Enseigner l'informatique à des linguistes (filières TAL)

Isabelle Tellier [1]

Cet article est basé sur l'exposé donné par Isabelle Tellier, Professeure en Sciences du langage et précédemment Professeure en Informatique, lors des journées pédagogiques de la SIF qui se sont déroulées à Paris (CNAM) du 23 au 24 juin 2015. Le thème de ces journées était « L'enseignement de l'informatique pour les humanités et les sciences sociales ».

Le traitement automatique des langues (TAL) est une discipline éminemment pluridisciplinaire, qui requiert une vraie double compétence en informatique et en linguistique. Des filières de niveau Master dédiées à ce domaine existent dans des départements de sciences du langage, où elles recrutent essentiellement des linguistes (c'est le cas à Sorbonne Nouvelle – Paris 3 où j'enseigne). Des initiations sont également souvent proposées au niveau licence (L2 ou L3). Comment se passent les enseignements d'informatique dans ce contexte ? Quelle informatique privilégier, comment y intéresser les étudiants ?

La chance de l'informatique, c'est que son histoire est très liée à celle de la linguistique : les deux disciplines ont beaucoup de points communs et de références communes (Chomsky en est une illustration exemplaire). Leurs évolutions récentes ont aussi suivi des itinéraires parallèles. En linguistique de manière générale, et en TAL en particulier, comme dans la plupart des branches de l'intelligence artificielle,

1. Université Sorbonne Nouvelle – Paris 3.

CC BY ND

on est passé ces 30 dernières années de descriptions fondées sur des modèles formels/symboliques à base de règles (analyses syntaxiques avec des grammaires formelles, analyses sémantiques avec des représentations logiques...) à des études empiriques fondées sur l'usage (linguistique de corpus, statistiques textuelles, fouille de données réelles...). La linguistique est une discipline qui revendique un ancrage scientifique fort. Avec ma formation initiale en informatique, je n'ai pas eu de difficultés à comprendre les problématiques et les méthodologies de recherche de mes collègues linguistes. Et inversement, convaincre les linguistes qu'ils ont tout à gagner à acquérir de bonnes compétences en informatique est aussi relativement facile, au moins pour les linguistes professionnels. Mais les étudiants ont souvent des réticences vis-à-vis des mathématiques (ils se sentent plus « littéraires » que scientifiques), il faut en tenir compte.

Les initiations en licence sont généralement orientées « données » et « outils » : codage des textes, formats de fichiers, initiation à la statistique textuelle, requétage par expressions régulières, analyses linguistiques en ligne, utilisation d'étiqueteurs ou d'analyseurs syntaxiques... De nombreuses notions fondamentales de l'informatique peuvent déjà être illustrées par ce type d'enseignements. La programmation est abordée après coup, une fois constatées les limites des outils. Ces cours sont spécifiques. De fait, l'enseignement de la programmation que j'ai reçu ne serait pas du tout adapté à ce contexte. J'ai appris à programmer par les nombres, en implémentant des traitements qui illustraient les cours de mathématiques : calcul des termes d'une suite (Fibonacci et compagnie), triangle de Pascal, tri de nombres, récursivité de la factorielle, résolutions d'équations (polynômes, pivot de Gauss, dichotomie, méthode de Newton...), manipulation de vecteurs ou de matrices... autant de problèmes qui intéresseraient bien peu mes étudiants. L'enseignement de l'informatique en France est clairement l'héritier des mathématiques. Avec les linguistes, il faut s'appuyer sur un autre ancrage, quitte à faire passer certaines notions mathématiques en douce.

Un premier terrain traditionnel, plus « neutre » en termes disciplinaires, est fourni par le domaine des jeux. Les jeux sont des sources d'exercices et de projets quasi infinis, toujours motivants pour les étudiants. Ceux qui utilisent la langue sont nombreux : jeu du pendu, mots croisés/fléchés/mélangés, codes secrets, Scrabble (le comptage de la valeur d'un mot posé sur des cases « lettre compte double/triple » équivaut d'ailleurs à réaliser un produit scalaire entre deux vecteurs, mais on n'est pas obligé de le savoir pour le programmer), jeu des « chiffres et des lettres », MasterMind avec des lettres (où il faut découvrir un mot caché), jeux oulipiens... Seul imprévu rencontré lors d'examens : ces jeux à base de lettres sont inconnus des étudiants qui ont appris à écrire avec des caractères idéographiques, il ne faut pas négliger l'explication des règles...

De manière générale, pour enseigner la programmation aux linguistes, il est judicieux de considérer que la donnée fondamentale est moins le nombre que la chaîne

de caractères. J'évoque ci-dessous quelques types d'exercices permettant d'introduire certains concepts fondamentaux de la programmation :

— Conditions : exercices de conjugaisons ou de déclinaisons (pour des langues qui s'y prêtent) automatiques, et autres flexions (programmer la règle de mise au pluriel des noms communs en français, en tenant compte des différents cas particuliers et exceptions, par exemple), jeux sur les codes ASCII pour transformer des majuscules en minuscules et inversement...

