Des humains dans la machine : la conception d'un algorithme de classification sémantique au prisme du concept d'objectivité

Églantine Schmitt

Dans Sciences du Design **2016/2 n° 4** , pages 83 à 97
Éditions Presses Universitaires de France

ISSN 2428-3711
ISBN 9782130734475
DOI 10.3917/sdd.004.0083
Date de mise en ligne : 07/12/2016

Des humains dans la machine : la conception d'un algorithme de classification sémantique au prisme du concept d'objectivité

Églantine Schmitt

Doctorante en épistémologie à l'Université de Technologie de Compiègne – Sorbonne Universités, EA 2223 COSTECH (UTC), groupe EPIN

Responsable études, Proxem, Paris, France

eglantine.schmitt@utc.fr

Mots-clés

Algorithmes
Épistémologie
Conception
Objectivité

Keywords

Algorithms
Epistemology
Design
Objectivity

Résumé

Un algorithme est le résultat de la formalisation d'une procédure qui, une fois implémentée dans un programme informatique, peut alors être rejouée indéfiniment sans intervention. La matérialité sociotechnique des programmes les inscrit dans des systèmes de contingences, de normes et d'habitudes, qui laissent la capacité d'action humaine au cœur du processus. Ni le caractère mécanique des programmes, ni la cohérence structurale de leurs fondements mathématiques, ne leur permettent de produire de l'objectivité par eux-mêmes. Elle provient de l'expertise de leurs concepteurs qui travaillent, soit par échange direct, soit par le biais d'outils d'évaluation, en interaction avec les utilisateurs bénéficiaires dont l'appréciation valide pragmatiquement les produits des algorithmes. C'est, en somme, le design des programmes par succession de choix humains qui fait d'eux des machines à produire des connaissances.

Abstract

An algorithm is a result of the formalization of a procedure which, when implemented in a computer program, can then be replayed indefinitely without intervention. The socio-technical materiality of programs puts them in systems of contingencies, standards and habits that leave the human capacity for action in the heart of the process. Neither the mechanical nature of the programs or the structural consistency of their mathematical foundations, will allow them to produce objectivity by themselves. It comes from the expertise of their designers interacting, either by direct exchange or through benchmarking tools, with end users whose appreciation pragmatically validates the products of the algorithms. It is, in short, the design of programs by succession of human choices that makes them machines for knowledge production.

Un algorithme est le résultat de la formalisation d'une procédure qui, une fois implémentée dans un programme informatique, peut alors être rejouée indéfiniment sans intervention (Sandvig, Hamilton, Karahalios et Lanfbort, 2015). Cette reproductibilité laisse à penser qu'un algorithme est un processus mécanique, nécessaire ou autonome qui ne laisserait aucune place à un certain degré de contingence et de jugement individuel. Je souhaiterais montrer au contraire que, si le fonctionnement d'un programme jouit d'une certaine autonomie, sa conception et son implémentation sont constituées d'une succession de moments de choix dans un espace de possibles qui laisse la capacité d'action humaine au cœur du processus.

À travers l'analyse d'un exemple de programme, ce travail se propose ainsi d'éclairer la pensée contemporaine sur les algorithmes en confrontant les vertus épistémiques qui leur sont conférées, et les critiques qui sont formulées à leur endroit sur le plan éthique, aux contingences matérielles qui préfigurent à leur conception. Pour ce faire, il s'intéresse à un point de vue peu entendu au-delà de sa communauté propre : non pas celui du chercheur qui les conçoit, ni celui de l'utilisateur qui en bénéficie (ou le subit), mais celui des développeurs, ingénieurs logiciel, ou informaticiens de métier, qui manipulent ces objets au quotidien.

Il répond ainsi au besoin de faire circuler les connaissances théoriques et pratiques mises en œuvre dans la conception des algorithmes entre le monde industriel et la recherche, entre l'informatique et les sciences de la culture (Cassirer, 1991), entre les théoriciens et les praticiens. Pour cela, la posture

épistémologique se propose de jouer le rôle de pont entre les mondes, et de construire l'espace et le langage d'une zone d'échange (Galison, 1996) entre les différents points de vue sur les algorithmes. Cette posture conduit à une entrée par la question gnoséologique (« comment caractériser les connaissances produites par les algorithmes ? ») qui permettra, à travers le prisme du concept d'objectivité (Daston et Galison, 2012), d'examiner comment les dispositifs techniques et les pratiques viennent compléter les structures logiques dans leur rôle de conditions transcendantales des résultats des artefacts computationnels. La matérialité sociotechnique des programmes, dans lesquels se réalisent les algorithmes, les inscrit dans des systèmes de contingences, de normes, de compromis qui caractérisent un objet social.

Je m'appuie sur un exemple concret de configuration d'un algorithme, issu d'une immersion en milieu informaticien, afin de mettre en lumière comment des normes, des contraintes et des habitudes s'impriment dans la matérialité technique des procédures de calcul pour en co-déterminer les résultats. Cette concrétisation d'un programme est considérée comme un travail de *design*, traduit en français par conception (Cormen et al., 2004). On verra ainsi qu'un algorithme ne fonctionne pas nécessairement ni exclusivement sous le régime de l'objectivité, que l'activité gnoséologique n'y est pas autotélique, et que leur usage renvoie plutôt à des systèmes de savoir-faire et de croyances d'obédience pragmatiste, dictés par la technicité de la conception de programmes et leurs conditions socioéconomiques de production.

1.— Les formes possibles de l'objectivité algorithmique

Dans leur livre *Objectivité* (2012), Lorraine Daston et Peter Galison proposent une histoire de la notion d'objectivité dans les sciences de la nature depuis le XVIII[e] siècle. Ils retracent pour cela les modes et les techniques de représentation de la nature qui ont émergé d'époque en époque.

Faire l'histoire d'une notion, c'est bien sûr signifier d'emblée que l'objectivité n'existe pas dans l'absolu, comme un idéal scientifique accessible aux meilleurs, mais comme une notion historicisée qui s'est construite, a évolué selon les époques, et n'a donc pas de définition universelle dogmatique telle qu'un philosophe des sciences pourrait en rechercher. L'objectivité n'y apparaît pas tant comme une propriété intrinsèque des savoirs scientifiques que comme un ethos auquel le chercheur s'identifie dans sa pratique de la science. Daston et Galison mettent en évidence plusieurs compréhensions historiquement constituées de ce que constitue une représentation objective de la nature, dont notamment :

— L'**objectivité mécanique** par l'effacement de la subjectivité de l'individu, de son ego, de ses idiosyncrasies stylistiques en substituant un instrument « sans volonté » à l'artisan humain. Dans ce sens, la photographie est perçue au moment de son invention comme plus objective que la peinture car elle est tenue pour capable de représenter la nature telle qu'elle est, contrairement à un artiste qui y imprègnera son style, ses affects, ses défauts, etc.

