LES CAHIERS DE L'ANALYSE DES DONNÉES

J.-P. BENZÉCRI
F. BENZÉCRI

Programmes de statistique linguistique fondés sur le tri par fusion de fichiers de texte

Les cahiers de l'analyse des données, tome 15, n° 1 (1990), p. 59-82

<http://www.numdam.org/item?id=CAD_1990__15_1_59_0>

Les Cahiers de l'Analyse des Données
Vol XV - 1990 - n°1 - pp. 59-82.

PROGRAMMES DE STATISTIQUE LINGUISTIQUE FONDÉS SUR LE TRI PAR FUSION DE FICHIERS DE TEXTE

[LING. TRI]

J.-P. & F. BENZÉCRI

0 Dénombrements linguistiques et principes algorithmiques

L'analyse des correspondances ayant été initialement conçue pour des applications à la linguistique, on ne s'étonnera pas que des tableaux croisant un ensemble de textes avec un ensemble de mots (ou de formes) aient été parmi les premiers traités à Rennes par B. Cordier (Madame Escoffier) avant 1965. Paru en 1981, le volume *Pratique de l'Analyse des Données en Linguistique et Lexicologie*, offre, en quelque 30 chapitres, un panorama des recherches effectuées sur ce thème.

Les tableaux analysés furent d'abord construits d'après les dénombrements effectués manuellement par les linguistes; tels ceux dont nous sommes redevables au regretté P. Guiraud, et que renferment ses *Indices du Théâtre Classique* . Cependant des équipes de précurseurs entreprenaient de constituer, avec l'aide de l'ordinateur, ce qu'on appellerait aujourd'hui des *Banques de données linguistiques.*

Sans prétendre faire un relevé exhaustif, nous citerons nos collègues et amis de l'*Index Thomisticus*, de Gallarate; du *Laboratoire d'Analyse Statistique des Langues Anciennes*, de Liège; du *Trésor de la Langue Française*, de Nancy; du *CATAB*, créé à Nancy et transféré à Lyon; ... Auxquels se sont adjoints, au fil des ans, les équipes de Louvain-la-Neuve, de Saint-Cloud, de Maredsous, de Nice ...

Un terme idéal serait l'élaboration complètement automatique de tous les tableaux que l'on peut désirer, à partir de textes, de dictionnaires et de grammaires d'abord saisis pour être soumis à l'ordinateur et élaborés ensuite par celui-ci. Nous sommes fort éloignés de ce terme! mais l'on peut citer, comme ayant contribué à nous en rapprocher, tous ceux qui, à l'instar du regretté B. Vauqois, de Grenoble, ont contribué à donner à la traduction automatique des bases scientifiques.

Les cahiers de l'analyse des données – 0339-3097/90/01 59 24/$ 4.40/

Plus accesible est le projet suivant: à partir d'un ou de plusieurs textes déjà saisis, effectuer des dénombrements de formes et construire des tableaux de correspondance. Ainsi, dans [ANA. LEX.], in *CAD*, Vol VI, n°2, pp. 229-243 (1981), L. Lebart décrit une *Procédure d'Analyse Lexicale écrite en langage FORTRAN*, et utilisée par lui, notamment pour le traitement des réponses libres aux questions ouvertes des enquêtes. Depuis lors, a paru, sur ce thème, un livre dû à L. Lebart et A. Salem.

Pour montrer en quoi les programmes existants manquent d'atteindre le *terme idéal*, nous dirons d'abord que ceux-ci dénombrent des formes et non des mots analysés: pour reprendre les exemples de L. Lebart, *blanche* est une autre forme que *blanches*; (et, ajouterons-nous, ces formes ne sont pas reconnues comme étant les féminins singulier et pluriel de l'adjectif *blanc*); tandis que *avoir* (nom) est une même forme que *avoir* (verbe).

Nous dirons ensuite que, les dénombrements se faisant en mémoire centrale, il faut, de quelque manière, réserver la place des formes; par exemple, comme dans [ANA. LEX.] prévoir, au plus, 30 formes d'une lettre, 90 de deux, 150 de trois, etc... Les formes retenues pour être dénombrées sont, ordinairement, les premières rencontrées; et le texte à traiter est lu, mot après mot, (ou plutôt *forme* après *forme*), chaque nouveau venu étant comparé au lexique résidant dans la mémoire centrale.

Pour s'affranchir de la première restriction, il faut s'avancer dans les voies de la syntaxe et de la sémantique: plusieurs logiciels de *traitement de textes* le font déjà empiriquement; mais on est loin de posséder la science de ces traitements; et nous ne proposons ici que des algorithmes susceptibles d'aider l'utilisateur compétent à incorporer, dans les fichiers créés automatiquement à partir des textes, des informations sur la valeur des occurrences des formes; e.g. *avoir* est nom ici, verbe là; *eurent* est une forme du verbe *avoir* ...

De la deuxième restriction, en revanche, nous nous affranchissons radicalement, en ne gardant en mémoire centrale ni liste ni tableau. En bref, le texte à traiter (supposé divisé en *chapitres* et *versets*; ce dernier terme nous venant du texte de l'*Évangile* qui a été notre premier exemple) est d'abord converti en une liste de formes, chacune écrite sur une ligne, et suivie, sur une autre ligne, de son adresse (numéros de chapitre et de verset). Puis, en appliquant l'algorithme de tri par fusion à des copies du fichier ainsi créé, on crée une liste des formes, rangées dans l'ordre alphabétique et suivies comme précédemment de leurs adresses (rangées elles-mêmes, pour chaque forme dans l'ordre croissant).

Il est clair qu'à partir d'une telle liste, construire un dictionnaire de formes, extraire de celui-ci une liste des formes à retenir, dénombrer par chapitre ou par versets les occurrences des formes retenues, ... ne sont que des opérations élémentaires.

Il nous reste seulement à démontrer que le tri par fusion, appliqué à 3 fichiers dont deux sont ouverts en lecture et un en écriture, est praticable, même sur un micro-ordinateur; l'avantage essentiel étant que, sans se soucier de gestion de mémoire, on peut créer tous les indices imaginables pour un texte, dans un temps qui ne croît, en fonction du nombre *n* des occurrences que comprend le texte, que comme n log(n)/log(2) (produit de *n* par log à base 2 de *n*); c'est-à-dire, à très peu près, linéairement; puisque le temps requis pour traiter un million d'occurrences n'est que 2000 fois celui pour mille occurrences.

Au §1, nous rappelons le principe du tri par fusion et donnons un programme réalisant ce tri sur 3 fichiers pour une suite de nombres entiers. Au §2, nous traitons de l'application du même algorithme de tri à un texte considéré comme une suite de formes. Au §3, nous présentons un ensemble de programmes conçus pour créer des tableaux de correspondance entre ensemble de textes (ou fragments de textes) et ensemble de formes. Au §4, nous traitons brièvement un exemple d'analyse.

