Complexité de Kolmogorov, une mise en perspective. Partie II : Classification, Traitement de l' Information et Dualité.

Marie Ferbus-Zanda

Complexité de Kolmogorov, une mise en perspective

Partie II : Classification, Traitement de l'Information et Dualité*

Marie Ferbus-Zanda
LIAFA, CNRS & Université Paris Diderot - Paris 7
Case 7014
75205 Paris Cedex 13 France
Marie.Ferbus@liafa.jussieu.fr

Abstract

We survey diverse approaches to the notion of information : from Shannon entropy to Kolmogorov complexity. Two of the main applications of Kolmogorov complexity are presented : randomness and classification. The survey is divided in two parts published in a same volume.
Part II is dedicated to the relation between logic and information system, within the scope of Kolmogorov algorithmic information theory. We present a recent application of Kolmogorov complexity : classification using compression, an idea with provocative implementation by authors such as Bennett, Vitányi and Cilibrasi. This stresses how Kolmogorov complexity, besides being a foundation to randomness, is also related to classification. Another approach to classification is also considered : the so-called "Google classification". It uses another original and attractive idea which is connected to the classification using compression and to Kolmogorov complexity from a conceptual point of view. We present and unify these different approaches to classification in terms of Bottom-Up versus Top-Down operational modes, of which we point the fundamental principles and the underlying duality. We look at the way these two dual modes are used in the design of information system, particularly the relational model for database introduced by Codd in the 70's. These operational modes are also reinterpreted in the context of the comprehension schema of axiomatic set theory ZF. This leads us to develop how Kolmogorov complexity is linked to intensionality, abstraction, classification and information system.

*Cet article est une traduction française de l'article : Ferbus-Zanda M. Kolmogorov Complexity in perspective. Partie II : Classification, Information Processing and Duality. A paraître *Synthese*, 2008-2010.

Keywords : Logic, Computer Science, Kolmogorov Complexity, Algorithmic Information Theory, Compression, Classification, Information System, Database, Bottom-Up versus Top-Down Approach, Intensionality, Abstraction.

Résumé

Nous exposons différentes approches du concept d'information : depuis la notion d'entropie de Shannon jusqu'à la théorie de la complexité de Kolmogorov. Nous présentons deux des principales applications de la complexité de Kolmogorov : l'aléatoirité et la classification. Cet exposé est divisé en deux parties publiées dans un même volume.
La partie II est consacrée à la relation entre la logique et les systèmes d'information, dans le cadre de la théorie algorithmique de l'information de Kolmogorov. Nous exposons une application récente de la complexité de Kolmogorov : la classification par compression, qui met en oeuvre une implémentation audacieuse de la complexité de Kolmogorov, par des auteurs comme Bennett, Vitányi et Cilibrasi. Cela permet, en outre, de dégager en quoi la complexité de Kolmogorov est liée à la classification, tout comme elle fonde l'aléatoirité. Nous présentons également une autre approche de la classification, la "Google classification". Celle-ci utilise une autre idée originale et particulièrement intéressante, et qui est connectée à la classification par compression et à la complexité de Kolmogorov d'un point de vue conceptuel. Nous présentons et unifions ces différentes approches de la classification en termes de modes opératoires Bottom-Up versus Top-Down dont nous indiquons les principes fondamentaux et la dualité sous-jacente. Nous étudions comment ces modes duals sont utilisés, dans l'appréhension des systèmes d'information, et tout particulièrement dans le modèle relationnel des bases de données introduit par Codd dans les années 70. Nous réinterprétons en outre ces modes opératoires dans le contexte du schéma de compréhension de la théorie axiomatique des ensembles ZF. Ceci nous amène à développer en quoi la complexité de Kolmogorov est liée à l'intentionnalité, à l'abstraction, à la classification et aux systèmes d'information.

Mots-Clefs : Logique, Informatique, Complexité de Kolmogorov, Théorie Algorithmique de l'Information, Compression, Classification, Systèmes d'Information, Bases de Données, Approche Bottom-Up versus Top-Down, Intentionnalité, Abstraction.

Table des matières

Note. Toutes les notations et définitions relatives à la complexité de Kolmogorov sont introduites dans la partie I [1].

1 Théorie algorithmique de l'information et classification

Des résultats étonnants ont été obtenus dans l'application de la théorie de la complexité de Andrei Nikolaevich Kolmogorov aux problèmes de classification d'objets, aussi divers que des textes d'écrivains, des pièces de musique, des copies d'étudiants à un examen (mal surveillé) et à un autre niveau : à la classification des langues naturelles, et à celle des espèces naturelles (phylogénie).

Les auteurs, principalement Charles Bennett, Paul Vitányi, Rudi Cilibrasi [2] ont élaboré des méthodes de plus en plus raffinées qui suivent les différentes étapes que nous détaillons ci-dessous.

1.1 Définition et représentation de la famille d'objets à classifier

Il s'agit dans un premier temps de définir une famille d'objets spécifiques pour lesquels on désire obtenir une classification.

Par exemple, un ensemble de textes d'écrivains russes pour lesquels on souhaite trouver un regroupement par auteurs. Dans ce cas simple, les textes considérés sont tous écrits dans leur langue d'origine : le russe.

Autre exemple : le domaine musical. Dans ce cas, il est nécessaire de pouvoir se référer à une traduction commune, i.e. à une *normalisation* des pièces de musique (représentant, ou si l'on veut, interprétant des partitions musicales) que l'on désire regrouper par compositeur. Cette représentation commune (qui doit de sucroît être traitable par un programme informatique) est nécessaire pour pouvoir comparer ces différents morceaux de musique. Citons Delahaye [14] :

> ≪ Partant de morceaux de musique codés dans le format MIDI (*Musical Instrumental Digital Interface*), les chercheurs ont constitué des fichiers normalisés de 36 morceaux de musique. La normalisation consiste pour chaque morceau à produire une version piano, qui elle-même détermine un fichier de données (une suite de nombres

1. Ferbus-Zanda M. & Grigorieff S. Kolmogorov Complexity in perspective. Part I : Information Theory and Randomness. To appear in *Synthese*, 2010.
On peut également consulter [22], [19], [13] et [31] ainsi que les travaux fondateurs de Andrei Nikolaevich Kolmogorov [28], Gegory Chaitin [3, 5] et Ray Solomonoff [33, 34].

2. Les exposés de Jean-Paul Delahaye [14, 15] (dont nous avons tiré grand profit) fournissent une première approche très éclairante sur ces travaux.

codés sur 8 chiffres binaires[3]). Sans cette normalisation, qui est une pure extraction d'information, rien ne fonctionnerait [...] ≫

Autre exemple d'un niveau différent : les 52 langues indo-européennes principales. Dans ce cas, on doit choisir un objet canonique (ici un texte), et ses représentations (ici des traductions) dans chacun des différents corpus considérés, comme par exemple la *Déclaration Universelle des Droits de l'Homme* et ses traductions dans les différentes langues prises en considération. Ce dernier exemple a d'ailleurs servi de test à la méthode de Vitányi. En ce qui concerne la classification des espèces naturelles (autre exemple développé par Vitányi), les éléments canoniques seront des séquences d'ADN.

Il s'agit donc de sélectionner, définir et normaliser une famille d'objets ou corpus que l'on désire classifier.

Le problème de la normalisation d'une famille d'objets est complexe et cela peut aussi être le cas de la définition d'une telle famille. *Grosso modo*, on peut diviser en différentes classes les types d'objets auxquels on a affaire :

- La famille d'objets à classifier est bien définie et la normalisation de ces objets (de leur représentation) peut se faire sans perte d'information. C'est le cas par exemple, des textes littéraires.
- La famille d'objets à classifier peut être finie, mais non connue voire de taille non bornée à l'avance comme c'est le cas avec les informations du World Wide Web (*Web*) (cf. la classification avec Google, section 3).
- Il y a des situations où une telle normalisation peut être difficile à trouver voire impossible, comme cela peut être le cas pour la peinture (les tableaux de peintres), pour le dessin, la photographie, le cinéma d'auteur, etc.

1.2 Comparaison du contenu commun en information

In fine, on se retrouve avec une famille de mots définis sur un même alphabet représentant les objets dont on désire comparer voire mesurer le contenu commun en information[4] (noter que l'on peut se ramener à un alphabet binaire).

Cette comparaison est réalisée en définissant une distance entre les paires de tels mots (binaires) avec l'intuition suivante :

> *Plus deux mots ont de contenu commun en information, plus ils sont proches, plus leur distance est petite et inversement, plus les deux mots ont un contenu commun en information pauvre, plus ils sont indépendants, non corrélés, plus leur distance est grande.*
> *Deux mots identiques ont une distance nulle et deux mots totalement indépendants (par exemple des mots représentant* 100 *lancer*

3. En fait sur un *byte* (ou encore un *octet*) qui est par définition une suite de 8 chiffres. On peut aussi voir un byte comme un nombre compris entre 0 et 255.

4. Le contenu en information d'un objet est détaillé dans la partie I. Pour Kolmogorov, c'est par définition, la comlexité algorithmique de l'objet.

de pièce) ont une distance proche de 1 *(pour une distance normalisée bornée par* 1*).*

Noter que les auteurs, dans leur approche de la classification d'information, suivent en cela une démarche analogue à celle de Claude Shannon et de Kolmogorov, qui a été initialement de définir une mesure *quantitative* du degré d'aléatoirité d'un mot, i.e. de son contenu en information. Exactement de la même façon que l'on a une mesure numérique pour les surfaces ou les volumes.

1.3 Classification

Il s'agit d'associer une classification aux objets ou corpus définis dans la section 1.1 sur la base des mesures numériques issues des distances définies dans la section 1.2. Cette étape est la moins formellement définie à l'heure actuelle. Les auteurs donnent des représentations sous-jacentes aux classifications obtenues sous forme de tableaux, d'arbres, de graphes, etc.

Il s'agit donc plus d'une *visualisation*, autrement dit, d'une *représentation graphique* de la classification obtenue que d'une *classification formelle*. Les auteurs ne disposent pas d'un puissant cadre de travail mathématique comme c'est le cas avec le *modèle relationnel des bases de données* élaboré par Edgar F. Codd dans les années 70 [10, 11] et de son extension (récente) au *modèle objet* avec les arbres. L'approche de Codd est actuellement une des seules approches mathématiques formelles (sinon la seule) de la notion de *structuration de l'information*. On peut ainsi dire que la structuration d'une classe d'informations ou (de représentations) d'objets (du monde réel comme l'appellent les informaticiens) est une base de données relationnelle qui est en elle même un objet mathématique formel parfaitement défini. On peut de plus interroger cette base de données et en extraire de "nouvelles" informations au moyen de *requêtes*, qui peuvent être écrites dans un langage formel (*l'algèbre relationnelle de Codd*). Noter d'ailleurs que cette approche théorique extrêmement originale est implémentée en machine depuis les années 80 et utilisée partout où il est question de bases de données.

La question qui se pose est par conséquent de savoir au juste comment on peut interpréter de façon plus formelle les tableaux et arbres obtenus dans la classification par compression et plus particulièrement comment extraire formellement des informations de cette classification.

La classification par compression ainsi obtenue, classification rudimentaire mais néanmoins précieuse, est donc *non formelle* et en ce sens analogue par exemple, à la classification des mots dans un dictionnaire des synonymes. On se retrouve face à un problème complexe sur lequel nous revenons dans la section 4. Néanmoins, Vitányi & coll. ont obtenu par cette méthode une classification des 52 langues indo-européennes, sous forme d'arbre, qui est celle mise en évidence par les linguistes, ce qui est un succès remarquable. Ils ont aussi obtenu des arbres phylogénétiques de classification des espèces naturelles conformes à ceux

obtenus par les paléontologues. Ces arbres, représentant les liens de parenté des espèces naturelles, sont établis au moyen de comparaison de séquences d'ADN.

1.4 Normalisation

Reste un problème important concernant l'utilisation d'une distance comme définie section 1.3. Pour obtenir une classification, il faut prendre en compte la quantité d'information contenue dans les objets considérés.

Citons Cilibrasi [7] :

> ≪ De grand objets (dans le sens, mots d'une grande longueur) qui se différencient par une partie minime sont intuitivement plus proches que de petits objets qui se différencient de cette même partie minime. Par exemple deux génômes entiers de l'ADN mitochondrial comportant 18000 bases qui diffèrent de 9000 bases sont très différents alors que les deux génômes entiers du noyau comportant 3×10^9 bases qui diffèrent seulement de 9000 bases sont très proches ≫

Comme on le verra, ce problème est relativement facile à résoudre, par une normalisation des distances obtenues. Noter qu'il s'agit là d'une autre forme de normalisation que celle décrite dans la section 1.1.

1.5 Compression

Finalement, remarquons que toutes ces méthodes reposent sur la complexité de Kolmogorov qui est, comme on le sait, une fonction non calculable (voir par exemple [22]).

L'idée remarquable introduite par Vitányi est la suivante :

- *Considérer la complexité de Kolmogorov d'un objet comme une forme idéale, ultime, optimale de la compression de la représentation de l'objet en question.*
- *Calculer des approximations de cette compression idéale au moyen d'algorithmes de compression effectifs et implémentés en machine comme gzip, bzip2, PPM, ...*

Notons que la qualité et l'efficacité de tels compresseurs est largement dûe à l'utilisation des outils statistiques. Par exemple *PPM* (*Prediction by Partial Matching*), est un puissant mélange de modèles statistiques [5] combinés aux arbres, aux arbres suffixes et aux tableaux suffixes.

L'efficacité remarquable de ces outils est due à des décennies de recherche en compression de données. Dans une évolution future, ces résultats ne pourront qu'aller en s'améliorant, et par conséquent, meilleures seront les approximations de la complexité de Kolmogorov. En remplaçant la complexité de Kolmogorov "pure" mais non calculable par un banal algorithme de compression comme gzip, Vitányi a franchit une étape audacieuse.

5. Nous reviendrons (section 4) sur le traitement de l'information par les statistiques.

2 Classification par compression

2.1 La distance d'information normalisée *(NID)*

Nous allons maintenant définir plus formellement ces notions. L'idée de base est de mesurer le contenu en information commun à deux mots binaires représentant des objets d'une famille pour laquelle on désire une classification.

Une première tentative de cette nature remonte aux années 90 [2] : Bennett et coll. définissent une notion de *distance informationnelle* entre deux mots x, y comme étant la taille du plus court programme qui transforme le mot x en y et le mot y en x. Ces considérations reposent sur la notion de *calcul réversible*. Une définition formelle possible pour une telle distance est :

$$ID'(x,y) = \text{le plus petit } |p| \text{ tel que } U(p,x) = y \text{ et } U(p,y) = x$$

où $U : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ est optimal pour la complexité conditionnelle $K(\ |\ )$ (cf. Partie I).
Nous allons principalement travailler avec la définition alternative suivante :

$$ID(x,y) = \max\{K(x|y), K(y|x)\}$$

L'intuition sous-jacente à ces définitions est que le plus court programme qui calcule x à partir de y et y à partir de x prend en considération les similarités entre x et y.

Remarquons que les deux définitions ne coincident pas (même à un terme logarithmique près) mais conduisent toutes deux à des développements similaires et à des applications efficaces.

Note. Dans la formule ci-dessus, K dénote la complexité de Kolmogorov usuelle ou encore sa variante préfixe-free (notée H ci-dessous). En fait, cela importe peu pour une raison simple : toutes les propriétés relatives à cette distance seront vraies à un terme $O(\log(|x|), \log(|y|))$ près et la différence entre $K(z|t)$ et $H(z|t)$ est bornée par $2\log(|z|)$. Pour des raisons historiques et conceptuelles, nous préférerons nous référer à la complexité de Kolmogorov usuelle.