— L'objectivité comme effort actif fondé sur le **jugement exercé** d'un sujet doué de volonté, nécessaire pour produire une représentation fidèle à l'objet. C'est cette forme qui est à l'œuvre lorsque le médecin expérimenté est capable, à force de pratique, de détecter une pathologie en examinant une radio. Elle implique des notions d'intuition et d'inter-

prétation qui viennent contredire ou compléter le caractère mécanique de l'image produite.

— L'**objectivité structurale** rejette le témoignage toujours trompeur des sens et de la variabilité de la perception, et recherche des structures invariantes telles que la logique, la syntaxe, les mathématiques, les langages formels, comme pour fonder nos représentations de la nature.

La caractérisation d'un algorithme comme raisonnement mécanique (Rieder, 2012) l'inscrit à première vue dans un ethos d'objectivité mécanique et structurale. Le mode calculatoire reposerait sur des fondements mathématiques (donc structuraux) sur lesquels il s'effectuerait de manière mécanique, arrivant toujours nécessairement au même résultat déterministe ; la capacité véridictionnelle des algorithmes proviendrait de la systématicité avec laquelle ils s'exécutent de manière répétée en suivant toujours les mêmes contraintes logiques. Bien qu'il existe des algorithmes non-déterministes, cette conception est relativement vraie des algorithmes en général en tant que procédures abstraites (Bachimont, 1996 ; Andler, 1998 ; Pégny, 2012). L'informatique au temps de Turing, vraisemblablement inspirée par les recherches sur le formalisme du cercle de Vienne et notamment des travaux de Carnap (Monnin, 2015), est conçue comme une théorie du calcul sur une machine idéalisée (Mosconi, 2014). En ce sens, elle apparaît comme une science non-expérimentale dérivée des mathématiques dont elle hérite en grande partie son épistémologie.

Naturellement, le monde de l'informatique a changé depuis le temps de Turing, et il y a sans doute autant de différence entre la science du calcul et de l'information développée dans les années 1940 et de l'informatique actuelle, qu'entre la machine universelle imaginée par Turing à la même époque et nos ordinateurs modernes. L'informatique s'est graduellement muée en sciences des artefacts, « sciences de l'artificiel » (Simon, 2004) « sciences de l'artéfacture » (Bachimont, 1996) ou encore « science des artefacts interactifs » (Lassègue, 1996) ; elle ne sert pas tant à représenter qu'à intervenir dans le réel, pour reprendre la dichotomie de Ian Hacking (1983). En tant qu'effort de développement d'elle-même, elle occupe un statut épistémologique singulier, entre science et technique, entre nature et culture, où agir sur le monde importe autant que de se comprendre et de se définir par soi et pour soi ; en ce sens, l'intelligence artificielle, qui s'efforce à la fois de produire des artefacts intelligents et de comprendre ce qu'est l'intelligence, est emblématique de la situation épistémologique de l'informatique comme pratique principalement épistémique. En tant qu'outils de production de connaissances d'autre chose qu'elles-mêmes, les sciences computationnelles sont également envisagées comme une troisième voie entre la dimension théorique et la dimension empirique de la recherche scientifique, reliée à l'une par la simulation qui prolonge l'activité modélisatrice, et à l'autre par l'analyse de données qui en permet une meilleure compréhension (Hummon et Fararo, 1995 ; Varenne, 2007). Elle n'occupe cependant pas qu'un statut scientifique, mais également technique, où la valorisation de savoir-faire la constitue en secteur d'activité économique, à une échelle qui n'est anecdotique ni du point de vue de la situation de l'informatique ni de celui de l'économie des pays développés.

Une analyse des algorithmes du point de vue de la recherche en informatique théorique – approche décrite ci-dessus, et assez caractéristique de la philosophie des sciences – leur attribue une forme d'objectivité mécanique et structurale, et qui ne résiste que partiellement à l'examen des conditions dans lesquelles ils sont effectivement employés de nos jours. Cet examen, plutôt que de réduire à néant la capacité gnoséologique des algorithmes, nous conduira à leur attribuer une autre forme d'objectivité, celle du jugement exercé, plus à même de rendre compte de leur situation contemporaine.

L'intérêt pour le rôle des algorithmes dans la production de connaissance ne doit pas conduire à minimiser la contribution épistémique des données analysées, qui mériterait un développement à elles seules. Elles sont a priori conçues comme une forme d'observation du réel que l'on souhaite étudier, et dont l'objectivité reste, elle aussi, à construire. Suivant l'adage informaticien « garbage in, garbage out », un algorithme ne peut produire un résultat satisfaisant (du point de vue de ses vertus épistémiques par exemples) que si les données qui lui sont fournies en entrée lui sont adaptées d'une part (les données et l'algorithme constituent un système d'adaptation mutuelle (Gillespie, 2012), a fortiori dans les procédures d'apprentissage automatique), et ont une quelconque valeur d'autre part. La croyance dans l'existence de données « brutes », non biaisées, non manipulées, qui seraient l'exact reflet des faits qu'elles représentent et parleraient d'elles-mêmes (Mayer-Schönberg et Cukier, 2012), traduit une conception de l'objectivité aujourd'hui largement reconnue comme illusoire d'un point de vue à la fois épistémologique et sociologique au sein des sciences humaines et sociales (Drucker, 2011 ; Denis et Goëta, 2013 ; Gitelman, 2013 ; Crawford, Gray et Miltner, 2014).