1 Tri par fusion sur des fichiers d'entiers

1.1 Principe du tri par fusion

Dans sa version la plus simple, l'algorithme de tri par fusion part de deux listes données, dont chacune est déjà rangée (par exemple dans l'ordre croissant), et en crée une troisième, rangée également, et comprenant les éléments des deux premières. Il vaut la peine de donner de l'algorithme une description imagée, bien adaptée au traitement de fichiers que nous avons en vue.

Deux files d'hommes, fA et fB, se présentent à un guichet: dans chacune des files, les hommes sont rangés par taille croissante, les petits devant, les grands ensuite. Le guichetier fait passer l'homme de tête de la file fA, s'il est plus petit que celui de fB (ou de même taille que lui); sinon, il fait passer l'homme de tête de fB. Il est clair qu'ainsi, après le passage du guichet, les individus de fA et de fB constituent une file unique fC, bien rangée.

On peut, en appliquant itérativement le même principe ranger les éléments d'une suite qui n'est, initialement, aucunement triée. Reprenons l'exemple d'une file d'hommes. Cette file peut être considérée comme une suite de paires (avec, éventuellement, un dernier individu non apparié). On demande d'abord au second individu de chaque paire de passer devant le premier s'il est strictement plus petit que celui-ci. On a ainsi une suite de paires triées; et l'on convient que ces paires sont associées 2 par 2: la 1-ère avec la 2-ème; la 3-ème avec la 4-ème; la 5-ème avec la 6-ème; ... On demande alors à chaque couple de paires associées de s'interclasser, en suivant la méthode donnée plus haut pour fA et fB. On a ainsi une suite de quadruplets bien rangés; etc... Finalement, c'est l'ensemble de la suite qui est bien rangée.

Cette manœuvre de petites files d'hommes qui s'entrecroisent peut sembler confuse! et il vaut mieux désormais (en attendant les mots, objets du §3) parler de nombres qu'il n'est pas utile de déplacer; mais qu'il suffit de copier et d'effacer.

1.2 Le programme de tri

L'on suppose effectuée une copie, nommée 'nomf', du fichier de nombres à ranger. Après avoir vérifié l'existence de cette copie, le programme l'ouvre, sous l'adresse symbolique tVr, et en crée deux copies, dont les noms de fichiers sont obtenus en ajoutant respectivement les lettres 'a' et 'b' au nom de fichier nomf; et qui seront ouvertes aux adresses respectives Vra et Vrb. Désormais, pour suivre les notations du programme, nous parlerons en bref de trois fichiers tVr, Vra et Vrb.

Le programme, destiné au micro-ordinateur Macintosh, consiste en une imbrication de boucles dont nous expliquons ici la structure hiérarchique. Le paramètre supérieur est *la*, ou longueur des blocs déjà bien rangés et que l'on interclasse. Au départ, *la* vaut 1; quand les blocs de longueur *la* ont été interclassés, on a des blocs bien classés de longueur 2**la*; donc on double *la*; et ainsi de suite jusqu'à ce que *la* soit ≥ au nombre total *cr* des entiers à ranger.

Pour décrire l'interclassement, partons du cas où *la* = 1. On lit dans Vra les entiers de rang impair, en sautant les entiers de rang pair; au contraire, dans Vrb, on lit les entiers de rang pair et on saute ceux de rang impair. Quand nous disons "sauter", il faut comprendre que des entiers sont lus, pour avancer dans le fichier, mais ne sont soumis à aucun traitement; tandis que les entiers lus intentionnellement sont destinés à être recopiés dans tVr, en un ordre convenable.

Quant *la* = 1, c'est simple: l'entier de tête, *ia*, lu dans Vra, est copié en tête de tVr, s'il est ≤ au deuxième entier, *ib*, lu dans Vrb; puis on copie celui-ci. Sinon, c'est le deuxième entier de Vrb qui est mis en tête de tVr, suivi du 1-er entier de Vra. On fait ensuite de même pour le 3-ème entier, lu dans Vra, et le 4-ème entier, lu dans Vrb; et ainsi de suite, jusqu'à parvenir à la fin des fichiers Vra et Vrb. On a alors un fichier tVr dont les paires successives sont bien classées; et ce fichier est recopié à la place de Vra et de Vrb.

La gestion des priorités d'écriture est plus complexe si *la* > 1. Nous tenons alors compte de 3 indices: *ina*, rang du premier individu du bloc de Vra en cours de lecture; et *na* et *nb*, rangs respectifs des entiers, *ia* et *ib*, que l'on vient de lire sur Vra et Vrb et qu'on doit comparer pour décider lequel des deux est l'entier *it* à copier sur tVr. Il faut prendre garde à ne pas dépasser en lecture la longueur *la* des blocs en cours de traitement: si, par exemple, le bloc pris sur Vra est entièrement copié sur tVr, on doit copier jusqu'à épuisement le bloc pris sur Vrb, sans avoir à effectuer de comparaison. Une autre particularité est qu'à la fin d'une itération en *la*, le dernier bloc de Vrb peut avoir une longueur *lb* < *la*; ou,

```
program trif(input,output);uses macintf,uver;
var i,ia,ib,it,fa,lu,er1:integer;cr,ina,la,lb,n,na,nb,nz,x:longint;
nomf,nom:string;rpc:char;
Vra:file of integer;Vrb:file of integer;tVr:file of integer;
begin er1:=0;rpc:='N';
while not((rpc='O')or(er1=6)) do begin
  write('le nom du fichier d entiers a trier est ');readln(nomf);
  rpc:=dialof(stringptr(@nomf),output);
  if (rpc='O') then begin
      write('ce nom est il confirme O ou N ');readln(rpc) end
    else er1:=er1+1 end;
if (rpc='O') then begin
 reset(tVr,nomf);cr:=0;
 rewrite(Vra,concat(nomf,'a'));rewrite(Vrb,concat(nomf,'b'));
 while not eof(tVr) do begin read(tVr,it);cr:=cr+1;
   write(Vra,it);write(Vrb,it);
   write(it:7);if (cr mod 10=0) then writeln end;
 close(Vra);close(Vrb);close(tVr);
 writeln;writeln('cr =',cr:10);
 la:=1;ina:=1;
 while (ina+la<=cr) do begin
   if(ina=1) then begin
     rewrite(tVr,concat(nomf,'t'));
     reset(Vra,concat(nomf,'a'));
     reset(Vrb,concat(nomf,'b')) end;
   x:=cr-(ina+la-1);if(x<la)then lb:=x else lb:=la;
   na:=ina;read(Vra,ia);
   nb:=ina+la;for i:=1 to la+1 do read(Vrb,ib);
   for n:=ina to (ina+la+lb)-1 do begin
       if(na<ina+la) and (nb<ina+la+lb) then
          if(ia<=ib) then begin it:=ia;na:=na+1;
       if not(na=ina+la) then read(Vra,ia) end
            else begin it:=ib;nb:=nb+1;
       if not(nb=ina+la+lb) then read(Vrb,ib) end
        else if(ina+la=na) then begin it:=ib;nb:=nb+1;
       if not(nb=ina+la+lb) then read(Vrb,ib) end
            else begin it:=ia;na:=na+1;
       if not(na=ina+la) then read(Vra,ia) end;
       write(tVr,it);write(it:7);if (n mod 10=0) then writeln end;
   for i:=1 to lb do read(Vra,ia);
   ina:=ina+la+lb;
   if (cr<ina+la) then begin n:=ina-1;
     while not eof(Vra) do begin read(Vra,it);n:=n+1;
       write(tVr,it);write(it:7);if (n mod 10=0) then writeln end;
     la:=2*la;ina:=1;
     writeln;writeln('la =',la:8);
     if (la<cr) then begin
       close(tVr);reset(tVr,concat(nomf,'t'));
       close(Vra);rewrite(Vra,concat(nomf,'a'));
       close(Vrb);rewrite(Vrb,concat(nomf,'b'));
       while not eof(tVr) do begin
         read(tVr,it);write(Vra,it);write(Vrb,it) end;end;
     close(tVr);close(Vra);close(Vrb) end;end;
end;end.
```