On a alors que *ID* et *ID'* satisfont les axiomes d'une distance *à un terme logarithmique près.*

Les axiomes stricts d'une distance d sont :

$$\begin{cases} d(x,x) = 0 & \textit{(identité)} \\ d(x,y) = d(y,x) & \textit{(symétrie)} \\ d(x,z) \leq d(x,y) + d(y,z) & \textit{(inégalité triangulaire)} \end{cases}$$

Théorème
Les axiomes de distance satisfaits par *ID* et *ID'*, à un terme logarithmique près, sont :

$$\begin{cases} \mathrm{d}(x,x) &= O(1) & (1)\\ \mathrm{d}(x,y) &= d(y,x) & (2)\\ \mathrm{d}(x,z) &\leq d(x,y)+d(y,z)+O(\log(d(x,y)+d(y,z))) & (3) \end{cases}$$

Démonstration. Nous traitons seulement le cas de *ID*. Soit $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ telle que $f(p,x) = x$ pour tout p, x. Le théorème d'invariance nous assure que $K(x|x) \leq K_f(x|x)+O(1)$. Considérons maintenant p comme étant le mot vide, alors on a $K_f(x|x) = 0$. D'où $ID(x,x) = O(1)$. L'égalité $ID(x,y) = ID(y,x)$ est évidente.
Soit maintenant p, p', q, q' les plus petits programmes tels que $U(p,y) = x$, $U(p',x) = y$, $U(q,z) = y$, $U(q',y) = z$. Alors on a : $K(x|y) = |p|$, $K(y|x) = |p'|$, $K(y|z) = |q|$, $K(z|y) = |q'|$. Considérons la fonction injective calculable $\langle\ \rangle : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ (cf. Proposition 1.6 dans la Partie I) qui est telle que $|\langle r,s\rangle| = |r| + |s| + O(\log|r|)$. Soit $\varphi : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ telle que l'on a $\varphi(\langle r,s\rangle, x) = U(s, U(r,x))$. Alors

$$\begin{array}{rcccccc} \varphi(\langle q,p\rangle, z) &=& U(p,U(q,z)) &=& U(p,y) &=& x\\ \varphi(\langle p',q'\rangle, x) &=& U(q',U(p',x)) &=& U(q',y) &=& z \end{array}$$

et ainsi, en appliquant le théorème d'invariance, on obtient :

$$\begin{aligned} K(x|z) &\leq K_\varphi(x|z)+O(1) \leq |\langle q,p\rangle| + O(1)\\ &= |q| + |p| + O(\log(|q|)) = K(y|z) + K(x|y) + O(\log(K(y|z))) \end{aligned}$$

et, de la même façon, $K(z|x) \leq K(y|x) + K(z|y) + O(\log(K(z|y)))$. Et donc on a :

$$\begin{aligned} \max(K(x|z), K(z|x)) &\leq \max(K(y|z)+K(x|y)+O(\log(K(y|z))),\\ &\qquad K(y|x)+K(z|y)+O(\log(K(z|y))))\\ &\leq \max(K(x|y)+K(y|x)) + \max(K(y|z)+K(z|y))\\ &\qquad +O(\log(\max(K(y|z), K(z|y)))) \end{aligned}$$

Ce qui veut dire que l'on a $ID(x,z) \leq ID(x,y)+ID(y,z)+O(\log(ID(y,z)))$, qui est un résultat légèrement plus fort que (3). □

De telles approximations des axiomes des distances sont suffisantes pour le développement de la théorie.
Afin d'éviter les effets de distorsion d'échelle, comme il a été dit dans la section 1.4, cette distance *ID* est normalisée en *NID* (*distance d'information normalisée*) de la façon suivante :

$$NID(x,y) = \frac{ID(x,y)}{\max(K(x), K(y))}$$

Reste le problème que cette distance est non calculable puisque K ne l'est pas. D'où l'approximation suggérée par Vitányi :

> *Considérer* NID *comme une distance idéale qui peut être approximée en remplaçant la fonction de Kolmogorov K par un algorithme de compression.*

2.2 La distance de compression normalisée *(NCD)*

L'approximation de $K(x)$ par $\Gamma(x)$ où Γ est un compresseur[6] ne suffit pas. On doit aussi pouvoir approximer la complexité conditionnelle de Kolmogorov $K(x|y)$. Vitányi choisit l'approximation suivante :

$$\Gamma(y|x) = \Gamma(xy) - \Gamma(x)$$

Les auteurs expliquent ainsi leur intuition :

Pour compresser le mot xy (x concaténé à y),

- *Le compresseur commence par compresser le mot x.*
- *Puis il compresse le mot y en omettant toute l'information de y qui est déjà dans x.*

Ainsi, la sortie du programme n'est pas une compression de y mais une compression de y avec toutes les informations relatives à x qui ont été supprimées, i.e. le résultat est une compression conditionnelle *de x sachant y.*

Notons que l'hypothèse selon laquelle, dans une compression du mot xy, le compresseur commence par compresser x, n'est pas évidente à priori et on peut se poser la question de savoir comment le compresseur récupère x dans xy. On peut considérer comme résolu le cas où x et y sont des mots aléatoires (i.e. incompressibles) ou encore le cas trivial où $x = y$. Mais qu'en est-il en dehors de ces cas extrêmes ? Le fait est que cela fonctionne bien. Est-ce dû au miracle de la modélisation ? ou y a-t-il quelque chose de plus profond ?

Avec cette approximation, plus le fait que $\Gamma(xy) = \Gamma(yx)$ (propriété qui n'est pas toujours évidente : cela va dépendre du compresseur utilisé), on obtient l'approximation suivante de *NID*, appelée *distance de compression normalisée*, et notée *NCD* :

$$\begin{aligned} NCD\,(x,y) &= \frac{\max\,(\Gamma(x|y)\ ,\ \Gamma(y|x))}{\max\,(\Gamma(x)\ ,\ \Gamma(y))} \\ &= \frac{\max\,(\Gamma(yx)-\Gamma(y)\ ,\ \Gamma(xy)-\Gamma(x))}{\max\,(\Gamma(x)\ ,\ \Gamma(y))} \\ &= \frac{\Gamma(xy)-\min\,(\Gamma(x)\ ,\ \Gamma(y))}{\max\,(\Gamma(x)\ ,\ \Gamma(y))} \end{aligned}$$

Noter que le regroupement d'objets réalisé conformément à la distance normalisée de compression *NCD*, et plus généralement la classification par compression, est une *boite noire*[7] comme le remarque Delahaye [15] : les mots sont

6. Cf. Partie I pour une définition formelle des compresseurs.

7. La notion de boite noire est un concept scientifique introduit en 1948, par Norbert Wiener dans : Wiener N. *Cybernetics or Control and Communication in the Animal and the Machine*. The Technology Press, 1948 & 2nd Ed. The MIT Press, 1965. Ce concept est un des principes fondamentaux qui sous-tendent la cybernétique et provient notamment des réflexions pluridisciplinaires des conférences Macy qui ont eu lieu à New-York entre 1942 et 1953 et dont est principalement issue la cybernétique et la théorie de l'information.

regroupés ensemble selon des spécificités qui nous échappent sauf si on en avait déjà une idée au préalable. En outre, on ne peut guère espérer que l'analyse des calculs effectués par le compresseur nous éclairent sur les regroupements obtenus.

Par exemple, qu'est-ce qui caractérise un texte de Tolstoï ? Qu'est-ce qui différencie les *styles* de Tolstoï et Dostoievski ? Reste que cela marche, des textes russes sont regroupés par auteurs par un compresseur qui ignore tout de la littérature russe...

Quand on a affaire à une classification obtenue par compression, on aimerait avoir une idée sur la classification obtenue. Mais ceci est de la sémantique : le compresseur est purement syntaxique et ne "comprenant" rien à ce qu'il fait, on ne peut espérer qu'il nous "aide" dans la compréhension (l'interprétation) de la classification obtenue (cf. section 4).

Cette situation est très proche de celle d'une machine, dans laquelle on a implémenté un sytème formel de déduction, et qui est capable de démontrer au hasard des théorèmes tout à fait complexes. Mais ces théorèmes sont prouvés "hors contexte", sans aucune sémantique. Et dans ce cas, comment les comprendre et les interpréter ? On ne peut évidemment pas espérer des indications de la part de la machine. Du moins dans le contexte actuel.

3 La Google classification

Bien qu'elle n'utilise pas *stricto sensu* la complexité de Kolmogorov, nous allons présenter une autre approche très récente de la classification dûe à Vitányi et Cilibrasi [9], qui fournit un outil qui promet d'être remarquablement performant.

3.1 La distance Google normalisée *(NGD)*

Il s'agit d'une méthode de classification très originale basée sur l'immense masse de données que constitue le Web, et qui sont accessibles grâce à des moteurs de recherche comme Google, qui permettent de faire des requêtes basiques constituées d'un simple mot-clef ou de conjonctions de mots-clefs. Noter que le Web n'est pas une base de données, au sens formel du terme, c'est une sorte de banque de données, ou si l'on veut un (gigantesque) système d'information informel, car les données sur le Web ne sont pas structurées comme c'est le cas des objets dans les bases de données relationnelles. Avec le Web, on a affaire à une forme rudimentaire de structuration, basée sue la notion de graphe et *d'interface graphique*, mais néanmoins nantie d'un langage de programmation objet, en l'occurence, Java. Ce qui est remarquable c'est l'existence d'une norme pour un tel langage de programmation, et qui soit de succroît Turing-complet (cf. section 4.2) – et ceci peut d'ailleurs expliquer le succès (et la mode) de Java et de *l'approche objet* qui sont pour l'essentiel dûs au succès du Web.

Pour donner une idée de cette méthode, citons Alberto Evangelista et Bjorn Kjos-Hanssen [20] :

> ≪ Quand le moteur de recherche Google est utilisé pour rechercher un mot x, Google indique le nombre de succès qu'il a trouvés pour ce mot. Le quotient de ce nombre par le nombre total de pages indexées par Google représente la probabilité d'apparition du mot x sur une page Web donnée [...]. Si un mot y a une probabilité conditionnelle plus élevée d'apparaître, en considérant qu'un mot x apparaît sur la page Web, plutôt que d'apparaître tout seul, alors on peut en conclure que les mots x et y sont corrélés. ≫

Prenons un exemple cité par Cilibrasi et Vitany [8], que nous complétons et mettons à jour [8]. Une recherche sous Google avec les mots-clefs respectifs : "cheval", "cavalier" et "molécule" retourne respectivement 156 millions, $62,2$ millions et $45,6$ millions de succès. Une recherche sur la paire de mots "cheval" et "cavalier" d'une part et "cheval" et "molécule" d'autre part retourne $2,66$ millions et $1,52$ millions de succès. Ceci permet de mettre en évidence une relation plus importante entre les mots "cheval" et "cavalier" qu'entre les mots "cheval" et "molécule".

Un autre exemple avec des tableaux célèbres de peintres : "le Déjeuner sur l'Herbe", "le Moulin de la Galette" et "la Joconde". Appelons les respectivement a, b et c. Une recherche par Google pour respectivement a, b et c donne 446 000, 278 000 and 1 310 000 succès. Alors qu'une recherche respective sur les conjonctions $a+b$, $a+c$ et $b+c$ retourne 13 700, 888 et 603 succès. Clairement, les deux tableaux de Jean Renoir sont plus souvent cités ensemble que chacun d'entre eux avec un tableau de Léonard de Vinci.
De cette façon, la méthode regroupe les tableaux de peintres par artiste, en utilisant ce qui est écrit sur le Web. Voir que ce processus n'associe pas les peintres aux tableaux (on doit les rajouter "à la main").

Formellement, Cilibrasi et Vitany [8, 9] définissent une *distance Google normalisée* par :

$$NGD(x,y) = \frac{\max(\log \Lambda(x), \log \Lambda(y)) - \log \Lambda(x,y)}{\log \Upsilon - \min(\log \Lambda(x), \log \Lambda(y))}$$

où $\Lambda(z_1, ...z_n)$ est le nombre total de succès pour la requête conjonctive $z_1 \wedge ... \wedge z_n$ pour $n \geq 1$ (qui s'écrit sous la forme : z_1 ... z_n sous Google). Si $n = 1$, on a tout simplement que $\Lambda(z)$ est le nombre total de succès pour la requête z. Υ est le nombre total de pages Web indexées par Google.

3.2 Discussion sur la méthode

Citons quelques points relatifs à l'utilisation d'une telle méthode de classification (la liste n'est pas exhaustive) :

8. Avec la réserve signalée dans le point 4 de la section 3.2.

1) Le nombre d'objets considérés en vue d'obtenir une classification et celle des représentants canoniques des différents corpus à classifier n'est pas choisi, ni même bornable à l'avance et se trouve être en continuelle évolution. Cet aspect dynamique et non contrôlé de la définition d'une famille est une expérience totalement nouvelle, du moins dans une approche formelle de la classification.

2) Des domaines qui échappent à priori à la classification par compression comme le domaine pictural (normalisation de tableaux divers de peintres à priori impossible ou tout au moins pas évidente à l'heure actuelle, ou encore (et pour les mêmes raisons) la sculpture, l'architecture, la photographie, le cinéma d'auteur, etc. deviennent facilement appréhendables. Car dans cette approche, ce ne sont pas les oeuvres en elles-mêmes qui sont considérées mais un discours (celui figurant sur le Web) sur ces oeuvres. Ce discours relève donc d'un "vrai" langage, au sens d'un langage naturel ou d'un langage formel. Noter que la notion de "langage pictural" est juste une métaphore, du moins si l'on considère que la communication infraverbale n'est pas un langage au sens usuel.

Le discours considéré par Google est celui des mots-clefs et des relations entre eux, mots-clefs provenant des requêtes proposées pour la NGD *et figurant dans les textes des utilisateurs du Web.*

On notera que certains travaux pourraient servir de base à une approche algorithmique (voire à une normalisation) des oeuvres d'art picturales, du cinéma d'auteur, etc. Nous pensons, en particulier, aux travaux de la psychanalyste Murielle Gagnebin qui a élaboré une théorie de *l'esthétique* et de *l'acte créateur*, fondée sur la psychanalyse et la philosophie. Ce modèle métapsychologique permet de dégager, à partir d'oeuvres d'art – et uniquement des oeuvres et non pas en considérant des discours sur lesdites oeuvres, ni des considérations sur les artistes – les mécanismes psychiques fondamentaux impliqués dans ces oeuvres. On peut imaginer qu'un tel modèle puisse être implémenté en un système expert.

3) Il y a cependant une limitation très importante à cette méthode, à savoir celle sous-jacente à ce qu'il est convenu d'appeler : *l'hypothèse du monde clos.* Ce qui peut se traduire ici par : *le monde selon Google*[9]*, l'information selon Google*, etc.

Si Google trouve et retourne une information, comment peut-on vérifier sa *pertinence* ? Autrement dit, qu'est-ce que cela veut dire ? Comment définir (de façon générale) une notion de pertinence pour les informations trouvées par Google ? Seule certitude, celle de l'incertitude ! Noter par ailleurs que lorsqu'on est confronté à des échecs successifs sous Google avec plusieurs séries de mots-clefs, on abandonne plus ou moins sa requête initiale et on la modifie (on change sa sémantique) jusqu'à ce que Google donne des réponses "pertinentes". Cette sorte d'échec est celui de la négation dans le langage de programmation Prolog (appelée la *négation par échec*), qui est plus faible que la négation classique de la logique et qui est connexe à l'hypothèse du monde clos des bases de données.

9. Irving J. *Le monde selon Garp.* Le Seuil, 1978.

Il est raisonnable d'abandonner de telles requêtes infructueuses et par conséquent de considérer les conjonctions de mots-clefs sous-jacentes comme dépourvues de signification. Toutefois, on ne doit jamais perdre de vue que tout ceci est relatif au monde clos, relativement petit, des données sur le Web, le seul accessible à Google ni sous-estimer le caractère fluctuant des informations disponibles sur le Web. Quand une requête réussit, le risque est d'arrêter la succession de requêtes en jeu et :

- D'oublier que des requêtes précédentes ont été essayées et ont échoué.
- D'omettre d'autres requêtes qui pourraient conduire à des réponses encore plus "pertinentes" de Google.
- D'oublier que les réponses que donne Google à une requête sont les réponses qu'il trouve à un instant donné dans une sorte *d'instantané* du Web, instantané qui est quelque part en contradiction avec ce qui fait l'essence même du Web : un système d'information en continuelle évolution, les mises à jour (insertion, suppression, modification d'informations, etc.) de ce système étant faites de surcroît dans un mode massivement parallèle puisque Google utilise près de 700 000 ordinateurs comme serveurs ! Les réponses de Google à une requête ne constituent en aucun cas des réponses *définitives*, ou si l'on veut *une réponse absolue*, comme on a pu en prendre l'habitude avec les exécutions de programmes parfaitement déterministes (et vu sous cet angle Prolog est infiniment plus "déterministe" que Google), ou avec avec les bases de données (quand elles sont bien écrites !). Réponses qui peuvent être d'ailleurs plus ou moins contradictoires entre elles, selon les sites que Google a retenus. En particulier, on peut être tenté de s'arrêter dès que l'on trouve un site avec une réponse qui nous convienne (c'est ce que l'on fait en réalité la plupart du temps).

4) On voit donc poindre des difficultés dans l'appréhension théorique du traitement des informations du Web par Google (et d'ailleurs par tout moteur de recherche). Nous nous sommes placés pour cette réflexion dans une perspective en quelque sorte idéale où Google fonctionne pour ses recherches selon des critères scientifiques ou tout au moins avec une certaine transparence – en particulier, sur la façon dont les pages Web sont indexées ou encore sur le nombre de pages Web réellement indexées. Signalons cependant qu'il y a un certain nombre de contreverses sur cette indexation et par conséquent sur l'exactitude des résultats trouvés par Google, notamment sur le nombre d'apparitions d'un mot donné sur les pages de la totalité du Web (sans même parler du contenu de ces pages). Le fait est que certains résultats de requêtes Google peuvent être très surprenants : la "logique googléenne" est pour le moins étrange (du moins si on la compare à la logique booléenne) comme le montre Jean Véronis de façon très percutente (et tout-à-fait scientifique) dans son blog [10].

10. Véronis J. Web : Google perd la boole. (Transl : Web : Googlean logic.) Blog. January 19, 2005 de : http ://aixtal.blogspot.com/2005/01/web-google-perd-la-boole.html .

Il est absolument nécessaire de formaliser la notion d'information sur le Web et les relations qui gouvernent les données qu'il contient, comme cela a été fait par Codd avec le modèle relationnel des bases de données, dans les années 70. Avant Codd, l'organisation et la structuration des données et de l'information dans une machine et leur accessibilité par la notion de requête, n'était sous-tendue par aucune approche mathématique solide et reposait pour l'essentiel sur des astuces techniques, ce qui est encore le cas pour les informations sur le Web. Cette remarquable approche innovante de la classification au moyen de Google et des données sur le Web en est encore donc à ses balbutiements.

Dans les sections qui suivent, nous présentons des considérations relatives à des notions formalisées et des idées qui ne sont pas encore formalisées (comme celles soulevées dans 1.3). Un travail de recherche est en cours et divers articles sur le sujet sont en préparation [11].