2.— La culture épistémique des concepteurs de programmes

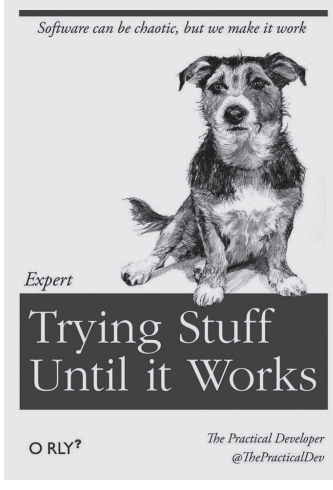
La pratique régulière de la programmation informatique peut favoriser avec le temps une certaine tournure d'esprit computationnelle qui tend à considérer chaque situation comme un problème à résoudre au moyen d'une procédure qu'il faut définir suffisamment formellement pour pouvoir la réappliquer au cas où la situation se présente à nouveau. Cet état d'esprit est utile à un bon informaticien, car la programmation est un exercice où les mêmes problèmes (de l'encodage à la parallélisation) se présentent sans cesse ; elle est en revanche particulièrement étrangère au chercheur en sciences humaines et sociales, pour lequel formaliser un objet et le « résoudre » veut dire cesser de le penser, de le travailler, et signe en somme la fin de son activité intellectuelle. Pour dépasser l'effet d'opacité des algorithmes et comprendre la façon dont ils sont conçus, le chercheur en SHS se doit cependant de comprendre cet état d'esprit computationnel, peut-être même de s'en imprégner, afin d'aller au-delà de l'impression de boîte noire. Il apparaît alors que la notion même d'opacité des algorithmes peut ne pas renvoyer à la protection d'un secret industriel, ou à un manque de familiarité avec les mathématiques et la programmation (Sandvig, Hamilton, Karahalios et Lanfbort, 2015), mais bien à une difficulté que rencontrent les informaticiens eux-mêmes d'une part pour comprendre le code de quelqu'un d'autre, et le résultat d'un calcul produit par un apprentissage automatique à partir de grands volumes de données et suivant un ensemble de paramètres complexes d'autre part (Burrell, 2016). L'opacité des algorithmes n'est alors pas tant le résultat d'une volonté d'opacification qui oppose d'un côté les informaticiens, et de l'autre le public et les chercheurs en SHS, mais une difficulté de la pratique même des algorithmes, qui met en tension leur interprétabilité et leur précision ou leur pouvoir prédictif (Breiman, 2001). Ce n'est qu'au prix d'une plongée dans cette technicité des algorithmes qu'il est possible de fonder des études critiques des algorithmes mettant en lumière les aspects cognitifs, sociaux et culturels de la conception des algorithmes.

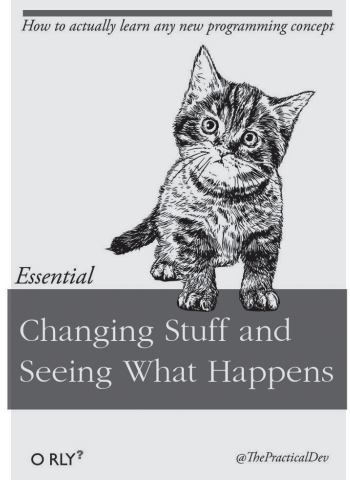
Les conditions dans lesquelles se déroule ma thèse, à savoir en parallèle d'une activité salariée chez un éditeur de logiciel, m'ont permis une immersion

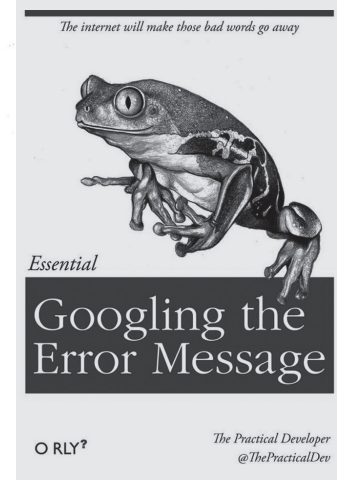
dans cette culture sans être perçue comme un « élément étranger » qui perturberait la spontanéité des activités et des échanges. De plus, mon initiation à la programmation (entendue comme pratique effective de l'informatique) avec le langage R, m'a fourni un angle d'investigation utile à la compréhension du travail de conception des algorithmes, ainsi qu'une mise en évidence de la diversité des pratiques et des postures qu'elle peut recouvrir ; je m'inscrivais pour ma part dans une optique de fouille de données. Cet article est une occasion de restituer une partie des observations que j'ai pu faire face à ma propre pratique et à celle de mes collègues.

Lors de cette initiation, l'une de mes plus grandes surprises a été de constater qu'il s'agissait davantage d'un travail de tâtonnement et de recherche d'information qu'un travail de réflexion logique. Je passais en somme peu de temps à réfléchir *in abstracto* à la façon dont j'allais résoudre un problème complexe (contrairement à ce à quoi je m'attendais), et je n'ai que rarement subi mon manque de culture mathématique ; l'énorme majorité des algorithmes très connus (comme par exemple les dix algorithmes les plus influents en *data mining* d'après Wu et al, 2008) existent sous forme de *packages* dans différents langages de programmation et utilisables en quelques lignes de code. En manipulant des algorithmes de fouille de données conçus et implémentés par d'autres, j'ai pu faire rapidement, en me hissant en quelque sorte sur les épaules des géants qui m'ont précédée, ce que je n'aurais jamais pu faire seule ; de nombreux *data scientists*, à mon sens, se trouvent dans cette situation. Je me retrouvais donc bien davantage à rechercher sur le Web (et tout particulièrement sur le forum StackOverflow) qui avait déjà résolu un problème similaire, comment j'allais pouvoir adapter sa solution, et pourquoi ce que je pensais avoir correctement copié ne voulait toujours pas fonctionner dans mon cas. Cette constatation est suffisamment partagée pour faire l'objet de plaisanteries telles que ces illustrations parodiant les manuels d'informatique de l'éditeur O'Reilly, dont les couvertures sont reconnaissables à leur mise en page et la présence emblématique d'un animal en noir et blanc.

Pour le praticien qui manipule ces outils au quotidien, l'informatique n'est pas tant un travail théorique (Lassègue, 1996) de manipulation syntaxique d'unité de sens (telle qu'on peut décrire les mathématiques), qu'une activité expérimentale (au sens presque littéral de travail par essai et erreur que Claude Bernard donne à la notion de méthode expérimentale) qui obtient des fragments de code, les éprouve, les teste, cherche à les comprendre, examine des résultats







et fait varier des paramètres en fonction, et repose de façon cumulative sur d'importantes archives de code déjà existantes.