parfois, le fichier Vra se termine par un bloc qu'il faut recopier sur tVf, tel quel; car il n'est apparié à aucun bloc de Vrb (ce fichier étant déjà lu jusqu'à la fin).

Quand se termine une itération en *la*, on a un fichier tVr dont les blocs successifs de longueur 2**la* sont bien classés; et ce fichier est recopié à la place de Vra et de Vrb. Finalement (ce qui n'est pas précisé dans le présent listage) les fichiers Vra et Vrb sont effacés (i.e. éliminés du catalogue du disque utilisé).

Si l'on fait le tri par fusion, non sur des fichiers ouverts en lecture et écriture, mais sur des tableaux de nombres gardés en mémoire centrale, on n'a besoin que de 2 tableaux, au lieu de 3 fichiers. On pourrait aussi substituer aux 2 fichiers Vra et Vrb un fichier Vr unique, ouvert en accès direct (i.e. sur lequel on lirait à la fois les deux blocs à interclasser): mais cette simplification (si tant est que c'en soit une) ne sera pas permise dans la suite; car nous traiterons, non des fichiers d'entiers tous enregistrés sur 2 ou 4 octets, mais des mots de longueur variable.

2 Tri par fusion sur les mots d'un texte

Nous considérerons successivement la préparation du texte ; puis les modifications à apporter à l'algorithme du §1.2; en énumérant les variantes du programme de tri des formes.

2.1 Préparation du texte

Le texte est d'abord saisi sur un éditeur, tel que MacWrite, MacAuteur, ...; avec, en tête, une ligne de titre qui ne sera pas prise en compte dans le tri. Et l'on crée un premier fichier en format de 'texte seul'; (c'est-à-dire une suite d'octets, numéros des caractères successifs - lettres, signes de ponctuation, aller à la ligne - dans le code ASCII; sans aucune des indications propre à l'éditeur, telle que taille des caractères, style gras ou italique, choix d'une police). Nous supposerons, pour fixer les notations, qu'il s'agit du texte grec de l'Évangile selon Saint Jean; et que le fichier en 'texte seul' a pour nom 'D:Jx' (où 'D' désigne le disque, ou le dossier, utilisé).

Avant d'être soumis au programme 'trim§' de tri par fusion d'une liste de mots, le texte 'D:Jx' doit être converti en un autre fichier de texte 'D:jx§' où chaque mot (forme) du texte initial est écrit seul sur une ligne; avec, sur la ligne suivante, le numéro du chapitre et celui du verset.

Il va sans dire que, dans un autre texte, les chapitres pourront être subdivisés en paragraphes; et qu'au lieu de chapitres d'un même texte, il pourra s'agir d'une suite de textes différents: par exemple, comme dans l'application considérée par L. Lebart, des textes de réponses données par un ensemble de sujets à une même question ouverte. Et les versets jouent ici le rôle de ce que M. Reinert appelle des "unités de contexte", c'est-à-dire des segments successifs au sein desquels il est loisible de dénombrer les mots.

Pour préparer le fichier 'D:Jx§', on doit convenir de certaines règles. Il faut fixer les caractères ayant valeur de séparateur entre mots: blancs, aller à la ligne, signes de ponctuation. Afin que la comparaison de deux mots quant à l'ordre

lexicographique se fasse directement sur les numéros des caractères dans le code ASCII, on transcrit toutes les capitales en minuscules.

Il faut ensuite traiter les caractères doubles et les signes diacritiques; et cela, de telle sorte que le tri produise, aussi rapidement que possible, des listes rangées dans l'ordre alphabétique usuel. Comme les conventions diffèrent selon la langue, nous utilisons présentement plusieurs programmes de conversion (ou de mise en forme): 'forgrc§', pour le grec; 'forgal§', pour le français; 'forlat§' pour le latin ...

En français, on dispose assez communément de caractères doubles: *œ*, *æ*; ainsi que *fi* et *fl*. Ces caractères sont éclatés en deux: *oe*, *ae*, *fi*, *fl*. Il y a de nombreuses lettres accentuées: *à*, *é*, *è*, *ô*, ... Nous convenons, présentement, pour plus de simplicité, de remplacer celles-ci par des lettres non accentuées: *a*, *e*, *o*. De même *ç* devient *c*. En espagnol, *ñ* constitue une lettre unique, qui suit le n dans l'ordre alphabétique: il se trouve que, compte tenu du numérotage ASCII, l'ordre alphabétique des mots est respecté si l'on éclate *ñ* en *n~*.

En grec, dans la tradition littéraire classique, les mots sont affectés de nombreux signes diacritiques, appelés "esprits" ou "accents", avec éventuellement deux signes sur une seule lettre. Dans la police de caractères que nous utilisons (police dérivée de la police "Alexandria" distribuée avec MacAuteur), les signes diacritiques (et les paires de signes destinées à un seul caractère) sont considérés comme des caractères isolés qui ne diffèrent des caractères usuels qu'en ce qu'ils s'impriment tout entier en arrière de leur point d'insertion, au dessus du caractère précédent (comme le font l'accent circonflexe et le tréma sur une machine à écrire usuelle). Nous avons choisi, pour accélérer le programme de tri, de supprimer tous les caractères diacritiques. (Certains de ces caractères ont été introduits dans une version ultérieure: cf. [NOUV. TEST. GREC], §2.1, in *CAD*, Vol XV, n°1.)

Quant aux caractères doubles ou en ligatures, si nombreux dans les polices anciennes, telles que le *Grec du Roy*, ils sont absents de notre police.

Reste le numérotage des chapitres et versets. Notre convention est d'introduire, dans le texte initial, le numéro de verset en tête de chaque verset, sans noter explicitement le numéro du chapitre. Le programme de conversion vérifie que les numéros de versets se suivent dans l'ordre naturel: {1, 2, 3, 4, ...} et accepte seulement des retours à 1, comme signal d'un nouveau chapitre. Le numéro du chapitre d'un mot est ainsi pris égal au nombre total des '1' rencontrés avant ce mot. Tant que le programme de conversion ne rencontre pas de numéro, il se borne à attribuer aux formes 0 pour numéros de chapitre et de verset.

