2004. In Fuhr et al. [43], pages 382–394. 20

[172] S. Yoo. An XML Retrieval Model based on Structural Proximities. In Fuhr et al. [40], pages 125–132. 27

[173] J. Zhou. Phrasal Terms in Real-Words IR Applications. In T. Strzalkowski, editor, *Natural Language Information Retrieval*, pages 215–259. Kluwer Academic Publisher, Dordrecht, NL, 1999. 19

[174] G. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949. 18

[175] J. Zobel. How reliable are the results of large scale information retrieval experiments. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314, Melbourne, Australia, Aug. 1998. ACM Press, New York City, NY, USA. 31

Ecole Nationale Supérieure des Mines de Saint-Etienne
Centre G2I
158, Cours Fauriel
42023 SAINT-ETIENNE CEDEX 2

www.emse.fr