L'expérience subjective que j'ai présentée est cependant loin de décrire tous les rapports à la pratique informatique qu'il peut exister. Si le chercheur en informatique théorique, l'informaticien de métier et le *data scientist* (pour dégager arbitrairement trois figures) s'appuient sur les mêmes fondements mathématiques théoriques, le premier les prolonge, le second les matérialise et le troisième exploite l'ensemble ; de plus, aucune de ces postures n'est mutuellement exclusive, et un informaticien de 2016 se situera probablement quelque part entre les trois. La conception de nouveaux algorithmes d'analyse de données est peu caractéristique du quotidien d'un informaticien de métier – ingénieur de développement logiciel ou intégrateur – qui construit des modules, les agence, les adapte et les maintient.

On renvoie bien souvent la figure de l'informaticien à la seule activité de programmation ; en pratique, il lui est nécessaire, pour accomplir correctement ses missions, de travailler comme un ingénieur, c'est-à-dire l'homme du projet (Bachimont, 2004 ; Choplin, 2013), la figure qui articule les enjeux et contraintes techniques à la réalité sociale, économique, pratique et politique dans laquelle s'inscrit ce projet. Il est concepteur d'un artefact qui, au-delà de sa matérialité technique, s'inscrit dans un système d'usages et de croyances qui doivent être pris en compte pour permettre l'utilisabilité et l'intelligibilité des programmes.

Dans le langage courant, le terme d'algorithme a pris une signification culturelle spécifique qui désigne paradoxalement tout dispositif de manipulation d'information d'une part, et d'autre part, quelques systèmes bien connus de hiérarchisation, de recommandation et ou catégorisation développés par certains éditeurs de sites Web ou de logiciel tels que Google, Amazon ou encore Netflix. Dans le vocabulaire quotidien des informaticiens de métier, le mot algorithme n'est pas tant utilisé que celui de programme, de méthode, de logiciel, de fonction… Lorsque cela se présente malgré tout, c'est notamment :

— au pluriel, pour désigner une famille de techniques d'analyse de données, comme par exemple « les algorithmes de clustering » ou « les algorithmes de tri » ;
— presque comme un synonyme de solution tactique, voire de ruse, pour désigner une façon de résoudre un problème au sein d'un système plus vaste, par exemple, un algorithme déterminant les dimensions et les proportions des différents éléments d'une visualisation de données au sein d'un logiciel d'analyse de données.

Dans le premier cas, les algorithmes sont typiquement des méthodes conçues par des chercheurs en informatique théorique sans les détails précis de leur implémentation, et désignent de ce fait une stratégie assez générale, une façon d'aborder un problème, voire quasiment une façon de penser à un problème. Pour l'informaticien de métier, un algorithme est un procédé trop abstrait pour pouvoir être utilisé tel quel ; il devra pour cela soit évaluer et comparer plusieurs implémentations déjà réalisées précédemment, en comparant l'environnement technique utilisé (langage de programmation, format de données, système d'exploitation, dépendances…), soit en écrire sa propre implémentation, qui ne sera jamais l'algorithme lui-même, mais une certaine interprétation ; dans certains cas, plusieurs implémentations produiront strictement le même résultat, dans d'autres, pas toujours, voire presque jamais. On a coutume de présenter les algorithmes comme des procédures strictes, formelles, non ambiguës, mais pour un informaticien de métier, ces procédures se situent encore à un niveau assez abstrait. On pourrait dire dans la pratique que les algorithmes n'existent pas en soi, seulement leurs représentations, via la matérialité du programme.

Dans sa culture comme dans sa pratique, l'écriture de programmes informatiques relève de la technique et de l'expérimentation plutôt que de la science et de la théorie ; comme de nombreux savoir-faire, chaque informaticien la pratique avec un certain style identifiable de ses semblables – dans le monde de l'édition logicielle, où plusieurs développeurs collaborent sur un même projet, il n'est pas rare qu'ils puissent identifier l'auteur d'un programme ou d'une partie d'un programme simplement en le lisant. Si la programmation informatique s'appuie donc bien sur des structures logiques, c'est davantage sur le mode du jeu que de celui de l'édification. Ces structures sont ses conditions de possibilités et non les garantes de son objectivité. L'objectivité structurale des logiciens et des mathématiciens telle que présentée par Daston et Galison fonde l'exactitude du calcul mais nullement sa capacité à représenter ou intervenir adéquatement dans le réel.

3.— Un algorithme de classification dans sa singularité

L'approche algorithmique se distingue de l'exécution manuelle d'une tâche par son caractère mécanique et systématique. De façon similaire, la production industrielle a permis de mécaniser et systématiser le geste de l'artisan ; l'informatique prolonge cette mécanisation de l'activité humaine en s'appliquant non pas à certaines actions du corps, mais à certaines opérations mentales. Le principal intérêt des ordinateurs qui nous entourent n'est pas de pouvoir réaliser des actions complexes mais de reproduire très rapidement des actions élémentaires. Si l'on voulait prêter des traits humains aux programmes, ce ne seraient pas l'ingéniosité ou la sagacité, mais l'obstination, l'imperméabilité à la fatigue et à l'ennui. Ecrire un programme, c'est décomposer rigoureusement une tâche en éléments suffisamment simples pour pouvoir être reproduits sans équivoque. Ce caractère mécanique en est-il pour autant le fondement de l'objectivité des algorithmes ? Je montrerai ici que s'il n'est pas dénué d'intérêt, il est aussi la contrainte avec laquelle l'informaticien doit jouer pour préserver, en regard de l'efficacité des programmes, l'intelligibilité des manipulations de données qu'ils rendent possibles. Pour mettre en évidence ce jeu, je propose un cas d'étude du moment où ce jeu se produit, fondé sur mes observations des pratiques d'un éditeur de logiciel. Par ce récit, qui expose les conditions pratiques de la conception d'un programme spécifique, je mets en tension la gnoséologie générale des algorithmes avec l'étude d'un exemple concret dans sa singularité ; de cette façon, je montre comment une réflexion sur les algorithmes doit prendre en compte la diversité et la variabilité des configurations matérielles et méthodologiques avant de remonter en généralité. Par cet examen, je propose également une méthode pour rendre compte du fonctionnement d'un algorithme qui pourra, à terme, s'appliquer à d'autres exemples et permettre une approche comparative entre des observations commensurables.