Toutes ces conventions apparaîtront au lecteur sur une page qui offre, côte à côte, le début du texte original, et des extraits de listes de formes, triées ou non.

ΚΑΤΑ ΙΩΑΝΝΗΝ

1 Ἐν ἀρχῇ ἦν ὁ Λόγος, καὶ ὁ Λόγος
ἦν πρὸς τὸν Θεόν, καὶ Θεὸς ἦν ὁ
Λόγος. 2 οὗτος ἦν ἐν ἀρχῇ πρὸς τὸν
Θεόν. 3 πάντα δι' αὐτοῦ ἐγένετο, καὶ
χωρὶς αὐτοῦ ἐγένετο οὐδὲ ἓν ὃ
γέγονεν. 4 ἐν αὐτῷ ζωὴ ἦν, καὶ ἡ ζωὴ
ἦν τὸ φῶς τῶν ἀνθρώπων. 5 καὶ τὸ
φῶς ἐν τῇ σκοτίᾳ φαίνει, καὶ ἡ
σκοτία αὐτο οὐ κατέλαβεν.

Ci-dessus, début du texte de l'Évangile selon Saint Jean tel qu'il a été saisi; on notera la ligne de titre et les numéros en tête de chaque verset.

Ci-contre, sur une colonne, début de la liste de formes, 'B:Jx§', créée par le programme 'forgrc§'; on remarquera que les lettres avec ιοτα souscrit ont été remplacées par des lettres simples.

Plus à droite, également sur une colonne, un fragment de la liste triée de formes, 'B:Jx§t', créée par le programme 'trim§'; ce fragment a été choisi parce qu'il comporte toutes les occurrences de plusieurs formes de fréquence 1, 2 ou 3.

εν	αγαπησει
1 1	14 23
αρχη	αγαπησω
1 1	14 21
ην	αγαπω
1 1	14 31
ο	αγαπων
1 1	14 21
λογοσ	αγαπων
1 1	14 21
και	αγαπων
1 1	14 24
ο	αγγελοσ
1 1	5 4
λογοσ	αγγελοσ
1 1	12 29
ην	αγγελουσ
1 1	1 51
προσ	αγγελουσ
1 1	20 12
τον	αγγελουσα
1 1	20 18
θεον	αγιαζω
1 1	17 19
και	αγιασον
1 1	17 17
θεοσ	αγιε
1 1	17 11
ην	αγιον
1 1	14 26
ο	αγιον
1 1	20 22
λογοσ	αγιοσ
1 1	6 69
ουτοσ	αγιω
1 2	1 33

2.2 L'algorithme de tri des formes

On modifie le programme du §1.2. Au lieu de trois fichiers d'entiers, on a trois fichiers de texte: Vra, Vrb, tVr. Aux ordres de lecture et d'écriture d'entiers *ia*, *ib*, *it*, on a des ordres de lecture et d'écriture de mots *ma*, *mb*, *mt*. Plus précisément, à un ordre "read(Vra,ia)", il correspond deux ordres successifs "readln(Vra,ma)" et "readln(Vra,qa)". C'est-à-dire qu'on lit d'abord, sur une ligne, la forme *ma*; puis, sur la ligne suivante, le couple de numéros *qa*, lequel

```
if (ma=mb) then acb:=1 else acb:=0;
lga:=length(ma);lgb:=length(mb);u:=0;w:=1;asb:=0;
if (acb=0) then while not(w=0) do begin w:=0;
  if (lga=u) then aaa:=0 else aaa:= ord(ma[u+1]);
  if (lgb=u) then bbb:=0 else bbb:= ord(mb[u+1]);
  if (aaa>bbb) then asb:=1;
  if (aaa=bbb) and not(aaa=0) then begin w:=1;u:=u+1 end end;
```

procédure de comparaison des mots suivant l'ordre lexicographique

ne participe pas au tri, mais est recopié sur le fichier tVr immédiatement après *ma*, quand l'algorithme ordonne de copier cette forme.

Quant à la comparaison des formes suivant l'ordre lexicographique, on l'effectue par un bloc d'instructions qui permet de remplacer par (asb=0) la condition (ia<=ib) utilisée dans le tri des nombres: asb=0 signifie ma=mb ou ma précède mb dans l'ordre alphabétique.

Aux détails près que nous avons signalés, le programme 'trim§' ne diffère pas de celui du §1.2.

Sur le listage 'B:Jx§t', les formes sont recopiées dans l'ordre alphabétique; chacune est répétée autant de fois qu'elle a d'occurrences dans le texte 'B:Jx'; avec les adresses rangées dans l'ordre croissant, par chapitres et versets. Ainsi, on a trois occurrences de αγαπων, ("aimant", ou "qui aime"), toutes trois dans le chapitre 14; deux dans le verset 21 et une dans le verset 24. Le listage trié 'B:Jx§t' commence par une longue liste de α (pronom relatif neutre pluriel, souvent traduit par "ce que").

Le programme 'trim§' se déroule en affichant à l'écran ce qui s'écrit sur le fichier en cours d'élaboration. L'utilisateur est ainsi averti du bon déroulement du tri, et peut même saisir au vol des bribes de résultats. Cependant, l'affichage *in extenso* augmente de 50% le temps d'exécution du tri. Quand on est assuré de la correction d'un programme, on a avantage à restreindre l'affichage.

Le programme 'trimu§' diffère de 'trim§' en ce qu'il est muet; plus précisément, 'trimu§' n'affiche que le nombre total *cr* d'occurrences du texte traité et la longueur *la* des blocs sur lesquels porte la fusion. Quand, sous une ligne 'la = 16' on voit s'afficher la ligne 'la = 64', on est averti de l'entrée dans une nouvelle itération. De plus, si, e.g., *cr* vaut 1372, on sait que la dernière itération sera avec 'la = 1024' (et consistera en la fusion de deux blocs bien classés, le premier de longueur 1024, le second de longueur 1372-1024 = 348): ceci permet d'estimer le temps nécessaire pour achever le tri, compte tenu de ce que toutes les itérations ont, à peu près, la même durée.

Sur l'ensemble de l'Évangile selon Saint Jean, qui compte environ 16000 occurrences, le programme 'trimu§' a achevé le tri en 3 heures et demi.

Une autre variante, 'trim', de 'trim§', traite des listes de formes (ou chaînes quelconques de caractères) ne comportant pas de lignes intercalées indiquant les numéros de chapitre et verset. En fait, on peut remarquer que, puisque 'trim§' ne s'enquiert aucunement du contenu des lignes intercalées qu'il copie, on peut ranger dans celles-ci des informations de toute sorte; et non seulement des numéros;. Nous préciserons, au §3.3 comment cela permet de créer des concordances.