4 Classification, approches Bottom-Up versus Top-Down et dualité

4.1 Modes Bottom-Up versus Top-Down

Ces approches de la classification par compression et avec Google (relativement à l'information apparaissant sur le Web dans le deuxième cas), sont extrêmement originales et présentent un immense intérêt. En effet, depuis l'essor prodigieux de l'informatique et des réseaux avec le Web, l'information a aqcuis en quelque sorte un nouveau statut et ces approches, reposant d'ailleurs sur ce qu'elles permettent justement d'expliciter, nous aident à appréhender cette nouvelle situation de l'information comme elle se présente dans le monde actuel, bien réel, des machines.

Nous avons souligné ci-dessus la difficulté à définir formellement la classification obtenue par compression ou avec Google, comme cela est fait avec la classification au moyen du modèle relationnel des bases de données de Codd. Du moins si on ne se contente pas d'un arbre ou d'un graphe comme étant une telle formalisation. Dans ce cas, la "récupération" d'information avec de telles structurations est peu ou mal formalisée. Noter d'ailleurs que c'est exactement la situation des informations figurant dans des fichiers "rangés" dans un système d'exploitation (un *OS*) puisqu'aucune base de donnée n'est sérieusement intégrée aux différents systèmes actuels (*Unix, Linux, MacOs, Windows* et leurs variantes).

Il nous apparaît alors opportun de reconsidérer ces différentes approches de la notion de classification dans une optique plus générale avec *les deux grands modes fondamentaux dont on dispose pour les définitions d'objets mathématiques*

Voir aussi : http ://aixtal.blogspot.com/2005/02/web-le-mystre -des-pages-manquantes-de.html and http ://aixtal.blogspot.com/2005/03/ google-5-milliards-de-sont-partis-en.html , 2005.

11. Notamment [25], [26] et Ferbus-Zanda M. *Logic, Information System and Metamorphosis of a Fundamental Duality.* En preparation.

et informatiques et que l'on retrouve dans l'exécution des programmes informatiques. Ces deux modes fondamentaux pour définir les objets mathématiques et les données ou programmes informatiques sont :

- Les *définitions itératives* (basées sur *la réunion ensembliste*)
- Les *définitions inductives* ou *par réccurrence* ou encore *"récursives"* (basées sur *l'intersection ensembliste*).

On a par exemple deux telles sortes de définitions pour les formules propositionnelles ou les termes et formules de la logique du premier ordre.

On rappelle que dans le modèle de calcul de Stephen Kleene [12] des fonctions récursives, les fonctions récursives partielles doivent être closes par trois (méta) opérations : *la composition, la récurrence primitive et la minimisation.*

- Les définitions itératives sont connectées à la minimisation (et à la notion de *successeur*). On peut qualifier ce type de définition comme étant une caractérisation de type "*Bottom-Up*".
- Les définitions inductives sont connectées à la récurrence primitive (et à la notion de *prédécesseur*). On peut qualifier ce type de définition comme étant une caractérisation de type "*Top-Down*".

Noter que la composition est relative aux deux types de caractérisation : bottom-up et top-down. Nous avons donné, dans la partie I, des formalisations de l'aléatoirité pour les objets infinis qui relèvent respectivement de ces deux approches bottom-up et top-down (cf. Partie I, section 5.1 et 5.2).

On retrouve ces deux modes dans l'exécution des programmes informatiques :

- Le *mode d'exécution itératif* est qualifié de *Bottom-Up*.
- Le *mode d'exécution récursif* est qualifié de *Top-Down*.

Ce deuxième cas nécessite l'utilisation d'une *pile* qui va croître dans un premier temps puis décroître et qui permet de stocker tous les résultats des calculs intermédiaires jusqu'à arriver aux "cas de base", i.e. aux étapes initiales de la définition inductive du programme qui est exécuté. Dans l'exécution d'un programme itératif, toutes les informations nécessaires au déroulement de cette exécution sont à tout moment disponibles et aucune pile de stockage n'est donc nécessaire. On voit ainsi que ces deux modes d'exécution diffèrent radicalement, du moins du point de vue d'un informaticien.

12. Kleene caractérise formellement (et dans son intégralité) la notion de *fonction récursive* (*calculable*), en ajoutant le schéma de minimisation (1936) aux schémas de composition et de récurrence primitive – ces deux schémas caractérisant les *fonction primitives récursives* qui forment une sous-classe stricte des fonctions calculables : la *fonction d'Ackermann* (1928) est une fonction calculable qui n'est pas primitive récursive. D'un point de vue programmation, le schéma de minimisation correspond à la boucle `while` (utilisée sous la forme : `while F(x) do P` où `F(x)` est une propriété à valeur booléenne et `P` est un programme). On peut également consulter la note 23. Le lecteur peut se reporter au livre de Shoenfield J. *Recursion theory*, Lectures Notes in Logic, (nouvelle édition) 2001.

Noter que le mode d'exécution itératif (resp. récursif) est imposé par une définition itérative (resp. inductive ou récursive) du programme à exécuter, bien que certains cas de programmes récursifs puissent être éxécutés de façon itérative (et donc sans la nécessité d'une pile [13]).

De la même façon, remarquons qu'il existe *deux modes* – appelons-les *Bottom-Up* et *Top-Down* – qui sont utilisés dans l'approche de la classification d'information et/ou d'objets (du monde réel) représentés formellement au moyen de mots et plus généralement de textes [14] ou encore d'ensembles de mots, définis sur un alphabet (que l'on peut prendre, rappelons-le, binaire) :

- *Dans le mode Bottom-Up, on entre en quelque sorte dans le détail des informations, autrement dit, on accède au contenu des textes (aux mots) qui représentent les différentes informations et/ou objets que l'on désire classifier (et à la signification des textes et/ou mots en question). Les textes, les ensembles de mots, etc. sont appréhendés de* l'intérieur *et leur signification est primordiale.*
- *Dans le mode Top-Down, on n'accède pas de la sorte au contenu des textes ou aux ensembles de mots, etc. Ceux-ci sont en fait appréhendés depuis* l'extérieur, *en quelque sorte "de haut bas"* [15], *ou pour le dire autrement, on utilise une sorte "d'oracle" pour appréhender les textes ou ensembles de mots, i.e. des moyens qui sont* extérieurs *à la compréhension des textes ou au contenu des ensembles et à la compréhension des mots.*

Prenons un exemple avec des classifications utilisant des mots-clefs pour des famille de textes (littéraires ou scientifiques) à structurer. On utilise alors les deux modes bottom-up et top-down pour classifier des textes de la façon suivante :

1) Pour choisir les mots-clefs, on utilise en général une approche bottom-up : les mots-clefs sont *choisis à partir du contenu des textes et de leur signification* – et ceci à dessein d'une recherche ultérieure. Et plus précisément, certains mots seront considérés comme pouvant servir de mots-clefs et seront déclarés comme tels. Typiquement, pour les articles scientifiques, un certain nombre de mots-clefs sont dégagés par l'auteur, l'éditeur de revue, le bibliothécaire, etc. en vue d'une classification future incluant le texte considéré. Cette façon de procéder

13. Il s'agit des définitions *récursives terminales*. Dans certains langages de programmation comme *LISP*, de tels programmes récursifs terminaux sont en général (lorsque l'exécuteur de programme est bien écrit) exécutés de façon itérative. Les programmes récursifs terminaux constituent un *cas limite* entre les programmes itératifs et les programmes récursifs.

14. Selon le niveau *d'abstraction* (ou si l'on préfère selon le degré de *raffinement*) où l'on se situe, un *texte* sera représenté par un *mot*, binaire par exemple, (les blancs séparant les différents mots du texte considéré étant alors encodés dans le mot binaire qui représente le texte) ou alors par une *suite de mots* binaires (chaque mot du texte étant représenté par un mot binaire de la suite). Rien n'interdit du reste de considérer des suites ou des ensembles de textes, et de mixer les suites et/ou ensembles... Dans cet article, nous considérons pour l'essentiel, et en particulier pour les exemples, des codages de textes au moyen de mots binaires et non pas de suites de mots binaires et nous nous intéresserons aux ensembles de tels textes.

15. C'est une façon de voir les choses ! Ce que reflète la terminologie anglo-saxonne "top-down" (que nous adoptons pour le français). Ce qui est essentiel c'est l'appréhension depuis *l'extérieur*, en opposition à l'appréhension depuis *l'intérieur*, des textes en question.

suppose la lecture préalable des textes, ainsi que leur compréhension.

Remarquer que la traduction d'un texte d'un langage naturel vers un autre, comme par exemple, la traduction de cet article du français vers l'anglais, exige une telle lecture/compréhension (subtile) des textes [16].

On peut aussi choisir des mots-clefs pour un texte, sur d'autre critères qu'en lisant le texte proprement dit. Par exemple, en lisant et comprenant le résumé (abstract) et/ou la table des matières (contents). Et dans ce cas, il s'agit également d'un mode bottom-up. On peut aussi consulter le contenu d'un index éventuel et dans ce *cas limite*, il s'agit alors d'un mode de type top-down : aucune compréhension des mots de l'index n'est requise pour y sélectionner des mots (mais rien n'interdit non plus de les comprendre), on s'intéresse juste aux différentes occurences des mots de l'index et à leur importance qui est connotée dans tout index bien fait. On peut d'ailleurs se passer d'un index et réaliser de tels calculs d'occurences de mots dans les textes : et c'est justement ce que fait Google pour ses recherches. Dans la pratique, les deux modes bottom-up et top-down sont souvent utilisés de façon conjointe (mode *mixte*).

Qelque soit la méthode utilisée, le choix des mots-clefs suppose en général (mais ce n'est pas toujours le cas) la connaissance préalable ou tout au moins l'idée à priori de la classification, plus ou moins préétablie, dont ces mots-clefs vont relever par la suite Cette connaissance est une forme (très abstraite) de *sémantique* [17] et peut évoluer avec le temps (nouveaux textes lus, nouveaux mots-clefs, etc.), et en général ce n'est pas la personne qui écrit le texte qui possède cette connaissance mais plutôt la personne chargée de "gérer" la classification des textes.

2) Une fois que l'on a choisi les mots-clefs d'un texte – et ceci quelque soit l'approche utilisée – et que ces mots-clefs et le texte sont répertoriés d'une façon ou d'une autre – on peut alors considérer qu'il existe une sorte de classification pour ce texte avec d'autres textes déjà répertoriés, via ses mots-clefs. On pourra alors, à partir de mots-clefs donnés, rechercher tous les textes ayant ces mots-clefs de définis comme tels. On voit ainsi se dégager, à partir des mots-clefs ainsi utilisés, une notion de *requête*. Dans une conception élargie des mots-clefs – et

16. Avec un traducteur automatiqe, qui est purement syntaxique, on se retrouve avec par exemple : Alonzo Church traduit par Église d'Alonzo (cf. Google) !

17. Signalons l'émergence d'un nouveau concept : la notion de *thésaurus* qui constitue une telle sorte de sémantique abstraite liée à une classification. Un thésaurus est un type particulier de *langage documentaire* (encore un nouveau concept), qui consiste à choisir pour un domaine particulier, un ensemble de mots reliées entre eux par des liens de différente nature (*synonymie*, *métaphore*, *analogie*, *comparaison*, *lien hiérarchique*, etc.) formant ainsi un graphe. Un thésaurus constitue ainsi une sorte de *vocabulaire normalisé* et classifié pour le domaine en question, ou pour le dire autrement, les mots d'un thésaurus forment un *dictionaire hiérarchisé* de mots-clefs pour un domaine. On peut alors rajouter ou non les définitions des mots ou se contenter de la classification des mots selon l'usage que l'on aura du thésaurus. C'est un outil de travail remarquable utilisé initialement dans les discipines liées à la documentation, pour les grandes banques de données, etc. et qui commence à être utilisé un peu partout. L'élaboration d'un thésaurus se fait selon un mode bottom-up ou top-down ou encore en mixant les deux méthodes, exactement comme dans le cas des mots-clefs. Nous revenons en détail sur la notion de thésaurus dans la section consacrée aux bases de données (cf. section 4.2).

c'est sur ce principe que fonctionne Google – on pourra aussi rechercher tous les textes contenant ces mots-clefs (i.e. ayant textuellement ces mots-clefs dans leur contenu), et dans ce cas, inutile de définir préalablement les mots-clefs relatifs à un texte.

Noter qu'alors (et dans tous les cas), dans ce type de recherche utilisant des mots-clefs, et ce point est fondamental, *il s'agit d'une appréhension des textes, autrement dit, d'une approche de leur classification selon un mode opératoire de type top-down*. C'est-à dire qu'à partir d'un ensemble de mots-clefs (une "conjonction" de mots-clefs), et qui va être une forme de question posée à un certain *oracle*, de façon à pouvoir appréhender des textes depuis l'extérieur, on va choisir un certain nombre de textes – et donc sans les lire ni les comprendre – parmi un ensemble de textes qui peut être très grand et même gigantesque comme le Web. Les textes ainsi sélectionnés pourront alors être (éventuellement) lus, et d'une certaine façon compris (ou tout au moins mieux compris). On pourra aussi les regrouper avec d'autres textes possédant des mots-clefs en commun et établir ainsi une classification de ces textes.

3) Dans l'approche de la classification avec Google, on se trouve dans une situation tout à fait similaire : le choix des mots-clefs proposés dans les requêtes pour Google (qui sont en fait des conjonctions de mots-clefs) peut être fait selon :

- Un mode bottom-up et alors ce choix est issu de la lecture et de la compréhension du contenu du Web.
- Un mode top-down, et ce choix est alors établi sur des critères tout à fait extérieurs au contenu du Web – bien qu'il soit difficile de ne pas être "influencé" par des lectures précédentes du Web...

On voit bien qu'en général, on mixe les deux mode bottom-up et top-down dans le choix des mots-clefs.

Une fois que l'on a choisi les mots-clefs – et ceci quelque soit le mode utilisé – on dispose alors d'une sorte d'oracle pour appréhender le contenu du Web, autrement dit, avec *la requête Google composée de ces mots-clefs, on va sélectionner des textes sur le Web – et même plus généralement des hypertextes ou du multimédia, autrement dit, des pages Web – avec un mode opératoire de type top-down*. Ces textes pourront alors être lus, classifiées, etc. Le moteur de recherche Google fonctionne donc comme un *oracle*, qui, lorsqu'on lui soumet une question (des mots-clefs), répond par un ensemble de sites Web. Son mode de fonctionnement est comme pour les oracles, invisible à l'utilisateur.

Rien n'empêche évidemment de parcourir le Web selon un mode bottom-up, autrement dit, en se passant des requêtes et en accédant alors au contenu du Web en en allant d'une page Web à une autre au moyen des *liens hypertextes*. Ces liens hypertextes font toute l'originalité du Web et sont très intéressants à étudier d'un point de vue théorique car ils véhiculent une forme de sémantique. *la notion de mot-clef (et plus généralement de mot) est en quelque sorte un concept limite : entre syntaxe et sémantique*. En général (et tout le monde l'a

plus ou moins expérimenté), *la navigation se fait en mixant les deux types d'approche : bottom-up avec les liens hyper-textes et top-down avec les requêtes.*

Noter que comme dans le cas précédent, le choix des différents mots-clefs proposés à Google en vue d'une classification est également une forme de sémantique. Remarquons toutefois, que dans une approche top-down du choix des mots-clefs, on peut tout à fait proposer des mots-clefs choisis complètement *au hasard* et par le calcul (avec par exemple des outils statistiques), en déduire des classifications de textes ainsi sélectionnés. Cette façon de procéder peut être intéressante si le volume des textes ainsi appréhendé est très large et ceci est bien le cas avec le Web. On peut toutefois douter qu'une telle approche de la classification – si elle est fondamentalement aléatoire – puisse donner des résultats réellement intéressants. Elle peut, toutefois, être combinée avec une approche plus "déterministe".

Reprenons maintenant le titre de cette section : *Modes Bottom-Up versus Top-Down*. On peut effectivement se demander pourquoi il existe deux tels modes dans la définition des objets mathématiques et informatiques, ainsi que dans l'exécution des programmes informatiques. *De Facto*, deux tels modes existent et ce sont les deux grands modes fondamentaux qui ont été dégagés, en particulier, par les théoriciens de la calculabilité dans la première moitié du XXème siècle. Nous avons vu que ces deux modes pouvaient être également considérés dans l'approche de la classification d'information et nous en avons donné un exemple avec les mots-clefs. Nous avons également vu comment l'utilisation de Google pour faire des recherches sur le Web, pouvait relever de ces approches.

Ceci nous montre que ces modes bottom-up et top-down dépassent largement le cadre de la classification et concernent, en réalité, tout traitement de l'information et par là-même toute théorie, un tant soit peu abstraite, de l'information. Cela concerne également toutes les disciplines qui ont affaire, d'une manière ou d' une autre, avec la notion de *représentation* ou encore de *définition* , de *description* , etc, dont la logique, la complexité de Kolmogorov et l'informatique, la sémiotique et également les sciences de la cognition car, ainsi que nous l'exposons dans [26], *le traitement de l'information par le cerveau humain s'articulerait fondamentalement autour de ces deux modes opératoires*. C'est en tout cas une approche particulièrement intéressante de la cognition, et qui se trouve en grande partie éclairée par l'évolution de la logique mathématique et de l'informatique.

Dans le cadre de cet article, nous allons nous intéresser aux modes bottom-up et top-down, dans deux types de situations concenant la classification et qui généralisent ce que l'on a décrit pour les mots-clefs, à savoir :

- La formalisation logique des systèmes d'information au moyen des bases de données (section 4.2).
- L'approche ensembliste de la notion de regroupement fondée par la théorie axiomatique des ensembles de Zermelo-Fraenkel (ZF). Nous y considérerons tout particulièrement le schéma de compréhension de ZF (section 5).