L'éditeur de logiciels envisagé est Proxem, qui commercialise un logiciel « d'exploration et de visualisation de données textuelles »[01] (et au sein duquel je suis salariée). Les entreprises qui achètent le logiciel le font dans le cadre d'un projet au cours duquel elles confient leurs données textuelles (ou documents) à Proxem. Les infolinguistes (salarié-e-s de l'éditeur formé-e-s en informatique et en linguistique) se chargent alors de traiter les données au moyen d'outils internes, suite à quoi les données analysées sont envoyées dans

01. https://www.proxem.com/ubiq-application-exploration-donnees-textuelles-saas/ (consulté le 7 avril 2016).

Sciences du Design — 04 — Novembre 2016

l'outil d'exploration auquel le client a accès. L'exemple qui va suivre est celui de la conception de l'un des outils internes utilisés par les infolinguistes : un analyseur sémantique, c'est-à-dire un programme (ou une certaine version d'un programme) qui analyse un ensemble de documents textuels pour en extraire des informations spécifiques et les organiser d'une certaine façon. Les documents textuels traités par ce programme se voient attribuer automatiquement une catégorie en fonction de leur contenu, et selon une logique définie au préalable par l'infolinguiste.

L'algorithme envisagé ici est singulier à plusieurs égards :

- il porte sur la manipulation de textes, là où la mouvance des *data sciences* parle le plus souvent de données structurées ;
- il est utilisé par plusieurs personnes qui forment une communauté de pratique mais restent des agents distincts confrontés à un besoin de collaborer, de travailler ensemble ;
- il est développé par un éditeur de logiciel, mais utilisé uniquement en interne par les salariés de l'entreprise ;
- parmi ses développeurs se trouvent à la fois d'anciens chercheurs et des informaticiens de métier ;
- il s'inscrit dans un système logiciel plus vaste de traitement et d'analyse textuels composé de boîtes à outils, d'interfaces, de tâches planifiées, d'espaces d'administrations, de fichiers de configurations, etc. ;
- l'usage de l'algorithme est prescrit par sa conception technique, mais aussi par les habitudes et conventions adoptées par ses utilisateurs, ainsi que par les contraintes posées par les bénéficiaires des résultats de la procédure (les clients finaux) ;
- il permet de ce fait l'apparition d'un savoir-faire partagé et de style d'utilisation propre à chaque utilisateur.

Les données à analyser par l'algorithme sont généralement fournies par le client, qui joue donc un rôle de garant de leur valeur épistémologique ; s'il souhaite les faire analyser, c'est précisément parce qu'il leur présuppose une représentativité, une significativité, une capacité à révéler des connaissances nouvelles ; le rôle du logiciel et de ses utilisateurs n'est pas d'être la source a priori de cette valeur, mais de la préserver à travers les différentes étapes de l'analyse. Selon l'offre commerciale choisie et la problématique à traiter, il peut s'agir d'avis de consommateurs, de contenus Web, de réponses aux questions ouvertes d'un sondage, ou encore de CV, de documents administratifs ou juridiques, etc. La nature des connaissances qui résulteront de l'analyse dépend de la teneur de ces données, et la valeur de ces connaissances, du degré auquel ces données constituent des observations du réel.

La transmission des données entre le client et le prestataire s'inscrit dans une technicité caractéristique de l'ensemble de la démarche ; les données ne sont pas magiquement absorbées par l'algorithme. Elles ne peuvent y être soumises telles quelles. La matérialité technique du programme implique entre autres qu'il exige un certain format d'entrée pour remplir son rôle initial. Concrètement, les données sont stockées dans un fichier, par exemple un tableau au format csv, constitué de colonnes dont une ou plusieurs contiennent du texte que l'on souhaite analyser. Si une mauvaise colonne est marquée comme contenant du texte, le programme s'exécutera cependant en produisant des résultats sans intérêt d'un point de vue épistémique mais corrects d'un point de vue purement computationnel : les différentes étapes de calcul auront bien été jouées, dans l'ordre prévu, etc. Au-delà de la remarque triviale selon laquelle il faut éviter de faire des erreurs quand on manipule un programme, il faut surtout retenir que le caractère aveugle et mécanique du computationnel s'exprime en ces termes, dans son incapacité à discerner ce qui pour l'agent est du texte ou non.

Il ne manipule pas des éléments de sens comme le ferait un lecteur qui aborde un texte, mais des symboles dont il n'exploite pas la dimension sémiotique. En ce sens, son caractère mécanique se présente comme quelque chose que l'agent humain doit compenser pour que l'ensemble fonctionne, et non comme une puissance aveugle et inéluctable contre laquelle les agents humains ne pourraient rien.

Ce dernier point est, bien sûr, à nuancer, dans la mesure où un programme contraint, c'est-à-dire prédétermine partiellement, la façon dont l'agent l'utilise pour exécuter une certaine tâche. Par exemple, si le programme propose de sélectionner les colonnes qui contiennent du texte, l'agent ne pourra pas sélectionner des lignes, ou des cellules, ce qui peut être problématique dans certains cas : sa capacité d'action est limitée par le programme. Il devra alors modifier le tableau de données pour pouvoir l'utiliser dans le programme, ou modifier, s'il le peut, le programme, pour prendre en charge ce nouveau cas d'usage. La démarche consistant à déterminer que l'utilisateur doit pouvoir choisir où se trouve le texte, qu'il doit pour cela sélectionner des colonnes dans un fichier, qu'il faut donc pouvoir pré-visualiser les données avant de les importer, et ainsi de suite, est une illustration concrète de la démarche de conception d'un programme. À travers une fonctionnalité relativement simple, par laquelle l'utilisateur indique au programme les zones de texte, on voit déjà apparaître la multitude de possibilités, et de choix parmi ces possibilités, qui incombent au concepteur du programme. Ce sont les modalités de ces choix, les facteurs de décision, que nous chercherons à examiner.