Au contraire, le programme 'triml' tient explicitement compte des numéros de chapitre et verset: il permet donc de traiter une liste dans laquelle les formes ne sont pas dans l'ordre naturel du texte et de ranger dans cet ordre chaque séquence de formes identiques. On peut ainsi reprendre des listes de formes qui ont été modifiées, par exemple pour ramener toutes les occurrences d'un verbe à la forme de l'infinitif; éventuellement grâce au programme de transformation 'trans§' (cf. §3.3.3).

3 Correspondance entre formes et chapitres ou versets

Nous considérerons successivement la création du lexique des formes et l'élaboration de ce lexique afin d'en extraire un ensemble de formes qu'on croisera avec l'ensemble des chapitres ou celui des versets; la création du tableau de correspondance lui-même; et enfin divers programmes, qui, sans aller jusqu'au terme idéal de l'analyse syntaxique et sémantique, aident à incorporer aux données analysées la connaissance linguistique qu'en a l'utilisateur des programmes.

3.1 Création et élaboration du lexique

En bref, le lexique, 'B:Jx$§', est la liste des formes se rencontrant au moins une fois dans le texte 'Jx', avec, pour chacune de celles-ci, l'indication de sa fréquence (plus précisément du nombre de ses occurrences dans 'Jx'). Le programme 'qamus§' passe donc aisément de la liste ordonnée des occurrences, 'B:Jx§t', au lexique: il nous suffit d'indiquer sous quelle forme celle-ci est présentée.

Le lexique 'B:Jx$§' est un texte; autrement dit, il peut se consulter comme un listage usuel; chaque forme est écrite isolée sur une seule ligne; et sa fréquence est donnée sur la ligne *précédente*. Cette dernière particularité permet d'obtenir un lexique des formes triées *par fréquence croissante*, en soumettant simplement 'B:Jx$§' au programme 'trim§'.

De façon précise, le lexique trié 'B:Jx$§t' donne d'abord la liste alphabétique des *hapax* (ou formes employées une seule fois dans 'Jx'), chacune précédée de nombre '1' écrit sur une ligne; puis la liste alphabétique des formes de fréquence 2, chacune précédée, de même, du nombre '2', écrit sur une ligne; et ainsi de suite, jusqu'à la forme καὶ (conjonction de coordination 'et') dont la fréquence est 831.

0030	0001	0148	αβρααμ
α	αγαγειν	οι	αιωνα
0011	0001	0150	αιωνιον
αβρααμ	αγαθα	η	αληθεια
0001	0001	0150	αληθησ
αγαγειν	αγαθον	ουκ	αλλοι
0001	0001	0153	αμην
αγαθα	αγαθοσ	το	ανθρωπον
0001	0001	0166	ανθρωποσ
αγαθον	αγαλλιαθηναι	εστιν	ανθρωπου
0001	0001	0169	απεκριθη
αγαθοσ	αγαπηθησεται	αυτου	απεκριθησαν
0001	0001	0172	απεστειλεν
αγαλλιαθηναι	αγαπησασ	αυτω	απηλθεν
0003	0001	0175	αποκτειναι
αγαπα	αγαπησει	αυτον	αρτον
0002	0001	0198	αρτοσ
αγαπασ	αγαπησω	ινησουσ	αρχιερεισ
0005	0001	0202	βασιλευσ
αγαπατε	αγαπω	ουν	γαλιλαιασ
0004	0001	0206	γραφη
αγαπη	αγγελουσα	εισ	γυνη
0001	0001	0210	.....
αγαπηθησεται	αγιαζω	δε	υδωρ
0003	0001	0237	υιον
αγαπην	αγιασον	τον	υιοσ
0001	0001	0238	υμασ
αγαπησασ	αγιε	εν	υμεισ
0001	0001	0245	υμιν
αγαπησει	αγιοσ	του	υμων
0001	0001	0270	υπαγω
αγαπησω	αγιω	οτι	φαρισαιοι
0001	0001	0610	φωσ
αγαπω	αγνισωσιν	ο	χριστοσ
0003	0001	0831	ψυχην
αγαπων	αγορασον	και	ωρα

Ci-dessus, sur quatre colonne, le lexique et son élaboration
à gauche: début du listage 'B:Jx$§' des formes rangées alphabétiquement;
au centre, sur deux colonnes, début et fin de 'B:Jx$t':
début de la liste des hapax, et fin de la liste des mots les plus fréquents;
à droite, sur une colonne, début et fin du dictionnaire, 'B:dijt', des formes retenues.

Il reste seulement un détail à préciser: afin que les indications numériques de fréquence, que comporte le lexique 'B:Jx$§', puissent être triées par 'trim§', qui est un programme triant des mots, les fréquences sont écrites avec des zéros

à gauche jusqu'à concurrence de 4 chiffres.

D'après la liste, 'B:Jx$§', des formes triées par fréquence, on peut commodément décider du choix des formes qui seront retenues pour construire un tableau de correspondance. Les hapax doivent évidemment être écartés, au moins en tant que formes. Ils ne pourraient être conservés qu'en les cumulant avec d'autres formes d'un même mot: par exemple, {αγαθα, αγαθον, αγαθοσ} sont 3 formes de la déclinaison du même adjectif "bon". Nous reviendrons, au §3.3, sur la réalisation éventuelle de telles transformations de formes en leur forme de base (e.g., nominatif masculin singulier pour un adjectif).

À l'autre extrémité de la liste, il est raisonnable d'écarter les mots outils, dont les fréquences sont très élevées; à moins qu'on ne s'intéresse précisément aux variations du style grammatical de l'œuvre d'un chapitre à l'autre. Dans le présent essai, nous n'avons retenu, d'entre les mots outils, que les pronoms de la 1-ère et de la personne: car ceux-ci permettent de distinguer les dialogues. Ainsi, à la fin de la liste alphabétique, 'B:dijt', des formes retenus, on remarque les quatre cas du pronom "vous": {υμασ, υμεισ, υμιν, υμων}.

Dans la version de l'analyse dont les résultats font l'objet du §3, on a conservé toutes les formes de mots pleins dont la fréquence dépasse 9; et aussi les noms propres. Toutefois, le nom d'αβρααμ, qui ne se trouve que dans le chapitre 8 (où il figure 11 fois), a été mis en supplémentaire dans l'analyse de correspondance par chapitres. La forme retenue la plus fréquente se trouve être - à tout Seigneur tout honneur - le nom même de ιησουσ, dont on a relevé 198 occurrences.

Il est clair que les résultats obtenus ne peuvent que dépendre du choix des formes conservées; et le but des analyses est précisément de découvrir sur quelles tranches du vocabulaire d'une langue il faut se baser, pour faire ressortir les divers caractères, de style, de genre littéraire, de thème, ou même d'auteur.

Quant aux opérations informatiques, on a d'abord modifié, sur le traitement de texte 'EDIT', le listage des formes triées par fréquence 'B:Jx$§t', en ne laissant subsister que les lignes afférentes aux formes qui satisfaisaient aux critères précisés ci-avant. Ce listage modifié (où subsistaient, à côté des formes proprement dites, des indications de fréquence que nous n'avions pas effacées, de 0009 à 0198) a été sauvegardé sous le nom de 'B:dij'; le sigle 'dij' ayant été choisi pour rappeler qu'il s'agit d'un dictionnaire pour Saint Jean.