Ces considérations vont nous éclairer sur le rôle que peut jouer la complexité de Kolmogorov dans la classification d'informations et plus précisément dans la notion de *regroupement* d'informations. Nous allons être amené à reconsidérer les notions d'intentionnalité, d'abstraction, de sémantique et de représentation dans ce cadre (cf section 6).

Par ailleurs, le fait que deux tels modes existent, aussi bien pour les définitions mathématiques et informatiques d'objets, de fonctions et de programmes, que pour l'exécution de ces programmes, est déjà profondémment intéressant en soi. Le fait de retrouver ces deux modes pour les diverses formes de traitement de l'information et dans différentes disciplines, fait de cette observation un sujet de recherche passionant. Clairement, ces deux modes *complémentaires* l'un de l'autre, forment une *relation de dualité*, une sorte de correspondance entre deux façons de procéder à la fois distinctes et en quelque sorte similaires l'une de l'autre [18]. Et plus précisément, on a vu que l'approche bottom-up (sur laquelle reposent les définitions itératives), est fondée sur la notion de réunion ensembliste et l'approche top-down (sur laquelle reposent les définitions inductives) est basée sur l'intersection ensembliste. Il est donc tout à fait naturel de reconsidérer ces approches dans le cadre des algèbres de Boole, théorie où la notion de dualité est typique, ce que nous faisons dans [25].

D'autres dualités fondamentales pour la logique et l'informatique sont également développées dans ces articles. Notamment, la dualité *syntaxe versus sémantique*, et également, la dualité *fonctionnel versus relationnel* [19] et qui concerne notamment la relation entre les algorithmes et la programmation (fonctionnelle) d'une part et les systèmes d'information discrets [20] et leur formalisation d'autre part. Rappelons que l'objet des systèmes d'information discrets est l'organisation (la structuration) d'informations de toute nature (admettant une représentation discrète) avec pour objectif de pouvoir extraire facilement des informations particulières. Clairement les systèmes d'informations sont reliés à la classification aussi il nous apparaît intéressant de les présenter dans cet article. Nous articulons cette présentation autour de la dualité bottom-up vs top-down, qui se trouve ainsi illustrée.

18. La notion abstraite *d'ismorphisme* en mathématiques est une forme de dualité. Certaines dualités ne se réduisent pas à des isomorphisme. Typiquement, les algèbres de Boole avec l'opération de complément (en plus des opérations additive et multiplicative), contiennent une dualité interne et sont à la base de dualités profondes comme la dualité de Stone qui relie la famille des algèbres de Boole et certains espaces topologiques. L'opération de complément est à la source de bien des problèmes et de résultats profonds...

19. Depuis l'invention par Gottlob Frege à la fin du XIXème siècle, de la logique mathématique et la formalisation du langage mathématique qui en est issue, le mathématicien est d'une certaine façon confronté, *de facto*, à deux catégories distinctes de symboles mathématiques : les *symboles de fonction* et les *symboles de relation* (ou de prédicat), en complément des symboles représentant les objets. A chacune de ces deux grandes classes de symboles correspondent respectivement les algorithmes et les systèmes d'information.

20. Les systèmes d'information pour lesquels nous mettons en évidence un type de programmation que nous qualifions de *programmation relationnelle* dans un rapport de recherche : Ferbus-Zanda M. *La méthode de résolution et le langage Prolog.* Rapport LITP, No-8676, 1986. Nous y présentons le lien entre la programmation fonctionnelle et la programmation relationnelle.

4.2 Systèmes d'information et bases de données : une approche formelle

Signalons avant tout le lien suivant : *les bases de données (les BD) sont aux systèmes d'information ce que les programmes informatiques sont aux algorithmes : une présentation, une écriture formelle, mathématique et la possibilité d'un traitement tout aussi formel*. Du moins si l'on considère ces termes avec une acceptation élargie (et intuitive) : les algorithmes, tout comme les sytèmes d'information, se présentant alors, d'une façon plus ou moins claire, dans un langage naturel et comportant en géréral un certain nombre de choses implicites plus ou moins importantes voire de non-dits beaucoup plus fâcheux. Remarquons qu'il existe des algorithmes et des systèmes d'information depuis des périodes très anciennes [21]. Dans les deux cas, cette écriture formelle est réalisée dans un cadre qui est pour l'essentiel la logique mathématique. On remarquera que la programmation et les algorithmes sont plus particulièrement liés au lambda-calcul alors que les bases de données et par conséquent les systèmes d'information sont plus spécialement liées à la théorie des ensembles.

En ce qui concerne les programmes et les algorithmes, signalons le travail remarquable de Yuri Gurevich [16], qui, grâce à une notion abstraite de machine : les Abstract State Machines (*ASM*) basée sur la théorie des modèles de la logique du premier ordre, fonde la notion d'algorithme, de la façon la plus raffinée qui soit : non seulement *en fondant la notion même d'algorithme, mais également en formalisant leur mode opératoire*. Et plus précisément, Gurevich s'intéresse à la *sémantique opérationnelle* – i.e. *la façon dont les algorithmes et les programmes sont exécutés* (et dont l'aboutissement est l'écriture d'un *interpréteur* et/ou d'un *complilateur* et d'un *exécuteur* pour les programmes). Ce point de vue opérationnel, éminemment constructif, complète ce qui est appelé la *sémantique dénotationnelle* – et qui rend compte de ce que *les algorithmes et les programmes calculent* [22].

C'est d'ailleurs en ce sens que Gurevich énnonce sa thèse :

> ≪ *Les ASM capturent le* pas à pas *d'exécution des algorithmes séquentiels.* ≫

Pour Gurevich, un algorithme donné (et en particulier, tout programme informatique) "est" une ASM particulière qui va mimer son fonctionnement et avec

21. On retrouve des descriptions exhaustives d'algorithmes dès l'époque des Babyloniens (IIème millenaire av. J.-C. – IIème siècle ap. J.-C.) concernant le commerce et les impôts. L'origine des sytèmes d'information est très récente et son succès est dû de la mécanographie (fin du XIXième siècle) et au développement de l'informatique, mais on peut trouver des traces de ce que l'on considèrerait aujourd'hui en substance comme des systèmes d'information – *un choix et une organisation de la présentation d'informations données et sur un sujet donné*, avec par exemple, le recensement romain – également à une époque très reculée, du moins si on fait abstraction de l'importance du volume des données concernées.

22. On notera qu'il s'agit respectivement de la sémantique de Arend Heyting et de la sémantique de Alfred Tarski.

ce point de vue, un algorithme est formel. Ainsi la thèse de Gurevich étend la thèse de Church-Turing [23] – tout au moins pour les algorithmes séquentiels – en ce sens qu'elle l'infère. Et plus précisément, la thèse de Church-Turing concerne la sémantique dénotationnelle (les divers modèles théoriques envisagés sont équivalents entre eux, on dit encore qu'ils sont *Turing-complets*). Gurevich étend cette thèse à la sémantique opérationnelle, définissant ainsi avec les ASM un modèle de calcul *algorithmiquement complet* (cf. également section 7 et [24]). Ce qui est remarquable dans les ASM, c'est que leur formalisation est en réalité simple et naturelle, alors que ce n'est en général pas le cas des diverses approches de la sémantique opérationnelle des programmes informatiques. Nous revenons sur les ASM (et sur leur relation avec la complexité de Kolmogorov et la classification) dans la conclusion.

En ce qui concerne, les systèmes d'information (qui sont une notion intuitive) et leur modélisation au moyen des bases de données (approche formelle), nous allons voir que, d'un point de vue historique et conceptuel, les choses ne sont pas déroulées aussi "facilement" qu'avec la programmation et la formulation de modèles théoriques de la notion de calculabilité, qui de fait, ont précédé l'apparition des ordinateurs. Dans le cas des systèmes d'information, l'inverse va se produire.

Pour ce qui est des bases de données, rappelons comme nous l'avons évoqué plus haut, que la première formalisation de la représentation des informations et de leur traitement – et c'est en gros ce que l'on appelle aujourd'hui les systèmes d'information – est dûe à Codd en 1970. Il s'agit du modèle relationnel des bases de données [10]. La très grande originalité de Codd est d'avoir pressenti qu'il existait des mathématiques dans la façon de "gérer" les informations dans les ordinateurs. Ceci peut paraître évident aujourd'hui mais à l'époque où Codd a conçu son modèle théorique – on fonctionnait encore avec des cartes perforées pour les programmes – les fichiers informatiques étaient stockés dans le plus grand désordre [24].

23. La thèse de Church-Turing énonce le fait suivant : "*Tout traitement ou calcul réalisable de façon mécanique, autrement dit, tout ce qui est calculable sur machine, peut être réalisé sur une machine de Turing*" (1936). Ainsi cette thèse affirme que la notion (intuitive) de *calculabilité effective* coïncide avec une notion mathématique formelle : *la calculabilité avec les machines de Turing*. Cette thèse a été énoncée antérieurement par Alonzo Church (1932) dans le modèle du λ-calcul (thèse de Church) qui à l'époque apparaissait comme beaucoup plus "théorique" que le modèle de calcul des machines de Alan Turing. A ce sujet, on peut consulter la note 55. Nous abordons le modèle de calcul de Kleene (1936) des fonctions récursives dans la section **??**. Une première définition formelle (complète) de la notion de fonction calculable a été trouvée par Jacques Herbrand (1931) et formalisée par Kurt Gödel (1932).

24. Le premier grand système d'exploitation a avoir utilisé la notion d'arbre (en fait de graphe) pour le stockage des fichiers est le système *Multics* qui a été conçu en 1965 (et progressivement abandonné depuis le milieu des années 80 jusqu'en 2000). Le système *Unix* (dont sont issus la plupart des OS actuels) qui l'a remplacé est entièrement dérivé de Multics, et intègre alors la notion d'utilisateurs multiples. Multics a également une une influence bénéfique sur le stockage des données dans les ordinateurs, puisqu'on parle de *modèle hiérarchique* et de *modèle réseau*. On notera toutefois que ces "modèles" n'ont reçu l'appellation de modèle qu'après l'invention du modèle relationnel de Codd ! L'organisation en graphe du Web est également issue de l'organisation des fichiers sous Multics.

Un des aspects les plus fondamentaux et sans précédent du modèle relationnel de Codd est la formalisation de la notion de *requête* qu'il fonde sur un calcul qu'il crée : *l'algèbre relationnelle* et qui est une sorte de *logique combinatoire avec des opérateurs opérant sur des tables*[25] reliées entre elles, comme certains opérateurs ensemblistes (*réunion*, *intersection*, *produit cartésien* et *projections*) ainsi que des opérateurs que Codd introduit comme la *sélection* et la *jointure* – cette dernière se trouvant être un opérateur fondamental aussi bien pour les bases de données relationnelles que comme opération logique. Codd développe également une *théorie de la normalisation* pour traiter du difficile problème de la *redondance* d'information dans les systèmes.

Aussi surprenant que cela puisse paraître, Codd (chercheur chez IBM) a vraiment dû se battre[26] pour imposer son modèle. C'est une petite société à l'époque, la société *Oracle*[27] qui en a vu tout l'intérêt et qui en a réalisé une première implémentation en 1980. La société IBM en a réalisé une quelques années plus tard. Actuellement, la plupart des SGBD reposent sur le modèle relationnel de Codd. Mentionnons que les bases de données sont encore largement sous-utilisées à l'heure actuelle dans nombre de domaines qui y gagneraient beaucoup. Cette situation devrait radicalement changer à l'avenir, compte tenu de l'importance de l'information numérique (qui s'impose à une vitesse qui dépasse largement tout ce qu'on aurait pu imaginer il y a encore peu de temps).

Citons également un autre modèle théorique des bases de données, le *modèle Entité/Association* de Peter Pin-Shan S. Chen [6], qui est une approche formelle des bases de données plus abstraite que celle de Codd, mais qui repose néanmoins pour l'essentiel sur le modèle relationnel de Codd. Ce modèle, dans lequel une base de donnée est représentée sous forme graphique – et qui rappelle la notion *d'organigramme* utilisée pour modéliser les programmes informatiques dans les années 60-70 – a notamment engendré tout un système de notation graphique pour la modélisation, le langage *UML*[28]. Le modèle théorique de Chen nous paraît également très important et il nous semble qu'il est loin d'avoir été utilisé en profondeur. Les *bases de données conceptuelles*, comme nous les appelons, fondées sur le modèle Entité/Association et qui constituent une extention abstraite logique des bases de données relationnelles, sont également appelées à jouer un rôle fondamental dans l'avenir en ce qui concerne le traitement de l'information, la classification et d'une façon générale toute théorie algorithmique de l'information.

25. et qui n'est pas sans rappeler le langage de programmation *Cobol*, créé en 1959.

26. Dans son livre ([11], 1990), Codd écrit la dédicace suivante : ≪ A mes camarades pilotes et à l'équipage de la Royal Air Force durant la deuxième guerre mondiale, et à mes professeurs de l'université d'Oxford. Ces personnes ont été la source de ma détermination pour me battre pour ce que je croyais être vrai durant les dix années ou plus pendant lesquelles le gouvernement, l'industrie et le commerce ont été fortement opposés à l'approche relationnelle de la gestion des bases de données. ≫.

27. La société Oracle qui "pèse" actuellement des millions de dollards.

28. UML pour *Unified Modelling Language* est un langage formel qui a vocation à servir à la modélisation dans de nombreux domaines dont en particulier l'informatique avec les bases de données et la *Conception Orientée Objet* (*COO*) dont ce langage (qui est en réalité une méthode) est issu.

Remarquons également que les concepts issus de la conception orientée objet sont également incontournables dans le traitement de l'information, les approches de la classification. Citons, en particulier, *l'héritage*, qui concerne le difficile problème du *partage* d'information, i.e. le fait qu'une même information puisse être utilisée par différents acteurs : attributs, processus, systèmes, utilisateurs, ainsi que la notion de *programmation évènementielle* : (une valeur pour un programme, une certaine information dans une base de donnée déclenche l'exécution d'un programme).

Signalons enfin, l'existence d'un autre modèle théorique pour les bases de données, le *modèle déductif*. On parle encore de *bases de données déductives*. Ce modèle est également fondamental car il étend le modèle relationnel de Codd au calcul des prédicats en rajoutant de l'intentionnalité (de l'abstraction) au modèle de Codd, en y intégrant *in extenso* les variable du premier ordre. Le langage de requêtes des BD bases de données déductives s'appelle *Datalog*. Noter qu'il en existe des implémentations qui fonctionnent plutôt bien, mais qui malheureusement ne se sont pas vraiment répandues en dehors des laboratoires de recherche. On notera qu'il n'existe pas à l'heure actuelle de "vrais" SGBD déductifs, c'est-à dire avec toutes les potentialités qu'offrent les SGBD relationnels. Ce qui est plutôt surprenant vu l'importance actuelle des systèmes d'information avec le Web.

On peut s'étonner – à juste titre – que divers modèles théoriques, tous aussi fondamentaux les uns que les autres, coexistent, sans réelle intégration. Les bases de données constituent une discipline, en fin de compte, récente, et probablement ces divers modèles théoriques convergereront dans l'avenir. C'est ce à quoi nous nous employons avec un travail en cours sur les bases de données conceptuelles [29], la logique apportant un cadre théorique fondateur. Considérons le problème général de la classification d'information. On peut dire que les bases de données, comme décrites ci-dessus dans les divers modèles théoriques constituent une *approche formelle* de cette question. Notamment avec la formalisation de la notion de requête qui devient alors une notion mathématique (et implémentée) bien plus sophistiquée que la simple utilisation de mots-clefs dont elle est tout simplement une généralisation [30].

Dans les bases de données – et ceci quelque soit le modèle théorique utilisé – on utilise en amont, une notion fondamentale, la notion *d'attribut* (que l'on peut voir comme des mots-clefs formels) et diverses sortes de regroupements ensemblistes portant sur les attributs de manière à constituer le *schéma relationnel* d'une base de données qui est la *classification formelle* recherchée des informations dont on est parti. Ceci concerne la partie structurelle (la *morphologie*) des bases de donnés. Une base de donnée proprement dite, est constituée alors de

29. Ferbus-Zanda M. (In preparation). *Logic and Information System : Relational and Conceptual Databases*. In preparation.

30. Nous devrions plutôt dire que la notion de mot-clef – généralisée avec le Web et l'utilisation des moteurs de recherche – est une particularisation des requêtes des bases de données, celles-ci étant largement antérieures à l'émergence du Web (dans les années 90).

tables (dans le modèle relationnel), dont les noms des colonnes sont des attributs pour la BD. Une ligne d'une table décrit une *entité* (du monde réel) : cette entité est alors représentée par les valeurs (de la ligne) pour chacun des attributs de la table (i.e. les noms des colonnes). Les tables d'une base de donnée sont reliées entre elles par des sortes de *pointeurs*, selon un "diagramme" qui repose sur le schéma relationnel choisi pour la base.

Le contenu des diverses tables consitue la *sémantique* (autrement dit le *contenu*) de la base à un instant donné. Chaque table, qui comporte en général plusieurs colonnes, peut alors être également vue comme un ensemble de lignes (les "tuples"). Le nombre de colonnes est fixe à l'avance alors que l'ensemble des lignes évolue au cours du temps. Chaque ligne est un ensemble de valeurs [31] d'attribut – une valeur par attribut de la table – qui est, rappelons-le, le nom d'une colonne de cette table. Cette notion de ligne correspond exactement à la notion de fiche dans les anciens fichiers physiques (ceux servant autrefois, par exemple, à la gestion d'une bibliothèque) ou encore à l'information contenue dans une carte perforée (mécanographie).