4.— La conception d'un algorithme de classification sémantique

La fonction principale du programme est une tâche de classification, c'est-à-dire un ensemble d'opérations visant à attribuer à chaque élément (ici, à chaque document) une classe prédéfinie dans une taxonomie. Dans la vie courante, nous manipulons sans cesse des taxonomies sans les interroger ; nous pouvons désigner un rhume, un oiseau, un instituteur, en nous faisant comprendre de nos interlocuteurs. Ces catégories sont pourtant le résultat d'une construction à la fois sociale et théorique grâce à laquelle nous tombons d'accord sur ce qu'elles contiennent la plupart du temps. La classification des maladies (Bowker et Star, 1999), la taxonomie phylogénétique (Desrosières, 2010), les nomenclatures socioprofessionnelles (Amossé, 2013) sont autant d'exemples de typologies historiquement constituées et plus ou moins stabilisées, bien que régulièrement remises en cause, et qu'il n'est donc pas possible d'aborder avec un regard naturaliste ou positiviste. Dans le cas de l'algorithme de Proxem, ces typologies reposent sur des unités linguistiques constituées en concepts, motifs ou thématiques, comme par exemple les différents types de problèmes rencontrés par un consommateur fréquentant un supermarché (attente en caisse, rupture de stock, propreté du magasin, etc.). Dans l'univers des enquêtes d'opinion, la définition de ces classes et leur attribution à un corpus de réponses s'appellent la codification et les classes sont appelées des codes ; le travail de codification est traditionnellement réalisé par la personne qui analyse des données (plutôt que par l'enquêteur, qui a cependant tendance à pré-codifier les réponses en les reformulant lors de leur retranscription) et il répond à un certain nombre de normes et d'habitudes partagées par une communauté professionnelle (Marc, 2001).

Formellement, le programme exécute un algorithme de classification supervisé, c'est-à-dire une procédure visant à rattacher automatiquement des éléments à des classes déterminées par avance. Cependant, le terme de classification supervisée fait généralement référence à une procédure d'apprentissage automatique (*machine learning*) par laquelle un algorithme de classification s'efforce, à partir d'exemples déjà classés, de déterminer automatiquement les règles selon lesquelles un exemple est rattaché à une classe. De nombreux algorithmes classiques dans la littérature en apprentissage automatique correspondent à cette démarche, par exemple les arbres de décision, les machines à vecteur de supports, la méthode des *k* plus proches voisins, les réseaux de neurones artificiels, etc. Néanmoins, un algorithme de classification supervisée n'est pas mécanique dans la mesure où son résultat dépend grandement des classes qui sont définies, des critères donnés à l'algorithme pour déduire ses règles, et de l'adéquation entre les classes et les données. Ainsi, un problème récurrent de la branche du traitement automatique du langage qui s'efforce d'attribuer une tonalité à un document est que la notion même de tonalité est difficilement définissable, et qu'il n'y a pas de consensus entre des agents humains s'efforçant de déterminer si un texte est « positif » ou « négatif » (Manning, 2011).

Par ailleurs, la classification automatique effective (par opposition au problème théorique, sans données) est un travail d'ingénierie qui repose sur de nombreux tâtonnements et ajustements manuels pour transformer les données, modifier des paramètres, essayer plusieurs techniques, voire combiner plusieurs techniques pour obtenir un meilleur résultat. Dans les débuts de l'apprentissage automatique, la comparaison entre techniques a pu être une question de validité théorique. Un algorithme pouvait être jugé meilleur qu'un autre a priori, indépendamment du problème précis posé (Domingos, 2012), notamment du fait de ses fondements mathématiques, de l'école statistique à laquelle il renvoie, de sa similarité supposée avec la cognition humaine. De nos jours, cette attitude est plus rare, en particulier chez les ingénieurs et dans l'industrie. Des outils de comparaison, comme par exemple le *package* « caret » (Kuhn, 2008) dans le langage de programmation R, permettent d'appliquer facilement une quarantaine de techniques distinctes et d'en évaluer les performances (Fernández-Delgado, Cernadas, Barro et Maorim, 2014). Par ailleurs, plusieurs techniques peuvent être combinées dans un modèle d'ensemble (*ensemble models*) dont les résultats sont généralement meilleurs que chaque technique prise indépendamment. Ce procédé a par exemple été utilisé par les gagnants de la compétition Netflix Prize organisée par le site Kaggle qui ont associé plus de 100 classifieurs différents en combinant leur travail et celui de l'équipe classée deuxième (Domingos, 2012). Dans la culture des praticiens de la classification automatique, il ne s'agit donc pas de comprendre pourquoi une technique donne de meilleurs résultats, mais de l'identifier et d'essayer de l'améliorer encore.

En ces termes, une procédure de classification supervisée ne produit pas de résultats mécaniquement sans intervention humaine. L'ingénieur intervient sur les données, les procédures, et les ajustements possibles en essayant d'améliorer les résultats par tâtonnements. Dans les systèmes utilisés en production, des paramètres et des règles sont souvent ajoutés manuellement pour garantir ou améliorer le fonctionnement du programme, par exemple parce que les résultats sont différents sur plusieurs serveurs et sur l'ordinateur personnel du concepteur, ou parce que le système produit systématiquement une erreur qui peut être corrigée. De plus un algorithme de classification tel qu'il a été présenté s'intègre presque toujours dans un système où se combinent, s'agencent, interagissent par rétroaction plusieurs programmes, méthodes, modules différents, eux-mêmes modifiés en permanence par les interactions des utilisateurs (Gomez-Uribe et Hunt, 2015).

Dans le cas de Proxem, la typologie qui sert de point de départ à l'algorithme est le résultat d'une négociation entre les infolinguistes et le client, les un-e-s montrant ce qu'une analyse du texte fait ressortir, l'autre décrivant des indicateurs qu'il cherche à mesurer au travers des données qu'il a fournies. Si l'on reformule dans des termes plus généraux, elle est le produit d'un travail itératif d'adéquation entre les observations et les hypothèses. Cette typologie n'est donc pas générée par l'algorithme, bien qu'ils soient co-dépendants dans la mesure où dans un sens, la typologie doit être constituée de classes techniquement repérables, et dans l'autre, l'algorithme ne fera jamais que suivre ces classes, et non en suggérer par exemple.