Le fichier 'B:dij' a été soumis au programme 'trim': ainsi, on obtenu un listage trié, 'B:dijt', où étaient rangées alphabétiquement, les nombres d'abord, puis les formes, d'αβρααμ à ωρα. On a éliminé les nombres: et sauvegardé, sous le même nom de 'B:dijt', la seule liste alphabétique des mots dont les occurrences, par chapitres, sont tabulées dans le listage 'B:Jxdij§corC' que nous décrirons maintenant avant d'en expliquer la création.

```
mots de dij x chapitres de B:Jx
21
ch@A ch@B ch@C ch@D ch@E ch@F ch@G ch@H ch@I ch@J ch@K ch@L ch@M ch@N ch@O ch@P
ch@Q ch@R ch@S ch@T ch@U
αβρααμ
   0    0    0    0    0    0    0   11    0    0    0    0    0    0    0    0
   0    0    0    0    0
αιωνα
   0    0    0    1    0    2    0    4    0    1    1    1    1    1    0    0
   0    0    0    0    0
αιωνιον
   0    0    3    2    2    4    0    0    0    1    0    1    0    0    0    0
   1    0    0    0    0
αληθεια
   1    0    0    2    1    0    0    3    0    0    0    0    0    1    0    0
   3    2    0    0    0
αληθης
   0    0    1    0    2    2    1    4    0    0    0    0    0    0    0    0
   0    0    0    0    1
αλλοι
   0    0    0    1    0    0    2    0    3    1    0    1    0    0    0    0
   0    1    0    1    2
αμην
   2    0    6    0    6    8    0    6    0    4    0    2    8    2    0    4
   0    0    0    0    2
```

3.2 Construction du tableau de correspondance

Le tableau de correspondance 'B:Jxdic§corC' est créé sous forme de texte, que l'on peut consulter, en respectant le format d'écriture que requiert le programme 'qori' d'analyse des correspondances.

En tête, on a une ligne de titre; puis, sur une autre ligne, le nombre des colonnes, soit 21, nombre des chapitres de l'évangile selon Saint Jean. Viennent ensuite les sigles des colonnes, de ch@A à ch@U. On reconnaît dans 'ch', l'initiale de "chapitre"; et, de 'A' à 'U', on a les 22 premières lettres de l'alphabet. Le nombre des chapitres pouvant aller jusqu'à 300, il est prévu de poursuivre le numérotage au-delà de 26 (lettre 'Z') en utilisant une numération de base 27 dont '@' est le zéro: ainsi, 'A@' ≈ 27; 'AA' ≈ 28; ...;'J@' ≈ 270.

On notera que, compte tenu des blancs de séparation, il tient 16 sigles dans une ligne de 80 caractères: cette disposition est conservée pour écrire, éventuellement sur plusieurs lignes, le dénombrement par chapitres des occurrences de chaque forme (c'est-à-dire ce qu'on appelle "ligne" du tableau de correspondance).

Initialement, les mots sont écrits *in extenso*, même si leur longueur dépasse 4 lettres: on sait que, dans ce cas, le programme 'qori' traite comme des commentaires les lettres surnuméraires et arrête le sigle à 4 lettres. Si, comme entre αληθεια, vérité, et αληθης, véridique, une confusion est à éviter, on peut modifier le tableau, en écrivant, e.g.: αλ91 αληθεια puis αλ92 αληθης.

On expliquera maintenant comment procède le programme 'tridic§'.

Le programme utilise pour données principales d'une part, la liste triée des formes d'un texte, avec leurs adresses, (dans notre exemple la liste 'B:Jx§t'); et, d'autre part, un lexique, ou liste de mots rangés dans l'ordre alphabétique (dans notre cas: 'B:dijt'). Il importe de signaler que le lexique peut être d'une origine quelconque: compilé sur le texte même qu'on étudie, ou issu de dénombrements faits sur un autre texte, voire sur un vaste corpus. La seule condition est que le lexique soit trié alphabétiquement, désigné par un sigle (nom de 4 caractères au plus) suivi de la lettre 't' (qui rappelle l'opération de tri; et est systématiquement ajoutée par le programme 'trim' utilisé ci-avant); et placé dans le même dossier que le texte sur lequel on effectue les dénombrements.

Le programme 'tridic§' offre deux options, cumul par chapitre et cumul par versets. Dans les deux cas, pour créer l'en-tête du tableau de correspondance, 'tridic§' doit connaître le nombre des chapitres du texte de base. C'est le moment pour nous de signaler que le programme 'forgrc§' (comme les programmes de mise en forme adaptés aux autres langues: 'forlat§', 'forgal§', 'forhsp§', ...; cf. §2.1) crée, outre une liste de formes (avec une forme par ligne et l'adresse sur la ligne suivante: cf. 'B:Jx§'), un petit fichier numérique (B:Jx§num, dans notre cas), contenant une suite d'entiers qui ne sont autres que les nombres de versets des chapitres successifs du texte (soit, dans le cas de l'Évangile selon Saint Jean: {51, 25, 36, ..., 25}).

Le programme vérifie que le nombre des colonnes prévues ne dépasse pas 300, et attribue à celles-ci des sigles. On a déjà expliqué les sigles des chapitres; si les dénombrements se font par versets, on attribue à chacun de ceux-ci un sigle dont les deux premiers caractères donnent, en base 27 (cf. *supra*), le numéro du chapitre; et les deux suivants le numéro du verset, en base 10: par exemple, le verset 5 du chapitre 2 serait '@B05'; le v. 20 du ch. 21: '@U20'. (L'usage automatique du code ASCII permettrait d'aller, au besoin, au-delà de 99 versets, en prenant le signe ':' pour désigner 10 dizaines).

L'algorithme de 'tridic§' n'est qu'une variante du tri par fusion. On lit simultanément les deux fichiers 'B:Jx§t' et 'B:dijt' de telle manière que l'on s'arrête sur toutes les fomes présentes à la fois dans le texte et dans le lexique. Pour ne pas en laisser échapper, il suffit de ne glisser dans la lecture d'un fichier que si la forme en lecture précède strictement, dans l'ordre alphabétique, celle lue sur l'autre fichier.

De façon précise, 'tridic§' crée deux fichiers de texte: d'une part un tableau de correspondance dont le nom est composé en ajoutant à celui du texte le sigle du dictionnaire, puis le suffixe '§cor' suivi de l'une des deux lettres 'C' ou 'V', selon que les dénombreements se font par chapitres ou par versets (dans notre cas on a: 'B:Jxdij§corC'); et d'autre part une liste de formes avec leurs adresses, qui n'est autre que la liste des occurrences du texte, dont a supprimé les formes

absentes du lexique; (dans notre cas on a'B:Jxdij§t', qui est donc un sous-fichier de 'B:Jx§t').