Par exemple, on pourra avoir une table concernant les auteurs correspondants aux livres que contient une médiathèque. Cette table pourra comporter, en particulier, les attributs : `NomAuteur`, `PrenomAuteur`, `PaysOrigine`, `AuteurPeriode` et par conséquent, la colonne correspondant à l'attribut `NomAuteur` comportera des noms d'auteurs (par exemple : `Duras`, `Sarraute`,`Yourcenar`, `Nothomb`, `Japp`, etc.). Une ligne d'une telle BD sera par exemple, si on se limite à ces 4 attributs : { `NomAuteur.Duras` , `PrenomAuteur.Marguerite` , `PaysOrigine.France` , `AuteurPeriode.XXème siècle` }, ou encore par le 4-uplet : (`Duras` , `Marguerite` , `France`, `XXème siècle`) – l'ordre contenu dans cet uplet dispensant alors de "répéter" les attributs devant chacune de leur valeur.

On accède alors à ce contenu par la notion de *requête*. Ce contenu évolue dans le temps lors des *mises à jour* des informations de la base comportant des *ajouts, suppressions*, ou *modifications d'informations*. On peut se représenter une base de données (sans les requêtes) un peu comme un ensemble de feuilles de calcul d'un tableur (comme *Excel*) qui seraient reliées entre elles par des liens formels (qui peuvent être alors gérées par les requêtes, ce que ne fait pas Excel, ou du moins pas simplement).

Les thésaurus (cf. note 17) sont en fait des bases de données. Le schéma relationel d'une telle BD est la structuration du thésaurus considéré, autrement dit, l'agencement, l'architecture du thésaurus. Le diagramme de la BD associée (et qui est donc une représentation graphique du schéma relationnel de

31. Les lignes d'une table dans une BD sont la plupart du temps présentées commes des uplets, ce qui est une erreur conceptuelle, car dans une table du modèle relationnel de Codd, il n' y a pas d'ordre entre les lignes, ni entre les colonnes. Codd est formel sur ce point qui a une importance d'un point de vue aussi bien conceptuel que pratique : les requêtes doivent être exprimées comme des conditions (i.e. des formules) de l'algèbre relationnelle, utilisant des noms d'attributs et des tables. Cela veut dire, par exemple, qu'on ne peut pas chercher dans une table, au moyen d'une requête la première, la deuxième, etc. ligne (ou colonne) de la table.

la base) rend compte, formellement de cette architecture. On voit bien qu'il peut y avoir plusieurs tables en jeu : par exemple, dans un thésaurus consacré à l'épistémologie des mathématiques, on pourra avoir une table consacrée à la logique mathématique, une autre aux probabilités, une à l'algèbre, une à la topologie, une à la géométrie, une à l'analyse fonctionnelle, au calcul différentiel et intégral, etc. ainsi que des tables consacrées à l'histoire des mathématiques, d'autres tables aux symboles utilisés, à la terminologie employée, d'autres tables consacrées aux mathématiciens, et avec les concepts qu'ils ont introduits, des tables consacrées aux philosophes, aux historiens des mathématiques, etc.

Bien entendu ce choix des différentes tables est complètement subjectif et on peut orienter la BD d'une toute autre façon, en l'axant par exemple sur *la synonymie, la quasi-synonymie, la connexité, l'analogie, la comparaison, la dualité, l'opposition*, etc. relatives aus différents mots du thésaurus. L'organisation interne d'une table donnée (le choix des colonnes, i.e. des attributs) dépend plus ou moins de ce que l'on veut faire avec ce thésaurus et des choix que l'on a fait pour les différentes tables. Les contenus des tables sont alors constitués de tous les mots choisis pour le thésaurus.

Si le thésaurus est complété avec des définitions, autrement dit, si le thésaurus n'est pas juste l'équivalent d'un dictionnaire hiérarchisé de synonymes, d'associations, etc. i.e. une structuration de mots-clefs, alors ces définitions sont rajoutées dans le contenu des tables de la BD avec des colonnes spécifiques. Mais ce qu'il faut bien voir, c'est que le schéma de la BD associée au thésaurus repose pour l'essentiel sur la partie "associative" du thésaurus (son graphe en fait) et non sur sa partie "définitionnelle". On remarquera également que c'est grâce à l'informatique et aux bases de données que des thésaurus complets peuvent être élaborés et utilisés : il n'est en effet pas envisageable de réaliser une version "papier", lisible, d'un dictionnaire qui soit à la fois un dictionnaire des synonymes et un dictionnaire classique, avec des définitions [32], alors qu'avec une bonne interface graphique, cela devient réalisable.

On notera que ce qui n'est pas représenté sous forme de table pourra être récupéré grâce à une requête. Par exemple, Si on opte pour la première forme de structuration (par discipline), tous les synonymes d'un mot donné du thésaurus pourront être récupérés, et ceci, indépendemment de leur appartenance à une discipline donnée, répertoriée en tant que table dans la BD du thésaurus. Ceci montre bien que le choix de la forme de structuration que l'on prend pour la base de donnée n'est pas du tout rédhibitoire sur l'usage que l'on fera par

32. Remarquer que de tels dictionnaires "complets" sont en réalité circulaires : le mot a est défini à partir du mot b, lui même défini défini à partir d'autres mots définis dans le dictionnaire. Seule une connaissance *externe* au dictionnaire, permet de réellement pouvoir appréhender le "sens" des mots. Noter qu'il s'agit d'une incomplétude plus ou moins "cachée". Par contre dans un dictionnaire de synonymes, la structuration repose fondamentalement sur les définitions circulaires – avec tout de même la réserve, pour les versions papier, qu'un mot donné, synonyme d'un autre mot, et référencé comme tel, ne comportera comme entrée, qu'une référence au mot dont il est synonyme. Par contre dans les versions électroniques, cette "redondance" n'est pas un problème : tout cela peut être géré, par exemple, avec des pointeurs.

la suite de la BD, puisqu'on pourra toujours récupérer les informations que l'on désire voir regroupées par des requêtes appropriées. Ceci est d'ailleurs en général complètement "caché" à l'utilisateur qui ignore quelle est l'organisation interne de la BD qu'il utilise. En général, on choisit une structuration qui facilite l'élaboration du schéma de la BD ou encore une structuration *optimisée* , c'est-à dire qui soit efficace pour l'exécution des requêtes (une BD peut contenir des millions de lignes dans les tables). Bien entendu, la synonymie dont il s'agit est relative au monde clos du thésaurus formalisé en BD.

Lorsqu'on exécute une requête dans les bases de données relationnelles, on obtient une *vue* qui est structurée comme une table de la BD, et qui n'en diffère que par le fait que les vues sont stockées en *mémoire vive* – la RAM (Random Access Memory), qui est une mémoire "volatile" en ce sens, qu'elle est effacée lors de l'extinction de l'ordinateur – alors que les tables "réelles" de la BD et qui représentent des données *persistentes* sont stockées dans une *mémoire de masse* (disque dur, etc.). Évidemment rien n'empêche de sauvegarder une vue.

Une remarque importante : on voit bien avec cet exemple, émerger la notion de BD de BD. En effet, on peut également constituer sur le même modèle, une base de données consacrée à l'épistémologie de l'informatique, une à l'épistémologie de la physique, même chose pour la chimie, la biologie, etc. et réunir ces différentes bases de données en une unique BD et avoir ainsi un thésaurus consacré à l'épistémologie. Et rien n'empêche de réunir d'autres disciplines à l'épistémologie. On voit alors que se pose le problème de savoir *à quel niveau d'abstraction et de raffinement, on décide de se placer* pour l'écriture du thésaurus ou de façon plus générale pour la réalisation d'une base de donnée et également quelques sont les limites du sujet que l'on traite... C'est là un des problèmes les plus difficiles de la modélisation et qui sous-tend d'ailleurs toute l'activité scientifique.

Une autre remarque s'impose avec cet exemple : nous avons peu parlé de "l'objet" dans le cadre de cet article. Il est clair qu'avec un tel exemple, on se rend compte que le côté *d'hiérachique* d'un thésaurus repose sur le concept *d'héritage* de la COO (évoquée plus haut), et qu'il est donc indispensable de rajouter certains concepts de l'approche objet dans le modèle relationnel de Codd [33], ce que nous tentons de faire avec les bases de données conceptuelles [34].

Pour en revenir au cas général, on remarque que *la notion de requête dans les bases de données, et qui est indissociable de la structuration d'une base de données avec le schéma relationnel qui lui est associé*, correspond à la notion de requête de Google, à ceci près – et la différence est de taille – que les

33. Codd était radicalement contre l'intégration de l'approche objet avec les bases de données relationnelles (en fait, dans la *première forme normale*, du modèle relationnel des bases de données, Codd interdit la décomposition des attributs en liste et en arbre – et qui correspond à la notion d'héritage de la COO). Ceci peut se comprendre à l'époque où il a élaboré son modèle, car l'approche objet est particulièrement déstructurante, et Codd était justement dans une démarche de structuration. Mais Codd était également contre le modèle Entité/Association de Chen (personne n'est parfait) !

34. *Ibid.* Note 29.

requêtes des bases de données relationnelles s'écrivent dans un langage de programmation bien plus sophistiqué que ne le sont les conjonctions de mots clefs proposées dans les requêtes pour Google. Le langage de programmation qui sert à l'écriture des requêtes s'appelle *SQL* (*Structured Query Language*) dans les implémentations [35] du modèle relationnel de Codd.

On notera que comme dans le cas des mots-clefs, le choix des attributs et des divers regroupements d'attributs pour une base de donnée est parfaitement subjectif : c'est de la *sémantique*, sémantique dont la formalisation se traduit par un schéma relationnel donné. Toutefois, une fois que ce choix est fait et que le schéma relationnel est posé, alors la forme des requêtes est en partie imposée par ce schéma – mais on peut bien sû choisir de demander ce que l'on désire. On a en donné une idée avec l'exemple du thésaurus. En ce qui concerne le Web, l'élaboration d'un tel schéma est rigoureusement impossible compte tenu de sa nature fondamentalement dynamique.

On notera qu'avec les bases de données, on a à tout moment, une idée très précise de la structuration sur laquelle on travaille (c'est un objet mathématique) et l'extraction d'information à partir de cette structuration se fait de façon rigoureuse, au moyen de la notion formelle de requête.

Signalons alors que les résultats obtenus à l'issue d'une requête sont *exhaustifs* relativement à la base de données considérée : on n'a *ni plus ni moins* que les objets de la base qui vont satisfaire la requête. Les informations contenues dans une base de données (bien formalisée) sont d'ailleurs parfaitement connues à un instant donné et les mises à jour de la base (ajout, suppression ou modification d'informations) sont totalement contrôlées. Ceci n'est évidemment pas la situation du Web avec un moteur de recherche pour extraire des informations et ce n'est pas non plus le cas des grandes banques de données (en biologie, médecine, cartographie, etc.) dont ni la structuration ni les requêtes ne sont fondées sur des bases mathématiques solides comme le sont les bases de données. Les banques de données sont en réalité des bases de données plus ou moins bien (ou plus ou moins mal) formalisées, autrement dit, elles pourraient être des bases de données à part entière alors que c'est intrinsèquement impossible pour le Web.

4.3 Bases de données et dualité bottom-up versus top-down

Situons maintenant l'élaboration et l'utilisation des bases de données dans l'optique des approches bottom-up et top-down. Comme nous allons le voir, on se trouve dans une situation tout-à fait analogue à la situation décrite plus haut pour les mots-clefs et les requêtes Google.

1) En ce qui concerne l'élaboration du schéma d'une base de donnée relation-

35. Une implémentation du modèle relationnel des bases de données de Codd consiste à écrire un *SGBD* (*Système de Gestion de Bases de Données*). Un SGBD comporte, en particulier, un interpréteur du langage SQL qui est une implémentation de l'algèbre relationnelle de Codd et qui constitue le calcul fondamental de ce modèle théorique.

nelle, on procède, au choix, en utilisant un mode opératoire de type bottom-up ou top-down – et on utilisera en général conjointement (en fait, alternativement) ces deux modes. Dans le mode bottom-up, on se sert du (futur) contenu de la BD pour élaborer son schéma (qui servira alors à structurer son contenu) alors que dans le mode top-down, l'élaboration du schéma de la BD se fait sur des considérations extérieures au futur contenu de la BD. On peut avoir l'impression que le mode opératoire de type bottom-up est paradoxal (utiliser le contenu pour construire une structuration du contenu en question). Il n'en n'est rien.

Dans la pratique, pour la construction d'un tel schéma pour une BD donnée, on part d'une idée de schéma que l'on représente graphiquement sur papier (démarche top-down), puis on l'implémente (c'est une sorte de *prototype* qui est très simple à programmer). Dans un second temps, on remplit les tables de la BD avec un peu de contenu (quelques lignes, que l'on appelle un "jeu de données") et on réajuste alors le schéma en conséquence (démarche bottom-up), et on recommence en le complétant à nouveau sur papier... Rappelons que le contenu d'une base de donnée est justement ce qui détermine la sémantique de la base, alors que la construction du schéma de la base est de la morphologie (de la synyaxe). *Avec une telle démarche mixte, on peut ainsi élaborer la partie morphologique (syntaxique) d'une BD en accédant – et de façon alternative – à une partie de la sémantique de la BD.*

Cette démarche n'est donc paradoxale qu'en apparence. La vrai difficulté est en réalité de délimiter le sujet – délimitation dictée par ce que l'on appelle le *cahier des charges* du système d'information à modéliser – et surtout de choisir le niveau d'abstraction/raffinement de chacun des composants (attributs, tables, etc.).

2) Le choix et l'écriture des requêtes arrive dans un deuxième temps et sur un mode tout à fait analogue à ce qui est décrit pour l'élaboration du schéma : bottom-up, top-down et mixte. Pour les BD compliquées, on peut être amené à élaborer schéma et requêtes plus ou moins simultanément. On l'a vu pour le thésaurus.

3) Ce n'est qu'une fois que le schéma d'une BD est définitif et que l'on en a écrit et implémenté les principales requêtes de base (dont certaines seront des vérifications de la cohérence de la base) que l'on peut alors réellement compléter le contenu de la BD. On peut évidemment rajouter autant de requêtes que l'on désire. Mais par contre des modifications, même mineures, du shéma (comme séparer un attribut en deux attributs – par exemple, transformer l'attribut `Artiste` en `Compositeur` et `Interprete` dans une BD consacrée à la musique) peuvent tourner à la catastrophe si la BD est de taille conséquente et déjà bien remplie.

4) *L'appréhension du contenu de la base peut alors se faire dans un mode entièrement top-down, grâce à l'utilisation des requêtes.* C'est en ceci que réside la très grande originalité des bases de données relationnelles. On peut appréhender de grandes masses de données entièrement depuis l'extérieur et ceci d'une façon complètement rigoureuse et mathématique. La notion de requête peut alors

être vue comme une question que l'on pose au SGBD dans lequel l'exécuteur de requêtes fonctionne *de facto* comme un *oracle* (puisque son travail est invisible à l'utilisateur). Bien entendu, rien n'empêche, si on le désire, d'adopter une démarche de type bottom-up pour parcourir le contenu d'une BD et pour y trouver les informations que l'on cherche. Il faut voir qu'avant le modèle relationnel de Codd, c'était en général la seule facon (en dehors de la mécanographie) que l'on avait de s'y prendre, avec les anciens "fichiers" physiques, i.e. les boites de fiches utilisées, par exemple, dans les bibliothèques d'une certaine importance et dont le classement était un vrai casse-tête dès que l'on voulait abandonner les tris alphabétiques (syntaxiques) pour accéder à des tris thématiques (sémantiques) !

4.4 Classification et dualité bottom-up versus top-down

Résumons ce qui précède. Une approche de la classification au moyen de mots-clefs, ou des requêtes de Google (comme avec la Google classification) ou encore avec les bases de données (et ceci, quelque soit le modèle théorique utilisé) sont toutes intrinsèquement de même nature : *Dans les différentes étapes d'élaboration et en particulier pour les mots-clefs et les requêtes, on peut adopter aussi bien un mode opératoire de type bottom-up que de type top-down* (et en général les deux modes sont utilisés alternativement dans un mode *mixte*). *Les requêtes ainsi constituées permettent alors d'appréhender des ensembles de textes dans un mode de type top-down (c'est-à dire sans avoir à comprendre la signification des textes), et d'en faire alors une classification.*

Dans l'approche de la classification par compression, on est dans un mode entièrement de type top-down. Noter que lorsqu'on procède ainsi à une classification par compression de textes, le cadre est alors *purement syntaxique*, on n'utilise même plus de mots-clefs ni de requêtes, qui véhiculent de fait une certaine sémantique (le choix des identificateurs utilisés reflète d'ailleurs cette sémantique). *On obtient alors de l'information relative aux textes, sans recourir à leur sémantique : juste en les compressant et donc par un calcul.*

A première vue cette approche peut tenir du "miracle" : on arrive à classifier des informations représentées par des textes sans rentrer dans le contenu des textes en question et sans avoir eu à les comprendre à aucun moment alors que clairement dans les méthodes précédentes, on est plus ou moins amené à utiliser un mode bottom-up (mais ce n'est pas non plus une obligation) pour élaborer des requêtes intéressantes (et dans le cas des bases de données pour élaborer le schéma de la base). Rappelons, comme nous l'avons évoqué plus haut, que la compression de textes est une science éminemment théorique et qu'un simple algorithme comme "gzip", d'usage courant et banalisé, a nécessité des dizaines d'années de recherches. On notera toutefois que dans l'utilisation de la méthode de classification par compression, on ne choisit pas non plus les textes complètement au hasard ! Toutefois, on ne voit pas bien ce qui pourrait limiter à l'avenir l'utilisation de cette méthode pour toutes les informations qui résident sur le Web.