La définition des classes est partiellement arbitraire du point de vue des données, mais aussi partiellement dérivée de la matière sur laquelle elles portent. Des techniques linguistiques et statistiques permettent de faire émerger des regroupements de données autour de prénotions qui pourront être constituées en classes. Pour cela les infolinguistes de Proxem utilisent un ensemble d'outils de fouille de texte dégageant des régularités mais aussi sur une culture linguistique constituée de normes individuelles, de conventions partagées entre collègues (sur les libellés et la hiérarchisation des classes, en particulier), et d'habitudes stabilisées par l'expérience. Le principe général est d'identifier les règles linguistiques qui permettront à l'algorithme de rattacher un document à une classe. Pour cela, les infolinguistes s'appuient sur les observables du texte : pour qu'une classe soit activable, il faut qu'elle repose sur un certain nombre de concepts lexicalisés, c'est-à-dire matérialisés par des ensembles de mots. Le formalisme développé par Proxem pour ces règles (Chaumartin 2012) permet de solliciter la présence ou l'absence de combinaisons de termes. L'écriture même de ces règles peut être manuelle (et renvoyer elle aussi à un jeu d'habitudes et un style de mise en œuvre, comme le tour de main du potier) ou partiellement automatisée à partir d'outils de statistique textuelle, de lexicométrie, de traitement automatique du langage, d'ingénierie des connaissances, etc. Elle s'appuie également sur des ressources sémantiques externes constituées manuellement comme DBpedia, WordNet, ou automatiquement comme la base de *word embeddings* de Mikolov, Chen, Corrado et Dean (2013), générées par apprentissage automatique sur de grands corpus de documents, et dont le principe peut par ailleurs être réappliqué par Proxem sur les données textuelles dont il dispose. Toutes ces techniques produisent des assemblages de termes qui peuvent servir de base à la constitution d'une règle pour l'algorithme. Les règles sont donc fixées manuellement, mais sur la base de suggestions qui peuvent être obtenues de manière au moins partiellement automatisée. Les règles utilisées dans un projet donné peuvent dans une certaine mesure être réutilisées dans un projet nouveau, linguistiquement et pragmatiquement similaire. D'un point de vue méthodologique, l'agrégation des bases de connaissances produites sous des régimes épistémologiques différents renvoie à un opportunisme à la Feyerabend (« anything goes », tout peut convenir) qui écrase les points de vue réalistes, pragmatistes ou relativistes, les approches symboliques, herméneutiques ou distributionnelles, et les autres particularités de ces différentes approches. Du point de vue des infolinguistes, il ne s'agit pas de modéliser le sens comme un calcul mais d'interpréter les suggestions automatiques et de faire des observations informées, de nature interprétative, à la manière d'une sémantique d'inspiration rastérienne (Pincemin 2012). Il n'existe pas de typologie idéale ni de règle parfaite, mais des approximations dont on s'efforce de réduire l'arbitraire.

Dans l'approche dite « ascendante » qui consiste à constituer les classes à partir des régularités dans les données constituées en règles, le contenu, les frontières et les libellés des classes sont soumises à un certain arbitraire qui se résout dans la négociation avec le client (qui précise les classes qui

lui importent, la signification qu'il leur attribue, et en un sens, leur donne leur légitimité) et dans une recherche d'équilibre entre les classes : par exemple une classe trop large, qui agrègerait beaucoup de documents que les autres, sera découpée en plusieurs pour des raisons d'homogénéisation. Malgré ces regroupements, le client perçoit le résultat de la classification dans sa positivité. Selon son degré d'implication dans la conception des classes, il les considère comme des faits naturels ou des construits ; la façon dont les documents sont rattachés aux classes lui est plus opaque même si le logiciel lui permet de voir quel fragment du texte justifie le rattachement. L'approche ascendante suggère cependant l'idée que les classes émergent pour ainsi dire naturellement des données, et acquièrent de ce fait une légitimité empirique.

À la suite de la constitution des classes et des règles, l'ensemble de ces paramètres – ainsi que le mode d'import des données, et les documents eux-mêmes – sont soumis à l'analyseur sémantique. L'analyseur est un programme qui exécute une succession d'opérations dont le chargement des données, leur formatage, le découpage du texte en mots et en phrases, et enfin l'application des règles qui ont été définies par l'infolinguiste. Le résultat fourni par le programme (qui est alors intégré dans le logiciel que Proxem fournit à ses clients afin de pouvoir le visualiser) consiste en une catégorisation, c'est-à-dire un ensemble de documents annotés sémantiquement selon les règles définies par les infolinguistes, qui permet de mesurer le nombre de documents appartenant à chaque classe.

L'algorithme applique rigoureusement l'ensemble des règles de classification sur l'ensemble des documents ; si on l'oppose à un agent humain, qui n'aurait pas de règles explicites, mais organiserait les documents selon ce qui lui semble raisonnable au fur et à mesure, sans être parfaitement cohérent dans ses choix, alors le caractère mécanique de la procédure informatique paraît évident. En ce sens, un analyseur sémantique évoque l'objectivité mécanique qui était attribuée à la photographie par opposition à la peinture, chargée de la subjectivité de l'artiste. Néanmoins, si l'application des règles est bien systématique, leur constitution est le résultat d'un ensemble de choix qui entérinent des négociations entre agents, l'intégration de normes scientifiques (notamment linguistiques) et techniques, et matérialisent également un savoir-faire non verbalisé. Ce sont ces choix, et non leur application systématique par le programme, qui portent l'essentiel de la charge épistémique de la procédure, et constituent les conditions d'intelligibilité des résultats par les utilisateurs finaux.

Conclusion

Qu'il s'agisse d'une classification manuelle, à base de règles, ou par apprentissage automatique, dans la phase de tâtonnement comme dans la phase de mise en usage effectif, les ingénieurs ne cessent de mettre en œuvre un jugement exercé qui fonde la valeur des résultats du calcul. Leur savoir-faire s'exprime comme un art, une *technè* aussi marquée par l'idiosyncrasie de la personne qui la pratique. Ni le caractère mécanique des programmes, ni la cohérence structurale de leurs fondements mathématiques, ne leur permettent de produire de l'objectivité par eux-mêmes. Ces caractéristiques les rendent plutôt capables de préserver la façon dont les ingénieurs confèrent une valeur épistémique aux données et à la procédure, selon les normes, les techniques et la culture épistémique qui leur sont propres, ou qu'ils partagent avec d'autres agents impliqués dans la validation des connaissances. Le caractère véridictionnel des résultats des algorithmes provient de l'expertise de leurs concepteurs

qui travaillent, soit par échange direct, soit par le biais d'outils d'évaluation, en interaction avec les utilisateurs bénéficiaires dont l'appréciation valide pragmatiquement les produits des algorithmes. C'est, en somme, le *design* des programmes qui fait d'eux des machines à produire des connaissances. ◀

RÉFÉRENCES

AMOSSÉ, T. (2013), « La nomenclature socio-professionnelle : une histoire revisitée », *Annales. Histoire, Sciences Sociales*, (4), p. 1039-1075.