Il est facile d'imaginer comment 'tridic§', quand il rencontre une forme présente simultanément dans le texte et le lexique, procède pour traiter le bloc correspondant du listage des formes du texte. Ce bloc est purement et simplement recopié dans 'B:Jxdij§t'; et les nombres d'occurrences, cumulées par chapitres ou par versets, sont écrits dans 'B:Jxdij§corC' (ou: 'B:Jxdij§corV'), en interposant des zéros s'il est nécessaire.

Si une forme du lexique est absente du texte, on peut, ou non, selon la spécification de l'utilisateur, créer dans le tableau de correspondance une ligne de zéros.

3.3 Programmes complémentaires

Pour construire, à partir du texte grec de l'Évangile selon Saint Jean, le tableau de correspondance 'B:Jxdij§corC' (entre formes lexicales et chapitres) dont l'analyse fait l'objet du §4, on a utilisé seulement 5 programmes: 'forgrc','trimu§', 'qamus', 'trim' et 'tridic§'. D'autres programmes peuvent faciliter le dépouillement de textes plus longs, ou aider à préciser la valeur syntaxique et sémantique des occurrences de certains mots ambigus.

3.3.1 Fusion entre fichiers de formes déjà triés

Si l'on utilise un Macintosh+, et non un modèle plus rapide tel que SE/30 ou IIx, le temps requis pour passer d'un fichier de 100000 formes, 'B:tx§', au fichier trié correspondant, B:tx§t', est d'une vingtaine d'heures. On peut craindre d'entreprendre un aussi long calcul. Or, puisque nous ne faisons que des tris par fusion, il est possible de fractionner le traitement en plusieurs étapes, chacune intéressante pour elle-même, et produisant des listes triées qui sont ensuite interclassées par fusion.

De façon précise, supposons que le texte de base 'B:tx' comprend 20 chapitres; on peut le séparer en deux fichiers, 'B:tax' et 'B:tbx', comprenant respectivement les chapitres 1 à 5 et 6 à 10. Par 'forgrc§' (ou un programme semblable propre à une autre langue) on construit les fichiers de formes (avec adresses après chaque forme): 'B:tax§' et 'B:tbx§'. Il faut seulement veiller à ce que, dans 'B:tbx§', les chapitres soient correctement numérotés (le premier verset, ou paragraphe, devant avoir pour adresse 6,1; et non 1,1). Il suffit pour cela, conformément au mode de numérotage expliqué au §2.1, de mettre en tête du texte 'B:tbx', immédiatement après la ligne de titre, 6 chiffres '1' séparés par des blancs.

On peut alors soumettre séparément à 'trimu§' les fichiers 'B:tax§' et 'B:tbx'; puis le programme spécial de fusion, 'fus§', permet d'obtenir le fichier trié, 'B:tx§t', du texte intégral des 10 chapitres, en interclassant, en une seule lecture, les fichiers 'B:tax§t' et 'B:tbx§t'.

```
εν
1 1     εν αρχη ην ο
αρχη
1 1    εν αρχη ην ο λογοσ
ην
1 1   εν αρχη ην ο λογοσ και
ο
1 1 εν αρχη ην ο λογοσ και ο
λογοσ
1 1 αρχη ην ο λογοσ και ο λογοσ
και
1 1 ην ο λογοσ και ο λογοσ ην
```

Ci-dessus: début du listage 'B:J1x(§' créé en vue de la concordance
Ci-dessous: lignes de la concordance 'B:J1x(§t' afférentes à φωσ

```
φωσ
1 4 ζωη ην το φωσ των ανθρωπων και
φωσ
1 5 ανθρωπων και το φωσ εν τη σκοτια
φωσ
1 8 ην εκεινοσ το φωσ αλλ ινα μαρτυρηση
φωσ
1 9 φωτοσ ην το φωσ το αληθινον ο
φωτιζει
1 9 το αληθινον ο φωτιζει παντα ανθρωπον ερχομενον
φωτοσ
1 7 μαρτυρηση περι του φωτοσ ινα παντεσ πιστευσωσιν
φωτοσ
1 8 μαρτυρηση περι του φωτοσ ην το φωσ
```

3.3.2 Création d'une concordance

On appelle concordance une liste donnant successivement, pour chacune des formes lexicales attestées dans un texte, un bloc de lignes comprenant toutes les occurrences de cette forme, dans un contexte plus ou moins large.

Un programme spécial, 'prCcrd§', permet de créer une concordance d'un texte 'B:tx'. De façon précise, 'prCcrd§' utilise pour donnée le fichier 'B:tx§' des formes (avec adresse en 2-ème ligne); et il crée un fichier 'B:tx(§' qui ne diffère de 'B:tx§' qu'en ce que, sur la deuxième ligne afférente à chaque forme, celle-ci se trouve récrite, après l'adresse, dans un contexte formé des 3 mots précédents et des 3 mots suivants.

Compte tenu de ce que le programme 'trimu§' recopie la deuxième ligne sans en traiter le contenu, on voit que 'trimu§' crée, à partir de 'B:tx(§', un fichier trié, 'B:tx(§t', qui est une concordance pour le text de base 'B:tx'.

Comme exemple, nous avons traité le texte 'B:J1x' du chapitre 1 de l'Évangile selon Saint Jean. Le listage 'B:J1x(§' a été créé en moins d'une minute et demi; mais il a fallu plus de 22 minutes pour créer la concordance

'B:J1x(§t' par le programme de tri 'trimu§'. En revanche, 10 minutes ont suffi pour trier le fichier usuel 'B:J1x§' des formes avec leurs adresses (sans contexte). Il n'y a pas à s'étonner de cette différence: le fichier 'B:J1x(§' a un nombre d'octets qui dépasse 4 fois celui de 'B:J1x§'; or nous estimons que 'trimu§' passe près de la moitié du temps en lecture et écriture.

Sinalons qu'un programme 'psCcrd§' permet de créer rapidement le listage 'B:tx§t' à partir de la concordance 'B:tx(§t', en supprimant les contextes.

3.3.3 Substitution de formes

On peut désirer substituer à certaines formes dérivées leur forme de base: par exemple remplacer des formes de noms par le nominatif singulier, ou des formes de verbes par l'infinitif; ou remplacer un mot rare par un synonyme usuel, etc... Ici encore, on recourt à l'algorithme de fusion.

Supposons créé un *lexique de transformation*, dont le listage comprend un nombre pair de lignes; les lignes de rang impair donnant les formes xxx à remplacer, avec sur la ligne suivante (de rang pair) la forme yyy destinée à prendre la place de xxx. Si ce lexique est désigné par le sigle 'lx1', nous convenons que le listage du lexique aura pour nom complet 'B:transllx1§'. En soumettant 'B:transllx1§' au programme 'trimu§', on obtient un lexique 'B:transllx1§t' où les formes à remplacer se succèdent dans l'ordre alphabétique (leurs substituts, inscrits sur les lignes paires, étant, par contre, dans un ordre quelconque).