Si l'on considère le problème de la classification d'information en général, noter

l'exception que constituent les *statistiques* dont la démarche est une approche formelle de la classification en général de type top-down, par exemple, en utilisant des méthodes calculant des *facteurs de corrélation* pour regrouper des objets et/ou des informations et ainsi en proposer une structuration. D'ailleurs Google et les algorithmes de compression utilisent de façon massive les statistiques. Toutefois, on peut aussi procéder en statistiques de façon bottom-up ou encore mixer les deux types d'approches. On verra mieux cet aspect des choses ci-dessous où nous proposons une version probabiliste du schéma de compréhension (cf. section 5.2).

5 Interprétation ensembliste de la dualité Bottom-Up versus Top-Down

Nous allons maintenant replacer les différentes approches de la classification envisagées dans cet article dans la perpective du schéma de compréhension de la théorie axiomatique des ensembles de Zermelo-Fraenkel, **ZF**, théorie qui peut être vue d'une certaine façon comme une des premières tentatives mathématiques formelles d'approche de la classification, le regroupement ensembliste étant la forme la plus rudimentaire de regroupement d'éléments. Noter au passage que le modèle relationnel de Codd pour les bases de données repose en grande partie sur la théorie (naïve) des ensembles, ce qui vu sous l'angle de la recherche d'un mode de structuration formelle n'est finalement pas vraiment étonnant.

Ainsi, la *dualité bottom-up versus top-down* que nous mettons en évidence dans la classification (cf. section 4), peut être illustrée par la façon dont "fonctionne" le schéma de compréhension ensembliste. Par ailleurs, nous présentons une version probabiliste du schéma de compréhension qui permet d'illustrer, entre autres, la dualité *exact versus approché*.

5.1 Le schéma de compréhension ensembliste

C'est l'approche des mathématiques "pures".

Il s'agit d'une approche globale, intrinsèquement déterministe qui s'articule autour d'une dichotomie fondamentale :

Vrai/Faux,
Prouvable/Contradictoire.

Une quête de l'absolu, de la perfection basée sur la *certitude*. Ceci est reflété dans le schéma de compréhension ensembliste classique

$$\forall x\ \exists y \quad y = \{z \in x\ ;\ \mathcal{P}(z)\}$$ [36]

où $\mathcal{P}$ est une *propriété connue, fixée à l'avance*. Le regroupement ensembliste est donc effectué à partir d'une propriété bien définie au sein de cette dichotomie.

36. Plus formellement : $\forall x\ \exists y\ \ \forall z\ \ (z \in y \iff (z \in x\ \wedge\ \mathcal{P}(z)))$.

Pour effectuer un tel regroupement, et construire ainsi un tel ensemble, on se se retrouve alors dans un mode opératoire de type top-down, l'ensemble étant construit à partir de la propriété $\mathcal{P}$.
Et plus précisément, si on adopte une démarche constructiviste :

- On part d'un certain ensemble x.
- On choisit alors une certaine propriété $\mathcal{P}$ relative aux éléments de l'esnemble x. Ceci peut être réalisé de façon bottom-up ou top-down, exactement comme dans le choix des mots-clefs en vue d'une requête, ou comme dans l'élaboration d'une requête d'une base de donnée relationnelle (cf. section 4.3). De la même façon, l'idée du regroupement ou si l'on veut le *choix* de ce regroupement (et qui est formalisé par la propriété $\mathcal{P}$) est en général parfaitement subjectif : c'est de la *sémantique*. On peut toutefois déterminer une telle propiété $\mathcal{P}$ de façon syntaxique : par un calcul (cf. section 6.1).
- A partir de cette propriété $\mathcal{P}$, on sélectionne alors les éléments de l'ensemble x qui vont satisfaire cette propriété $\mathcal{P}$.

Le shéma de compréhension[37] nous autorise à construire ainsi un tel ensemble (dans la théorie axiomatique des ensembles ZF).

On remarque que la solution au paradoxe de Russel[38] – sur lequel on "tombe" effectivement si on ne relativise pas les ensembles z à un certain ensemble x de départ, ou de façon équivalente, si l'on considère un ensemble de tous les ensembles – trouve ici pleinement son sens : dans cette démarche il faut bien partir de quelque chose, et en fin de compte d'un certain ensemble d'objets, pour élaborer une telle propriété ! D'ailleurs si l'élaboration de la propriété $\mathcal{P}$ se fait dans un mode mixte, comme c'est possible de le faire pour les requêtes des bases de données relationnelles, on pourra en fait, partir d'une certaine idée pour la propriété $\mathcal{P}$ (basée évidemment sur ce qu'est l'ensemble x), puis "piocher" quelques éléments dans l'ensemble x considéré, pour commencer à contruire la propriété $\mathcal{P}$, et recommencer à piocher dans l'ensemble x et ajuster alors $\mathcal{P}$ et ainsi de suite.

Une fois que l'on a "mis au point" une telle propriété (mais que l'on peut également trouver *in extenso*), on peut alors faire le regroupement de tous les éléments de x qui vont satisfaire $\mathcal{P}$. Bien entendu, dans la littérature mathématique, on ne va jamais présenter une propriété sous une telle forme tâtonannte. On donnera directement le résultat ! C'est pourtant bien la façon dont les choses s'élaborent en général, et les informaticiens y sont coutumiers : la mise au

37. Noter que l'on peut imaginer de mettre un certain nombre de restrictions sur la propriété $\mathcal{P}$, en particulier pour éviter les phénomènes de circularité, par exemple, si la propriété $\mathcal{P}$ contient une quantification sur l'ensemble y qu'elle est censé définir. On parle alors *d'imprédicativité* (Henri Poincarré).

38. Si pour le schéma de compréhension, on pose ($\mathcal{P}$ étant une certaine propriété fixée) : $\exists y \quad y = \{\ z\ ;\ \mathcal{P}(z)\ \}$, i.e. $\exists y \ \ \forall z \ \ (z \in y \quad \longleftrightarrow \quad \mathcal{P}(z))$, alors il suffit de prendre pour la propriété $\mathcal{P} : \mathcal{P}(u)$ si et seulement si $u \notin u$ pour obtenir une contradiction (on aura $y \in y$ si et seulement si $y \notin y$).

point d'une BD ou d'un programme se fait ainsi de façon modulaire, mais c'est également bien souvent le cas des mathématiques.

Ce qui est important de voir c'est qu'alors le regroupement, autrement dit la constitution (et pour certains la construction effective) de l'ensemble y peut se faire dans *un mode opératoire de type top-down, qui est en fait un mode intentionnel. L'intentionnalité, ou si l'on veut l'abstraction étant exprimée par la propriété* $\mathcal{P}$. Cette propriété joue le rôle d'une question qui se traduit par une instanciation du schéma de compréhension, lequel fonctionne donc comme un *oracle*. La réponse de l'oracle est précisément l'ensemble des éléments de x qui vont satisfaire $\mathcal{P}$. Ce mode opératoire est en opposition (dual en fait) à une *description extensionnelle* de y (c'est-à dire élément par élément) et qui s'effectue donc dans un mode bottom-up.

Noter que, le fait de connaître à l'avance cette propriété est très particulier et ne reflète pas la plupart des situations "réelles" . Nous développons ci-dessous cet aspect des choses en proposant un schéma de compréhension "probabiliste". Nous montrons alors dans la section 6, en quoi ce shéma probabiliste peut lui-même être généralisé au moyen de la complexité de Kolmogorov. Ceci nous ramène à la relation entre la théorie algorithmique de l'information et la classification qui sont au coeur de ce travail.

5.2 Le schéma de compréhension probabiliste

Dans l'approche probabiliste, nettement plus pragmatique que l'approche logique, *l'incertitude* est prise en considération, elle est bornée et est traitée de façon mathématique [39].

On peut représenter ceci avec une version probabiliste du schéma de compréhension en remplaçant le fait d'être vrai pour la propriété $\mathcal{P}(z)$ pour certaines instances de z, par une *limitation du degré d'incertitude* que l'on a sur la vérité de $\mathcal{P}(z)$. Et de façon formelle, en même temps que z, on est amené à considérer un nouveau paramètre pour $\mathcal{P}$, à savoir l'évènement ω d'un espace de probabilités Ω et on doit alors fixer un intervalle de confiance I de $[0, 1]$ (qui représente un certain intervalle de prédiction). Dénotons par μ la loi de probabilité sur Ω, le schéma de compréhension probabiliste peut alors s'énoncer sous la forme :

$$\forall x \; \exists y \quad y = \{z \in x \; ; \; \mu(\; \{\omega \in \Omega \; ; \; \mathcal{P}(z, \omega)\}\;) \in I\}$$

De la même façon que dans le cas du schéma de compréhension ensembliste, *on se retouve dans un mode opératoire de type top-down pour effectuer un tel regroupement et construire ainsi un tel ensemble* à partir de la propriété $\mathcal{P}$ et de l'intervalle I. Et ceci, même si l'on prend en compte un certain degré d'incertitude sur la vérité ou le caractère prouvable de la propriété $\mathcal{P}(z)$ (que l'on remplace par $\mu(\{\omega \in \; \Omega \; ; \; \mathcal{P}(z, \omega)\}) \in I$) pour des intances particulières de z.

39. Le lecteur peut se référer à William Feller [21] ainsi qu'à Kolmogorov [27, 29] et Chaitin [4].

On notera qu'une fois encore, on se retrouve dans une situation bien particulière car cette propriété, même si sa vérité est plus ou moins incertaine, est bien *définie et fixée à l'avance ainsi que l'intervalle de confiance* I. On se rapproche toutefois davantage avec ce schéma, de nombre de situations rencontrées dans le monde réel. Comme dans le cas précédent, une telle propriété $\mathcal{P}$ (et l'intervalle I) permettent *de constituer l'ensemble* y *dans un mode opératoire de type top-down, autrement dit, ils fournissent une description intentionnelle, abstraite de l'ensemble* y. On voit bien qu'il est alors naturel de considérer comme plus haut, un *oracle* sous-jacent (le schéma de compréhension probabiliste) qui, si on lui soumet la propriété $\mathcal{P}$ et l'intervalle I, renvoit l'ensemble y ; cette réponse n'étant pas parfaitement déterminée, l'intervalle I bornant cette incertitude. Noter que comme ci-dessus, le *choix* de $\mathcal{P}$ et I est d'ordre *sémantique*. Remarquer que d'autres versions pour un schéma de compréhension probabiliste peuvent être considérées.

En ce qui concerne les regroupements d'informations qui relèvent d'un mode purement top-down, à la fois dans l'élaboration de propriétés qui permettent le regroupement d'informations, ainsi que dans le regroupement proprement dit, et donc dans la constitution d'ensembles de telles informations, on pourra se reporter à la section suivante sur l'intentionnalité et la complexité de Kolmogorov (section 6).

Rappelons simplement (cf. section 4.4) que la classification par compression et un certain nombre de méthodes d'inférence statistique permettent d'avoir une telle approche entièrement de type top-down. Le cas de la classification avec Google est exactement le même que pour les schémas de compréhension ensembliste et probabiliste (les mots-clefs proposés dans les requêtes de Google vont jouer le rôle de la propriété $\mathcal{P}$), ou encore avec la classification au moyen des bases de données à ceci près – et la différence est de taille – *qu'avec Google, tout est en continuel état de variation : les réponses, tout comme les mots-clefs proposés dans les requêtes.*

6 Information, intentionnalité, abstraction et complexité de Kolmogorov

6.1 Classification, bases de données, intentionnalité, abstraction, sémantique et théorie algorithmique de l'information

Nous avons souligné dans la section 4 l'importance de l'expansion du Web et l'immense intérêt que représentent la classification par compression et la classification avec Google. On peut d'ailleurs voir le Web comme une sorte de *système expert* phénoménal, dans la mesure où c'est un immense système d'information (et ceci est l'aspect réseau – logiciel et matériel – entre des machines et des serveurs), mais également avec le constat que les machines en question sont utilisées – et programmées – par des humains (et leur cerveau) avec un niveau

d'intelligence incomparable avec le monde syntaxique des machines (qui ne font donc que calculer...).

L'utilisation de la classification par compression (et la classification avec Google) seront très vraissemblablement utilisées avec l'information circulant sur le Web et cela concerne également une part des méthodes d'inférence statistiques. Remarquons que toutes ces approches sont étroitement corrélées : comme dans toute approche de la classification (cf. section 4.4 et section 5), elles permettent une *appréhension top-down de l'information*. En particulier, *elles peuvent servir à l'appréhension du contenu en information d'un texte donné (et plus généralement d'un ensemble de textes) et ceci sans accéder au contenu depuis l'intérieur du texte, i.e. sans lire et comprendre le texte*. Ces méthodes procèdent par analogie avec d'autres textes dont la signification est connue ou encore par comparaison de leurs contenu en information respectifs. Ce sont en quelques sorte des "profilers" qui peuvent s'avérer, si on les applique à l'information circulant sur le Web, (redoutablement) efficaces dans l'avenir [40].

Toutefois, nous avons également précisé en quoi ces méthodes ne sont pas suffisamment développées formellement, en particulier en ce qui concerne la notion de requête : si on a une classification d'information, l'idée est de pouvoir récupérer de l'information (et ceci de façon rigoureuse, formelle) à partir de cette classification. On peut d'ailleurs noter que la notion de requête concernant le Web (avec Google ou tout autre moteur de recherche) n'est pas non plus vraiment formalisée.

Nous avons vu qu'avec le modèle relationnel des bases de données de Codd, la structuration et le traitement de l'information contenue dans des fichiers d'ordinateur, pouvaient être appréhendées de façon entièrement mathématique, à la fois dans le schéma d'une BD et dans les requêtes que la BD propose (les requêtes dépendant étroitement du schéma). Comme nous l'avons remarqué, avant Codd, personne n'a envisagé un tel traitement de l'information dans les machines et Codd a du se battre pour faire accepter un tel modèle mathématique et d'ailleurs les systèmes d'exploitation actuels des machines n'utilisent toujours pas réellement les bases de données. Il nous apparaît donc fondamental de réfléchir à des formalisations possibles pour la classification par compression et la classification avec Google, ainsi qu'à la formalisation de la notion de requête sur le Web. Remarquons qu'avec Google (ou avec tout autre moteur de recherche du Web), on n'a aucune idée pour évaluer de façon générale le degré d'incertitude des réponses fournies. Google donne entre 0% et 100% de réponses "pertinentes". Ces réponses sont *imprévisibles* et en *continuel état de variation*. Situation pour le moins pas évidente ! Toutefois, il nous apparaît raisonnable, dans un premier temps, de faire abstraction du caractère plus ou moins fluctuant de Google (ainsi que de son côté pas nécessairement scientifique, cf. section 3.2, point 4), dans l'appréhension mathématique de ces méthodes.

Ainsi, on part d'un regroupement ou plus généralement d'une classification

40. Est-il besoin de rappeler qu'une fois qu'une information a été rentrée sur la "Toile", il est quasiment impossible de l'effacer...

que l'on a obtenus au moyen de conjonctions de mots-clefs proposés dans des requêtes soumises à Google, ou encore d'un regroupement ou d'une classification obtenus par compression ou observé par des méthodes statistiques.

Dans le cas simple d'un regroupement, on en déduit alors l'existence d'une propriété, d'une "loi", qui est une forme de régularité. L'émergence d'une telle loi coïncide avec l'existence d'un certain degré d'intentionnalité *dans le regroupement effectué. Autrement dit, on a mis en évidence un regroupement d'objets, dont la description peut être compressée au moyen de la propriété en question. Il s'agira alors d'une description intentionnelle (lorsque la compression a été effectuée). On peut voir ceci comme une forme (élargie) top-down du schéma de compréhension ensembliste ou probabiliste : la propriété utilisée dans les regroupements ensemblistes n'est pas connue et fixée à l'avance.*

Pour des classifications plus sophistiquées, on aura des *regroupements d'ordre supérieur* i.e. des regroupements de regroupements, etc. Autrement dit, plusieurs propriétés seront en jeu (selon les situations, ce nombre peut même être infini, en théorie du moins). Noter qu'avec une analyse fine de la modélisation au moyen des bases de données relationnelles, on s'aperçoit qu'actuellement, dans la plupart des cas, quelques niveaux suffisent pour modéliser nombre de systèmes d'information discrets (du monde réel). On peut donc imaginer une situation analogue pour les classifications obtenues dans les approches de type top-down comme celles évoquées ci-dessus ou du moins celles qui sont relatives au monde réel actuel.

Dans le cas d'un regroupement aléatoire, la description du regroupement en question ne correspond à aucune loi : aucune classification n'est possible. *Seule une description extensionnelle (élément par élément) du regroupement peut être donnée : elle est intrinsèquement non intentionnelle. On peut dire non "intentionalisable", autrement dit, il n'existe pas de description plus courte et plus abstraite et donc plus intentionnelle qui soit équivalente.* Ou pour le dire autrement, une telle description est incompressible.

Ceci montre à quel point la théorie de la complexité de Kolmogorov est une théorie d'avant-garde, lorsqu'on la considère avec plusieurs points de vue, c'est-à dire en étudiant le caractère aléatoire d'un mot aussi bien que son contenu en information, ou encore la possibilité de compression de ce mot. D'une certaine façon, l'aléatoirité est le "contraire" de la classification, et plus précisément, on a une dualité : *aléatoirité versus classification*, du fait de l'existence même de la théorie de la complexité algorithmique de Kolmogorov qui permet d'appréhender l'information sous ces deux aspects, comme l'explicite d'emblée Kolmogorov lui-même [28].