ANDLER, D. (1998), « Turing : pensée du calcul, calcul de la pensée », in *Les années 1930 : réaffirmation du formalisme*, Paris, Vrin, p. 1-33.

BACHIMONT, B. (1996), *Herméneutique matérielle et artéfacture, Des machines qui pensent aux machines qui donnent à penser*, Thèse de doctorat de l'École Polytechnique.

BACHIMONT, B. (2004), *Arts et sciences du numérique : Ingénierie des connaissances et critique de la raison computationnelle*, Mémoire d'habilitation à diriger des recherches de l'Université de technologie de Compiègne.

BOWKER, G. C., ET STAR, S. L. (1999), *Sorting things out. Classification and its consequences*, Cambridge, Massachusetts, MIT Press.

BREIMAN, L., COX, D., ET BREIMAN, L. (2001), « Comment-Statistical Modeling : The Two Cultures », *Statistical Science*, *16*(3), p. 199-231.

BURRELL, J. (2016), « How the machine 'thinks : Understanding opacity in machine learning algorithms », *Big Data & Society*, *3*(1), p. 1-12.

CASSIRER, E. (1991), *Logique des sciences de la culture*, Paris, Editions du Cerf.

CHAUMARTIN, F.-R. (2012), *Antelope, une plate-forme de TAL permettant d'extraire les sens du texte*, Thèse de doctorat en linguistique de l'université Paris Diderot (Paris VII).

CHOPLIN, H. (2013), *L'Ingénieur contemporain, le philosophe et le scientifique*, Paris, Les Belles Lettres, collection Encre marine.

CORMEN, T., LEISERSON, C., RIVEST, R., STEIN, C., HANEN, C., MUNIER, A., & PICOULEAU, C. (2004), *INTRODUCTION À L'ALGORITHMIQUE. Cours et exercices,* Paris, Dunod.

DENIS, J., & GOËTA, S. (2013), « La fabrique des données brutes. Le travail en coulisses de l'open data. », communication présentée pendant la journée d'études SACRED *Penser l'écosystème des données. Les enjeux scientifiques et politiques des données numériques*, 13 février 2013.

DESROSIÈRES, A. (2010), *La politique des grands nombres. Histoire de la raison statistique*, Paris, La Découverte.

DASTON, L. J., & GALISON, P. L. (2012), *Objectivité*, Dijon, Les Presses du réel.

DOMINGOS, P. (2012), « A few useful things to know about machine learning », *Communications of the ACM*, *55*(10).

DRUCKER, J. (2011), « Humanities approaches to graphical display ». *Digital Humanities Quarterly*, *5*(1), p. 1-23.

GALISON, P. (1996), « Computer simulations and the trading zone », in *The disunity of science : Boundaries, Contexts, and Power*, Redwood, California, Stanford University Press.

GILLESPIE, T. (à paraître), « The Relevance of Algorithms », in T. Gillespie, P. Boczkowski, & K. Foot (Eds.), *Media Technologies*, Cambridge, Massachusetts, MIT Press.

GITELMAN, L. (2013), *Raw Data Is an Oxymoron*, Cambridge, Massachusetts, MIT Press.

GOMEZ-URIBE, C. A., & HUNT, N. (2015), « The Netflix Recommender System : Algorithms, Business Value, and Innovation », *ACM Transactions on Management Information Systems*, *6*(4), p. 1-19.

HACKING, I. (1983), *Representing and Intervening*, Cambridge, Cambridge University Press.

HUMMON, N. P., & FARARO, T. J. (1995), « The emergence of computational sociology », *The Journal of Mathematical Sociology*, *20*(2-3), p. 79-87.

LASSÈGUE, J. (1996), « La méthode expérimentale, la modélisation informatique et l'intelligence artificielle », *Intellectica*, *22*(1), p. 21-65.

MARC, X. (2001), « Les modalités de recueil des réponses libres en institut de sondage : le rôle de l'enquêteur, les consignes et les procédures de contrôle, les perspectives d'amélioration », *Journal de la société française de statistique*, *142*(4), p. 21-28.

MANNING, C. D. (2011), « Part-of-Speech Tagging from 97% to 100% : Is It Time for Some Linguistics ? », in Alexander Gelbukh (dir.),*Computational Linguistics and Intelligent Text Processing, 12th International Conference, CICLing 2011, Proceedings, Part I*, Springer, Lecture Notes in Computer Science 6608, p. 171-189.

MAYER-SCHÖNBERGER, V., & CUKIER, K. (2013), *Big Data : A Revolution That Will Transform How We Live, Work, and Think*, Londres, John Murray.

MIKOLOV, T., CHEN, K., CORRADO, G., & DEAN, J. (2013), « Distributed Representations of Words and Phrases and their Compositionality », in *Proceedings of Neural Information Processing Systems*, p. 1-9.

MONNIN, A. (2015). « L'ingénierie philosophique de Rudolf Carnap. De l'IA au Web sémantique », *Cahiers Philosophiques*, *141*, p. 27-53.

MOSCONI, J. (2014), « Constructivity and Computability in Historical and Philosophical Perspective », in J. Dubucs & M. Bourdeau (dir.), *Constructivity and Computability in Historical and Philosophical Perspective*, Dordrecht, Springer Netherlands, p. 37-56.

RIEDER, B. (2012), « *Probability at Work : Information Filtering as Technique* », en ligne (consulté le 28 avril 2016) : http://goo.gl/l9aT0j

PÉGNY, M. (2013), *Sur les limites empiriques du calcul. Calculabilité, complexité et physique*, Thèse de doctorat de l'Université Paris 1 Panthéon-Sorbonne.

PINCEMIN, B. (2012), « Sémantique interprétative et textométrie », *Texto ! Textes et Cultures*, *XVII*, p. 1-21.

SANDVIG, C., HAMILTON, K., KARAHALIOS, K., & LANGBORT, C. (2015), « Can an Algorithm be Unethical ? », communication présentée à la conférence *65th annual meeting of the International Communication Association*.

SIMON, H. A. (1969), *The Sciences of the Artificial*, Cambridge, Massachusetts, MIT Press.

VARENNE, F. (2007), *Du modèle à la simulation informatique*, Paris, Vrin.

WU, X., KUMAR, V., ROSS QUINLAN, J., GHOSH, J., YANG, Q., MOTODA, H., STEINBERG, D. (2008), « Top 10 algorithms in data mining », *Knowledge and Information Systems* (14).