— Boucles : jeu sur le codage/décodage de chaînes avec le codage de type « César » (décalage des lettres de l'alphabet, ce qui permet au passage d'introduire les modulo quand on dépasse 'z'), comptage de certains caractères/chaînes dans d'autres chaînes...

— Tableaux, structures de données complexes : compte d'occurrences de lettres/mots, stockage de dictionnaires monolingues/bilingues ou de métadonnées d'un livre. Par ailleurs, les structures d'arbres abstraits (syntaxiques) sont familières aux étudiants de sciences du langage, les manipuler leur est donc naturel.

— Algorithmes de tri, introduction à la complexité : le tri des livres d'une bibliothèque sera plus parlant pour ces étudiants que le tri d'un ensemble de nombres.

— Récursivité : au-delà des traditionnels palindromes, la plupart des programmes de manipulation de chaînes de caractères peuvent aussi s'écrire de manière récursive. Programmer des grammaires formelles peut bien sûr les séduire (mais, en dehors des langages réguliers, elles sont beaucoup plus faciles à utiliser en mode « génération aléatoire » qu'en analyse). Elles offrent l'occasion de faire le lien entre la récursivité de la langue (par exemple le fait que des groupes nominaux peuvent inclure d'autres groupes nominaux) et celle des programmes qui les modélisent. Ainsi, pour générer les phrases incluant toutes sortes de « liens familiaux » décrites par l'automate de la Figure 1, le mieux est d'écrire autant de fonctions que d'états dans l'automate, et de les faire s'appeler mutuellement.

Pour des projets plus complexes et complets, on peut par exemple proposer la programmation d'un correcteur orthographique à partir d'un dictionnaire de formes fléchies et du calcul de la distance d'édition entre chaînes de caractères, ou celle d'un petit moteur de recherche, en partant du comptage des mots dans un texte et du calcul de la « distance » entre deux textes à partir de leurs mots communs (distances de Jaccard, de Dice, proximité cosinus... tiens, les maths reviennent encore par la fenêtre). En master, dans ces formations, les cours de fouille de textes que je donne sont les mêmes que ceux que j'avais préparés pour les filières informatique.

Les goûts et dispositions des étudiants de sciences du langage pour ces enseignements sont évidemment variables. Mais leurs études les ont bien préparés, en les habituant à traiter les textes comme des séquences structurées de symboles. Ils ont

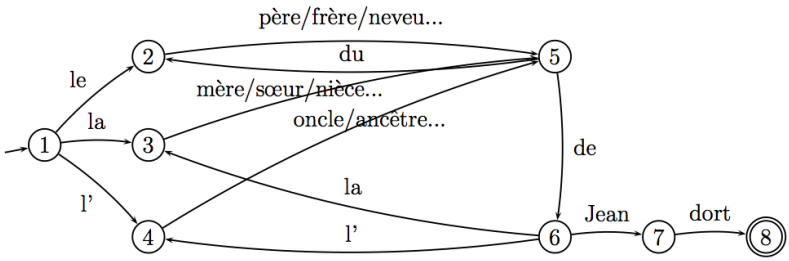

FIGURE 1. Automate décrivant un ensemble de phrases.

appris à les manipuler et à les analyser indépendamment de leur sens, c'est un point de départ précieux pour assimiler les bases de l'informatique. J'ai souvent constaté qu'ils sont sensibles, plus que les étudiants informaticiens, aux limites des outils de fouille de textes et d'ingénierie linguistique actuellement disponibles et savent mieux les critiquer. Ils rédigent aussi généralement leurs projets bien mieux qu'eux... Notons au passage que les ratios filles/garçons dans ces filières sont presque inverses de ceux observés en informatique (90 % vs. 10 %, environ...). À défaut de recruter des filles pour faire de l'informatique, je me réjouis au moins d'enseigner l'informatique aux filles là où elles sont, et de leur montrer qu'elles peuvent aimer ça ! Je peux ainsi attester que, chaque année, d'excellent(e)s étudiant(e)s se révèlent. Je connais plusieurs linguistes ayant effectué une thèse de TAL qui ont obtenu, in fine, leur qualification en section 27. Mais, en dehors de ces cas particuliers, il n'est pas douteux que l'informatique va prendre une place de plus en plus importante dans la formation des linguistes. L'expérience que je retranscris ici devrait aussi, à mon sens, encourager l'enseignement de la programmation à destination des littéraires : à condition de partir de leurs centres d'intérêts, ils peuvent se passionner pour le sujet.