On notera que cette dualité est une quasi opposition, bien que l'aléatoirité ne soit pas non plus le *chaos* (cf. Partie I). Ceci nous fait alors entrevoir des liens profonds entre la complexité de Kolmogorov et les bases de données relationnelles qui constituent actuellement, comme on l'a vu, la seule approche logique qui soit implémentée (et largement diffusée) des systèmes d'information. Cette complexité apparaît également incontournable dès lors que l'on s'intéresse aux

problèmes de classification, ce qui n'est pas surprenant puisque la complexité de Kolmogorov est avant tout une théorie de l'information !

Revenons maintenant sur la démarche même de Kolmogorov et remarquons que celle-ci relève d'une *approche top-down*. En effet il suffit de regarder la définition de base de la complexité de Kolmogorov :

> *La taille du plus court programme qui retourne une output donnée* (l'output étant un mot binaire qui représente un objet donné, fixé à l'avance) [41].

Plus grande est la complexité de Kolmogorov d'un objet, plus grands sont les programmes dont l'exécution retourne cet objet, plus l'objet est aléatoire, plus grand est son contenu en information, plus les programmes qui le produisent sont incompressibles, moins les descriptions d'un tel objet seront intentionelles, moins elles seront abstraites, moins les propriétés qui permettent de décrire cet objet seront abstraites (lorsqu'on les regarde de façon syntaxique).

Dans cette définition on n'entre pas dans le contenu de l'output (dans le détail de l'objet si l'on veut, l'objet est donc pris comme un tout). *On se contente d'appréhender cet objet de l'extérieur, au moyen d'un programme et/ou d'une propriété qui permet de le décrire.* C'est bien une démarche de type top-down tout comme le sont la classification par compression, la classification avec Google avec des mots-clefs choisis au hasard et un certain nombre de méthodes d'inférence statistiques. Celà nous laisse entrevoir que ces méthodes de classification ont à voir entre elles et que la théorie de la complexité de Kolmogorov pourrait fournir un cadre mathématique formel unificateur.

Autrement dit, grâce à le théorie de Kolmogorov, on est capable de mesurer la complexité d'un objet (au sens de Kolmogorov,) i.e. de donner une mesure quantitative du *degré d'intentionnalité* ou encore du *degré d'abstraction*, que peut contenir une description calculable de l'objet. Il est remarquable que ceci puisse être fait sans aucune "connaissance" préalable de la structure de l'objet et que cela nous permette justement d'appréhender cette structure.

6.2 Complexité de Kolmogorov et théories de l'information, sémiotique

Comparons maintenant les différentes façons de considérer le concept d'information selon Shannon (cf. Part I), Kolmogorov, Codd et quelques autres chercheurs :

- Pour Shannon (1948) [32], une information est un *message* qui est transmis sur un support matériel physique, autrement dit, une information est un *signal* dont la transmission peut se faire avec plus ou moins de perte.

41. $K_\varphi(y) = \min\{|p| : \varphi(p) = y\}$ où $K_\varphi : \mathcal{O} \to \mathbb{N}$ avec $\varphi : \{0,1\}^* \to \mathcal{O}$ qui est une fonction récursive partielle (intuitivement φ est un exécuteur pour les programmes p comme l'est un interpréteur *LISP*) et $\mathcal{O}$ un ensemble muni d'une structure de calculabilité. On prend la convention que $\min \emptyset = +\infty$ (cf. Part I).

Noter que dans cette approche, on a affaire à une *conception dynamique de la notion d'information* et que le support de communication est considéré comme étant fondamental.
Shannon s'intéresse alors à la robustesse de cette transmission et est amené à dégager la notion de *quantité d'information* contenue dans les messages transmis. Pour mesurer la diminution ou l'augmentation de cette quantité, Shannon a recours à un concept emprunté à la thermodynamique : la notion *d'entropie* sur laquelle il fonde sa théorie de l'information. Il explicite alors, sur des bases mathématiques, comment traiter de l'information transmise sur des canaux de communication avec un certain bruit. Dans la théorie de Shannon, l'information (le message) est représentée par un mot et est basée sur le codage des lettres ou groupes de lettres des mots (cf. Partie I). C'est donc une analyse purement syntaxique dans un mot et des messages qu'il représentent. Il n'y a pas d'analyse sémantique.
Ainsi, Shannon élabore une théorie mathématique, mesurant quantitativement le contenu en information d'un message transmis avec une certaine perte du signal. Les principales applications de cette théorie (qui présente un immense intérêt) sont reliées aux télécommunications (ce qui n'est pas surprenant : Shannon travaillait dans les laboratoires Bell).

- Les travaux de Shannon prennent leur source dans la cybernétique concue par Wiener- à la fin des années 40 (cf. note 7). Elle a trouvé un plein épanouissement dans le cadre des conférences Macy (1942 – 1953), auxquelles Shannon a assisté. Avant Wiener et ces conférences, il n'existait pour ainsi dire pas de théorie de l'information.
  La cybernétique est une théorie qui fonde en particulier, le concept de *sytème autorégulé*, en termes de : *comportement global*, *d'échanges*, de *communication* et *d'interactions*. C'est une approche fondamentalement top-down de l'information et des systèmes. Wiener parle encore d'une ≪ *science des relations et analogies entre organismes (vivants) et machines*[42] ≫. Il étudie notamment les *processus aléatoires* et le *"bruit"* émis lors des échanges dans un système. Une notion fondamentale de sa théorie est le *feedback* : ≪ Un objet est contrôlé par la marge d'erreur qui le sépare à un moment donné de l'objectif qu'il cherche à atteindre ≫. Ceci préfigure bien la théorie de l'information de Shannon (qui a été un élève de Wiener).
  La vision de Wiener des *machines* est une vision d'avant-garde ! Ses travaux sont à l'origine de nombre de découvertes et en particulier sur les aspects sociologiques, psychologiques et biologiques de la *communication* et de *l'interaction* et plus généralement sur toutes les théories de l'information. En plus de tous les courants de recherche que la théorie de Wiener a engendrés, signalons que cette théorie a également eu une influence im-

42. Le livre que Wiener publia en 1948 et à propos duquel, il eut des échanges avec von Neumann : *Cybernetics or Control and Communication in the Animal and the Machine*, provoqua un certain tohu-bohu...

portante sur une part de la *sémiotique*[43] moderne.

- Citons notamment Umberto Eco[44], dans *l'oeuvre ouverte*[45] (1962) où il est question d'analyser le problème de *l'ouverture* (que l'on peut voir comme une certaine forme de non-déterminisme ou encore de pluralité d'interprétations) des oeuvres d'art. Eco fait largement référence à Wiener dans le chapitre 3 : *Ouverture, Information, Communication*. Il pointe, avec une grande pertinence, la nécessité de distinguer[46] :

  > ≪ [...] la *signification d'un message* et *l'information* qu'il apporte. ≫

  Autrement dit, il convient de bien différencier la *sémantique* d'un message et son *contenu en information*. Eco donne un exemple très simple (que nous réarangeons un peu) et très éclairant, illustrant cette distinction : à savoir, le message "Il neigera demain sur Paris". Ce message n'a pas le même contenu en information selon que l'on se trouve en décembre ou en plein mois d'août !
  Il ajoute :

  > ≪ Wiener disait en somme que signification et information sont des synonymes, liés l'un comme à l'autre à l'entropie et au désordre. [...l'information dépend également de la source dont provient le message. ≫

  Autrement dit, et contrairement à Wiener (et Shannon), Eco met en évidence que le contenu en information d'un message (et d'une certaine façon sa *pertinence*) dépend du *contexte* où le message est considéré. Nous verrons ci-dessous comment Kolmogorov traite de ce problème.

- Pour Kolmogorov (1965) (cf. également Chaitin (1966) et Solomonoff (1964)), l'aspect fondamental de l'information est le *contenu en information* d'un objet donné, et ceci indépendemment de toute considération sur l'utilisation de cette information (comme message par exemple). C'est donc une *conception statique de l'information*.

  Ce qui intéresse Kolmogorov, c'est d'une part, de fonder mathématiquement la notion *d'aléatoirité* et d'autre part, d'expliciter la notion de *contenu en information* d'un objet donné, et ceci, de façon *intrinsèque* à l'objet. Ainsi, ce que cherche Kolmogorov, c'est d'élaborer une théorie de l'information qui soit plus abstraite que celle de Shannon et qui serait basée sur la sémantique et pas seulement sur un objet "physique" comme un mot. Sa solution consiste à considérer des programmes informatiques (vus comme

43. Théorie élaborée par Charles Sanders Pierce (1839 - 1914).

44. Eco, titulaire de la chaire de sémiotique et directeur de l'École Supérieure des Sciences Humaines à l'Université de Bologne, est actuellement professeur émérite. Il a publié de très nombreux essais et des romans où il met en quelque sorte en pratique ses théories sémiologiques et du langage.

45. Eco U. *L'oeuvre ouverte*. Bompiani, 1962 & Le Seuil, 1965.

46. *Ibid*. Note 45.

des descriptions calculables) – il se place en fait dans la théorie de la calculabilité – qui calculent et retournent des objets en output. Il s'intéresse alors à la longueur du plus petit programme. Ainsi, en considérant à la fois, les programmes et ce que les programmes calculent, Kolmogorov prend en compte, dans sa théorie de l'information, à la fois un aspect syntaxique (la longueur du programme) et un aspect sémantique (ce que le programme calcule).

Avec la complexité de Kolmogorov, on a donc une mesure mathématique "objective" du contenu en information d'un objet. De plus cette mesure est définie de façon inhérente à l'objet – et elle en donne alors une spécification de son contenu en information qui est en quelque sorte *universelle* – car elle ne dépend pas (à une constante près) du langage de programmation utilisé pour écrire les programmes : c'est le contenu du *théorème d'invariance de Kolmogorov*. Pour définir ainsi une notion mathématique "absolue" d'aléatoirité, Kolmogorov est donc amené à faire radicalement abstraction du support matériel physique de l'information. C'est ainsi que se trouve élaborée la théorie algorithmique de l'information, permettant de "calculer" [47] la complexité d'un objet. Avec la notion de *complexité conditionnelle*, Kolmogorov raffinera cette notion de complexité intrinsèque à un objet en la *relativisant à un contexte* (*l'input*, *l'oracle, etc.* pour le programme) et qui représente de l'information supplémentaire. Ceci correspond très exactement au problème soulevé par Eco sur la distinction fondamentale entre signification et contenu en information.

Kolmogorov donne ainsi les bases de la *théorie algorithmique de l'information*, qui peut alors être vue aussi bien comme *une théorie fondant la notion d'aléatoirité, que comme une théorie pouvant servir de fondement à la notion de classification, de structuration de l'information.*

- Pour Codd (1970), l'aspect fondamental de l'information réside, comme on l'a vu, dans la *structuration* de cette information et dans la possibilité de *retrouver de l'information de façon exhaustive*, à travers la structuration choisie, celle-ci étant décrite de façon formelle. La théorie de Codd repose fondamentalement sur la logique mathématique. C'est donc sur *l'aspect statique de l'information* que Codd fonde sa recherche. Noter que pour élaborer sa théorie, tout comme Kolmogorov, Codd fait abstraction du support physique de l'information, provoquant une sorte de révolution dans le monde du traitement de l'information (chez IBM), qui jusque là, reposait sur le fait que le support de l'information étaient les *fichiers*. Quelque part, information et fichiers faisaient un tout.
  On notera également que dans la modélisation des systèmes d'information au moyen des bases de données relationnelles, le point subtil soulevé par Eco sur la distinction entre sémantique et contenu en information est

47. Rappelons que l'idée profondément originale de Vitányi, à l'origine de la classification par compression, consiste à calculer une *valeur aprochée* de cette complexité, grâce aux algorithmes de compression de données.

pris très au sérieux : on parle de *pertinence d'une information pour un système d'information donné*. Et c'est même sur cette distinction fondamentale que se construit le schéma relationnel d'une base de données. Par exemple, dans une base de données formalisant la gestion d'une université, on *choisira* ou non de considérer que les loisirs des étudiants sont une information à retenir ou au contraire à éliminer. Ce choix est évidemment parfaitement subjectif. C'est de la *sémantique*. Si l'attribut `LoisirEtudiant` est retenu, alors il fera partie du schéma relationnel de la base de données qui est la *contrepartie syntaxique de ce que l'on retient comme sémantique "constitutionnelle" du système d'information*.

6.3 Théorie algorithmique de l'information, représentation et abstraction

La complexité de Kolmogorov ne s'applique pas à priori aux objets que l'on considère mais seulement aux mots binaires associés dans une représentation que l'on a choisie pour ces objets. Toutefois, pour les différentes représentations usuelles, ceci a une incidence mineure (c'est le contenu du *théorème d'invariance*). Aussi, on parle (abusivement) de la complexité de Kolmogorov d'objets et non de *la complexité de Kolmogorov de représentation d'objets*.

Cependant, si on considère des représentations d'ordre supérieur, ceci ne reste plus vrai. Par exemple, si on représente les entiers comme cardinaux d'ensembles (finis) récursivement énumérables. En fait, la complexité de Kolmogorov permet de comparer les représentations des entiers d'ordre supérieur, conduisant ainsi à une *hiérarchie* propre des sémantiques naturelles des entiers (*itérateurs de Church, cardinaux, ordinaux, etc.*) comme nous le montrons dans [23]. Cette hiérarchie peut être mise en parallèle avec une hiérarchie de complexités de Kolmogorov induite par la prise en considération de calculs infinis et/ou d'oracles.

Ceci nous a permis, entre autres, de dégager le fait que la complexité de Kolmogorov peut servir à établir une forme de classification de différentes sémantiques des entiers, chose assez étonnante. On peut voir aussi cette classification des différentes représentations des entiers comme *une classification du degré d'intentionnalité des représentations en question, i.e. une sorte de classification du caractère plus ou moins abstrait des différentes définitions des entiers, provenant des différentes sémantiques considérées*.
Nous développons une part des aspects techniques de ces considérations dans [23] [48].

7 Conclusion

Les considérations précédentes montrent, en particulier, que non seulement la complexité de Kolmogorov permet de fonder la notion d'aléatoirité mais de plus,

48. Ainsi que dans un travail en cours : Ferbus-Zanda M. & Grigorieff S. *Kolmogorov complexity and higher order set theoretical representations of integers* et Ferbus-Zanda M. & Grigorieff S. *Infinite computations, Kolmogorov complexity and base dependency*.

cette théorie est intrinsèquement liée à tout ce qui concerne fondamentalement l'information : les notions de *contenu en information* et de *compression*, la notion de *classification* et de *structuration*, et plus généralement les *bases de données* et les *systèmes d'information* (tels qu'ils se présentent actuellement). Cette théorie est également liée aux notions *d'intentionnalité* et *d'abstraction*, ou encore aux notions de *représentation*, de *syntaxe* et de *sémantique*. Vaste programme !

Ce double aspect (aléatoirité et classification) – dégagé par Kolmogorov à l'origine de sa théorie [28] – se trouve d'ailleurs en partie caractérisé, lorsque pour parler de la complexité de Kolmogorov (et pour la distinguer de la *théorie de l'information* de Shannon), on l'appelle dans bien des situations : *la théorie algorithmique de l'information*. On peut donc vraiment en espérer des applications dans des domaines qui pouvaient en paraître à priori étrangers et surtout cette théorie nous apparaît comme pouvant fournir un *cadre théorique unificateur* pour nombre d'approches du traitement de l'information.

Toutefois, il nous semble intéressant d'envisager une forme d'extension de la complexité de Kolmogorov, fondée pour l'essentiel sur la théorie des *fonctions calculables* et donc sur les *algorithmes*, en étendant celle-ci aux *ensembles*, aux *systèmes d'information* et aux *bases de données*. De cette façon, nous adoptons un point de vue *relationnel*, *non déterministe*, comparé au point de vue *fonctionnel*, essentiellement *déterministe*, initialement considéré par Kolmogorov et qui est celui qui est couramment adopté (ceci va d'ailleurs de pair avec la reconsidération des ASM dans le cadre du relationnel). La complexité de Kolmogorov et les ASM peuvent alors être approfondies dans le cadre de la dualité *fonctionnel versus relationnel* (cf. section **??** et section 4.2) [49].

Cela sous-entend que l'on s'intéresse à la complexité de Kolmogorov avec un point de vue plus raffiné (*plus structurel*, autrement dit, un point de vue *qualitatif*) que le point de vue initial de Kolmogorov, pour qui un programme et une output sont des mots binaires – mots binaires qui peuvent d'ailleurs représenter des ensembles, des graphes, des systèmes d'information, etc. et qui cherche avant tout à donner une *définition quantitative de la complexité d'un objet*.

C'est également dans une telle approche qualitative que se situe Codd lui-même, en élaborant le modèle relationnel des bases de données : la notion formelle d'attribut sur laquelle cette théorie est fondée, est justement de représenter et d'intégrer dans un cadre mathématique des caractéristiques d'ordre qualitatif pour les objets reliés entre eux par des liens de nature également qualitative. Une BD est une spécification formelle mais néanmoins mathématique tout aussi "scientifique" qu'un algorithme, qui traite les données et qui fait des calculs.

On peut, en particulier, s'intéresser, non pas à la taille du plus petit programme (calculant une output donnée), mais au programme lui-même et pourquoi pas

49. Nous étudions la dualité du fonctionnel et du relationnel dans [25] et la relation entre les ASM et la complexité de Kolmogorov ainsi que la reconsidération de ces théories avec le point de vue relationnel dans un article en préparation : Ferbus-Zanda M. *Kolmogorov Complexity and ASM : the relational point of view*, in preparation.

à l'ensemble de tous les programmes (calculant l'output). C'est avec une telle démarche, que l'on voit émerger des liens entre la théorie algorithmique de l'information et les ASM de Gurevich [50]. Ceci ouvre des perpectives intéressantes pour l'avenir. Comme nous en a fait, d'ailleurs, part un jour, Gurevich [51], en nous faisant remarquer que d'une certaine façon, la théorie de la complexité de Kolmogorov est encore loin d'avoir épuisé ses possibilités en matière d'application... La classification d'information par compression (et la classification avec Google) nous le montrent bien. C'est aussi avec une telle perspective structurelle que des variantes de la complexité de Kolmogorov ont été introduites comme la profondeur logique de Bennett [1] qui fait intervenir le temps de calcul des programmes dont un objet est l'output. On parle encore de *complexité organisée* d'un objet.

Toujours dans cet esprit (avec un tel niveau de raffinement), on peut en fin de compte se poser la question :

> *Pourquoi prendre le plus petit programme ? Qu'est-ce que ce plus petit programme a de particulier ?.*

La réponse nous vient de l'observation des ASM et de la correspondance de Curry-Howard :

> *Le plus petit programme, c'est le plus abstrait possible.*

En effet, la correspondance de Curry-Howard met en relation la logique et le λ-calcul – et c'est en ce sens que cette "correspondance" est un isomorphisme – et par extrapolation la logique et la programmation informatique. La correspondance de Curry-Howard joue un rôle fondamental dans l'articulation entre la théorie de la démonstration, les lambdas calculs typés, la théorie des catégories ainsi qu'avec les modèles de calcul qu'ils soient théoriques ou implémentés en machines, comme le sont les langages de programmation. Elle était connue par Curry pour la logique combinatoire dès 1934 et pour les systèmes de preuves à la Hilbert en 1958. Elle est étendue par William Howard en 1969, qui publie en 1980, un article [52] qui fait date [53].

50. comme nous l'avons entrepris dans *Ibid.* Note 49.

51. Il y a quelques années, lors d'un séjour à Paris.

52. Howard W. *The formulas-as-types notion of construction*, in *Essays on Combinatory Logic, Lambda Calculus and Formalism*. Seldin J.P., Hindley J.R. eds., Academic Press, pp. 479-490, 1980.

53. Joachim Lambeck publie également (années 70) sur cette correspondance concernant les combinateurs des catégories cartésiennes fermées et la logique propositionnelle intuitionniste. Noter que Nicolaas Debruijn (*Système Authomath*) et Per Martin-Löf ont également eu une influence décisive sur l'isomorphisme de Curry-Howard originel. Martin-Löf voyait le lambda-calcul typé qu'il développait, comme un (vrai) langage de programmation (Cf. Martin-Löf P. *Constructive Mathematics and Computer Programming*. Paper read at the 6-th International Congress for Logic, Methodology and Philosophy of Science, Hannover, 22 – 29 August 1979.) De la même façon, Thierry Coquand élabore la *théorie des construction* (1986) sur laquelle repose le système *Coq* développé initialement par Gérard Huet à l'INRIA (France) dans les années 80. (Cf. également note 55).

Sans entrer les détails, dans la correspondance de Curry-Howard, on considère que :

- Les formules logiques correspondent aux types des λ-calculs typés et aux types de données abstraits – comme on les appelle en informatique.
- Les preuves logiques correspondent aux λ-termes et aux programmes informatiques.
- L'élimination des coupures dans une preuve [54] correspond à la normalisation par diverses règles dont la β-réduction [55] des λ-termes et à l'exécution des programmes informatiques.

On voit alors se profiler le caractère *abstrait* des programmes évoqué plus haut. En effet, *la plus petite preuve logique (considérée dans un certain contexte) est en fait la preuve contenant le plus de* coupures. On a vu (cf. Note 54) que dans certains cas, *une coupure est une forme d'abstraction*. Signalons qu'une démonstration dont on a éliminé les coupures (et ceci revient donc dans certaines situations à remplacer un "cas général" par une pléiade de "cas particuliers"), est bornée en taille dans l'absolu par une "tour d'exponentielles"...

> *Plus une démonstration comporte de coupures, plus elle abstraite, et nous pouvons dire d'une certaine manière que plus cette démonstration est abstraite, plus elle compressée.*

De la même façon,

> *Plus un λ-terme contient de rédexes*[56]*, plus le λ-terme est abstrait, plus il est compressé.*

54. La notion de *coupure dans le Calcul des Séquents et la Déduction Naturelle* est une notion fondamentale en théorie de la démonstration, introduite par Gerhard Gentzen dans les années 30 – ainsi que ces deux calculs logiques. Dans certains cas, peut voir une coupure comme une forme d'abstraction où l'on remplace une multiplicité de cas particuliers par un cas général. Dans le calcul des séquents, une coupure est définie au moyen de la *règle de coupure*, qui est une généralisation du *Modus Ponens*. Le résultat fondamental de Gentzen est le *Hauptsatz*, qui énonce que toute démonstration dans le calcul des séquents peut être transformée en une démonstration de la même conclusion et qui n'utilise pas cette règle de coupure.

55. La β-réduction du λ-calcul originel de Church peut être complétée par diverses règles de réduction pour des contantes qui sont rajoutées au λ-calcul, de façon à étendre la correspondance originelle de Curry-Howard, entre la logique intuitioniste et le λ-calcul typé usuel, à la logique classique avec notamment, la notion de *continuation* , Thimothy Griffin, 1990. Voire, à certains axiomes comme *l'axiome du choix dépendant*. Et ceci est au coeur du travail de Jean-Louis Krivine, qui a introduit un certain nombre de ces *constantes fondamentales* et dont l'interprétation informatique est particulièrement profonde. (cf. Krivine J.L. Dependent choice, 'quote' and the clock. *Theoretical Computer Science*. 308, p. 259-276, 2003. voir aussi : http ://www.pps.jussieu.fr/∼krivine/).

56. Un rédexe pour un λ-terme donné t, est un sous-terme de t contenant une réduction qui peut se faire en une étape, au moyen, par exemple, de la β-réduction, i.e. un sous-terme de la forme : $((\lambda x.u)v)$ qui se réduit alors en $u[v/x]$ et qui est le terme u dans lequel on remplace toute occurence de x par v – en évitant les phénomènes de "capture" de variables.

Et pour les programmes informatiques, la notion de coupure peut également se définir pour les langages de programmation avec leurs primitives usuelles. On a, par exemple, qu'un programme contenant :

```
for i = 1 to 1000000 do print(i)
```

est plus abstrait que le même programme dans lequel on remplace cette séquence par :

```
do print(1) and do print(2) and ... and print(1000000)
```

La boucle `for` permet ainsi de réaliser des coupures. On aura donc un résulat similaires aux précédents :

> *Plus un programme contient de coupures, plus le programme est abstrait, plus il est compressé.*

On remarque que plus un programme est ainsi compressé au moyen de coupures, plus ce programme est *déclaratif*, autrement dit, moins le texte du programme contient de *contrôle*, i.e. de morceaux réalisant une exécution de parties du programme en question. Un programme entièrement compressé est totalement déclaratif.

Et les ASM, dans ce contexte ?

Comme on l'a vu, Les ASM permettent de représenter – et de façon particulièrement simple – le pas à pas de l'exécution de n'importe quel algorithme séquentiel au moyen d'un certain nombre de notions dont les modèles de la logique du premier ordre et quelques primitives de programmation. On imagine bien que la notion de coupure est particulièrement intéressante à étudier dans le cadre de cette théorie. Et toujours dans cet esprit, on voit également se profiler un lien profond entre les ASM et le λ-calcul et la correspondance de Curry-Howard. Pour le lecteur intéressé par ces questions, signalons un article à paraître pour un colloque en l'honneur de Yuri Gurevich [24], et dans lequel nous représentons les ASM dans le λ-calcul, et montrons ainsi que, tout comme les ASM, le λ-calcul est *algorithmiquement complet*.

Pour en revenir à la complexité de Kolmogorov, nous pouvons dire alors que :

> *Le plus petit programme qui calcule une output donnée est par conséquent le programme le plus abstrait, ou en se plaçant dans le λ-calcul, le λ-terme qui contient le plus de rédexes ou encore dans la perspective de la théorie de la démonstration, la preuve qui contient le plus de coupures.*

Dans tous les cas, il s'agit d'une forme d'abstraction. et ceci ne nous surprend pas : nous avons vu précédemment que la complexité de Kolmogorov est est fondamentalement liée à la notion d'abstraction.

En nous replaçant dans la perpective de l'information, nous pouvons dire, en somme que :

La connaissance est de l'information abstraite, compressée, comportant une part d'intentionnalité.

Connaissance qui sera à son tour compressée, etc. On notera que c'est bien sur ce mode que fonctionne le cerveau humain avec le langage et les mathématiques. Remarquons d'ailleurs que certaines abstractions ont un caractère "accidentel", dans le sens où il y a un moment donné où elles se sont produites et elles changent alors radicalement l'état de la connaissance. Un tel exemple d'abstraction est l'invention de la *transcription phonétique* des langues indo-européennes : avec une poignée de symboles (comme les lettres de l'alpahabet grec ou romain et quelques signes complémentaires), on peut écrire tous les textes de ces langues. On peut également les énoncer – ce qui ne veut pas dire les comprendre – : il suffit simplement de connaître un petit nombre de règles de prononciation spécifiques à chaque langue. On ne retrouve pas cette abstraction dans l'écriture chinoise...

Noter que c'est bien ce que nous montre la complexité de Kolmogorov : la découverte d'une (bonne) propriété constructive concernant, par exemple, un entier extrêmement long à écrire (chiffre par chiffre), et dont on ne savait pas grand chose, peut permettre alors de caractériser – et de décrire – cet entier de façon courte, abstraite, compressée. La connaissance se trouve ainsi accrue. De la même façon, la découverte d'une théorie comme le calcul intégral, diverses parties de la géométrie ou encore la géométrie fractale permet de donner des descriptions séquentielles courtes (calculables) de formes.

En particulier, il apparaît que la théorie de la complexité de Kolmogorov peut être une théorie très précieuse pour aborder de façon mathématique les approches de la classification qui sont pour l'essentiel, à l'heure actuelle, en dehors des bases de données relationnelles, des méthodes heuristiques (non encore pleinement formalisées dans ce que l'on peut attendre d'une méthode de classification), comme le sont, en particulier, la classification par compression et la classification avec Google. On peut aussi en espérer des applications dans d'autres domaines comme la sémiologie ou les sciences cognitives ou encore la biologie avec le génome, comme le montre, de façon remarquable, le biologiste Antoine Danchin dans un ouvrage paru en 1998 [12]. D'ailleurs la classification par compression est utilisée par certains biologistes dans cette perspective.

On conclura en soulignant encore l'immense utilité des méthodes de classification par compression ou au moyen de Google avec ce mode opératoire de type top-down étant donné que dans bien des cas, on se trouve avec des familles d'objets (quand elles peuvent être définies) dont on n'a aucune idée de la façon dont ils sont structurés entre eux. *On se trouve dans un monde purement syntaxique, un monde que l'on cherche à appréhender au moyen d'une certaine sémantique.* C'est, par exemple, le cas de l'ensemble des séquences de l'ADN des organismes du vivant et celui de la quantité pyramidale de fichiers figurant sur le Web...

Pour ce dernier exemple, on évoquera, sans toutefois partager son pessimisme,

cette citation de Edsger W. Dijkstra, si clairvoyante, extraite du discours [17] qu'il prononça lors de la remise du prix Turing en 1972 [57] :

> ≪ Tant qu'il n'y avait pas de machines, la programmation n'était pas un problème ; quand nous avons eu quelques ordinateurs de faible puissance, la programmation devint un problème moyen et maintenant que nous avons des ordinateurs gigantesques la programmation est devenue un problème tout aussi gigantesque. En ce sens l'industrie de l'électronique n'a pas résolu un seul problème, elle n'a fait qu'en créer. Elle a créé le problème de l'utilisation de son propre produit. ≫

Remerciements.

Pour Francine Ptakhine, qui m'a donné la liberté de penser et d'écrire.
Merci à Serge Grigorieff et à Chloé Ferbus pour leur écoute, les échanges fructueux et pour leur relecture attentive, et merci à Maurice Nivat qui m'accueillie au LITP en 1983.

Références

[1] Bennett C. Logical Depth and Physical Complexity. Dans *The Universal Turing Machine – a Half-Century Survey*. R. Herken (ed). Oxford University Press, p. 227–257, 1988.

[2] Bennett C., Gács P., Li M., Vitányi P. & Zurek W. Information distance. *IEEE Trans. on Information Theory*, 44(4) : 1407–1423, 1998.

[3] Chaitin G. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13 : 547–569, 1966.

[4] Chaitin G. On the length of programs for computing finite binary sequences : statistical considerations. *Journal of the ACM*, 16 : 145–159, 1969.

[5] Chaitin G. A theory of program size formally identical to information theory. *Journal of the ACM*, 22 : 329–340, 1975.

[6] Chen P.S. The Entity-Relationship Model : Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1) : 9–36, 1976.

[7] Cilibrasi R. Clustering by compression. *IEEE Trans. on Information Theory*, 51(4) : 1523–1545, 2003.

[8] Cilibrasi R. & Vitányi P. Google teaches computers the meaning of words. *ERCIM News*, 61, April 2005.

[9] Cilibrasi R. & Vitányi P. The Google similarity distance. *IEEE Trans. on Knowledge and Data Engineering*, 19(3) : 370–383, 2007.

57. Un recueil d'écrits sélectionnés de Dijkstra a été publié en 1982 [18].

[10] Codd E.W. A relational model of data for large shared databanks. *CACM*, 13, No 6, juin 1970.

[11] Codd E.W. *The relational model for database management. Version 2*. Addison-Wesley, 1990.

[12] Danchin A. *La barque de Delphes. Ce que révèle le texte des génomes*. Odile Jacob, 1998.

[13] Delahaye J.P. *Information, complexité, hasard*. Hermès, 1999 (2d edition).

[14] Delahaye J.P. Classer musiques, langues, images, textes et génomes. *Pour La Science*, 316 : 98–103, 2004.

[15] Delahaye J.P. *Complexités : Aux limites des mathématiques et de l'informatique*. Belin-Pour la Science, 2006.

[16] Dershowitz N. & Gurevich Y. A natural axiomatization of computability and proof of Church's thesis. *The Bulletin of Symbolic Logic*, Vol 14, Number 3, Sept. 2008.

[17] Dijkstra E.W. The Humble Programmer. *ACM Turing Lecture*, 1972. Disponible sur le Web : http ://www.cs.utexas.edu/ EWD/transcriptions/EWD03xx/EWD340.html

[18] Dijkstra E.W. *Selected writings on computing : A personal perpective*. Springer-Verlag, 1982.

[19] Durand B. & Zvonkin A. Complexité de Kolmogorov, dans *L'héritage de Kolmogorov en mathématiques*. E. Charpentier, A. Lesne, N. Nikolski (eds). Belin, p. 269–287, 2004.

[20] Evangelista A. & Kjos-Hanssen B. Google distance between words. *Frontiers in Undergraduate Research*, Univ. of Connecticut, 2006.

[21] Feller W. *Introduction to probability theory and its applications*, volume 1. John Wiley, 1968 (3d edition).

[22] Ferbus-Zanda M. & Grigorieff S. Is randomnes native to computer science ? In *Current Trends in Theoretical Computer Science*. G. Paun, G. Rozenberg, A. Salomaa (eds.). World Scientific, pages 141–179, 2004.

[23] Ferbus-Zanda M. & Grigorieff S. Kolmogorov complexity and set theoretical representations of integers. *Math. Logic Quarterly*, 52(4) : 381–409, 2006.

[24] Ferbus-Zanda M. & Grigorieff S. ASM and operational algorithmic completeness of Lambda Calculus, in *Studies in Honor of Yuri Gurevich. Lecture Notes in Computer Science*. To appear.

[25] Ferbus-Zanda M. Duality : Logic, Computer Science and Boolean Algebras. Soon submitted.

[26] Ferbus-Zanda M. Logic and Information System : Cybernetics, Cognition Theory and Psychoanalysis. Soon submitted.

[27] Kolmogorov A.N. *Grundbegriffe der Wahscheinlichkeitsrechnung*. Springer-Verlag, 1933. English translation : *Foundations of the Theory of Probability*, Chelsea, 1956.

[28] Kolmogorov A.N. Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1(1) : 1–7, 1965.

[29] Kolmogorov A.N. Combinatorial foundation of information theory and the calculus of probability. *Russian Math. Surveys*, 38(4) : 29–40, 1983.

[30] Li M., Chen X., Li X., Ma B. & Vitányi P. The similarity metrics. *14th ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[31] Li M. & Vitányi P. *An introduction to Kolmogorov Complexity and its applications*. Springer, 2d Edition, 1997.

[32] Shannon C.E. The mathematical theory of communication. *Bell System Tech. J.*, 27 :379–423, 1948.

[33] Solomonoff R. A formal theory of inductive inference, part I. *Information and control*, 7 : 1–22, 1964.

[34] Solomonoff R. A formal theory of inductive inference, part II. *Information and control*, 7 : 224–254, 1964.