On imagine aisément comment le programme de transformation 'trans§', ayant en lecture, d'une part le *lexique de transformation* ordonné 'B:transllx1§t', et d'autre part le fichier de formes triées 'B:tx§t' afférent à un texte 'B:tx', peut, en seul passage, créer un listage 'B:txllx1l' qui ne diffère de 'B:tx§t' qu'en ce qu'y ont été effectuées toutes les substitutions spécifiées par le lexique.

Il faut prendre garde que dans 'B:txllx1l' l'ordre des formes n'est pas nécessairement alphabétique; qu'après substitution, on peut avoir deux blocs distincts d'occurrences d'une même forme (l'un de ces blocs, ou les deux, ayant été créé par substitution); que si ces deux blocs se suivent immédiatement ils peuvent former un bloc unique où n'est pas respecté l'ordre naturel des adresses (par chapitre et verset).

Le programme 'trimu§' suffit à assurer l'ordre alphabétique correct des formes, mais non l'ordre naturel des adresses pour chaque forme. On obtient un rangement parfait avec la variante 'triml' du programme de tri (cf. §2.2 *in fine*). À partir de 'B:txllx1l', 'triml' produit un listage 'B:txllx1§t', identique à celui qu'on aurait obtenu en soumettant à 'trimu§' le fichier 'B:txllx1§' correspondant à un texte 'B:txllx1' obtenu en effectuant sur le texte de base 'B:tx' les substitutions spécifiées par le lexique 'B:transllx1§'; l'avantage de notre

démarche étant qu'on évite d'utiliser directement le texte 'B:tx||x1', qui ne peut être créé par fusion.

4 Correspondance entre formes de mots et chapitres du texte grec de l'Évangile selon Saint Jean

On a soumis à l'analyse des correspondances le tableau (154 , 21) croisant l'ensemble des 154 formes (retenues selon les critères donnés au §3.1) avec les 21 chapitres de l'Évangile selon Saint Jean; puis, sur chacun des deux ensembles en correspondance, on a effectué une classification ascendante hiérarchique, en créant des listages Vacor d'aide à l'interprétation.

Dans l'interprétation, on ne peut séparer l'analyse factorielle des classifications; car, d'une part, l'ensemble des 154 mots (formes) ne pouvant être placé sur un graphique tenant dans une page, on figure seulement les centres de classes de mots; et, d'autre part, la multiplicité des dimensions ne permet de lire en projection sur des plans les proximités réelles de l'espace; mais l'exposé n'en sera pas moins partagé en trois §.

Une analyse du même texte, mais basée sur un autre choix du vocabulaire et avec réduction de toutes les formes à la forme de base (nominatif singulier ou infinitif), a déjà été publiée par Mgr. B. de Solages et l'Abbé J.-M. Vacherot (in *Pratique de l'Analyse des Données en Linguistique et Lexicologie*): nous mettrons à profit l'interprétation proposée par ces éminents chercheurs.

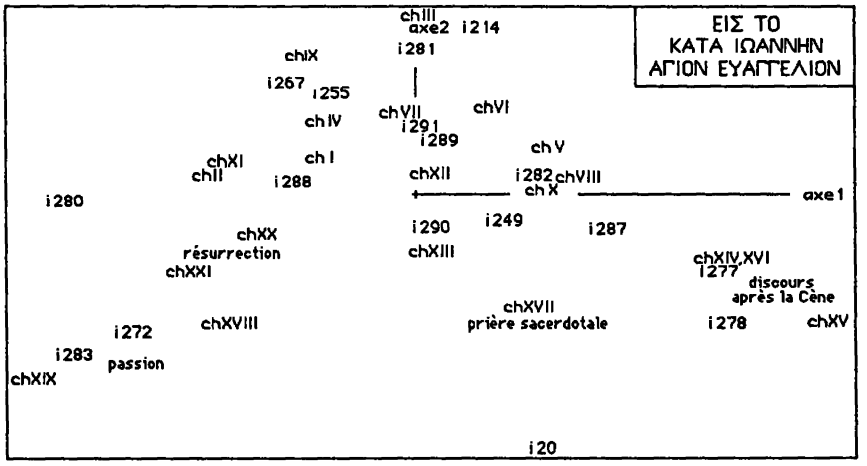
4.1 Analyse de correspondance

```
mots de dij x chapitres de B:Jx
trace  :   2.022e+0
rang   :    1     2     3     4     5     6     7     8     9    10
lambda : 2833  2096  1754  1578  1499  1150  1043   977   930   857 e-4
taux   : 1401  1036   867   780   741   569   516   483   460   424 e-4
cumul  : 1401  2437  3305  4085  4826  5395  5910  6393  6853  7277 e-4
```

Avec 21 colonnes (chapitres), on a 20 facteurs non triviaux; on voit que les taux afférents aux axes successifs décroîssent lentement; mais l'axe 1 est nettement séparé de l'axe 2, comme celui-ci l'est des suivants: ce qui nous engage à considérer le plan (1,2).

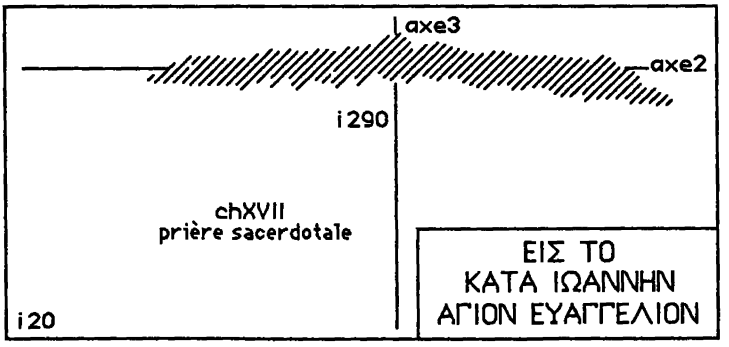
Sur l'axe 1, les trois chapitres {14, 15, 16} du discours du Seigneur après la Cène (F1>0) s'opposent aux deux chapitres {18, 19} du récit de la Passion, suivis de ceux du récit de la Résurrection {20, 21} (F1<0): On parlera donc, comme le font les auteurs cités, d'opposition entre discours et récits.

Plus précisément, la classification agrège, en une classe j35, le chapitre, 20 déjà cité, et les chapitres {7, 9, 11, 2, 4} que nous tenterons de caractériser comme des récits mettant en jeux de nombreux personnages: Noces de Cana (2), entretien avec la Samaritaine (4), Jésus au Temple de Jérusalem (7), guérison d'un aveugle-né (9), résurrection de Lazare (11).



Du côté (F1>0) (avec le discours après la Cène) on trouve la classe j30: {10, 5, 8}, trois chapitres, où dans son enseignement le Christ affronte une foule hostile.

Au sommet de l'axe 2, on trouve le chapitre 3, épisode unique dans l'Évangile: un dignitaire juif, un pharisien nommé Nicodème, vient, de nuit, en secret, se soumettre à l'enseignement de celui qu'il appelle ραββι, "Maître".

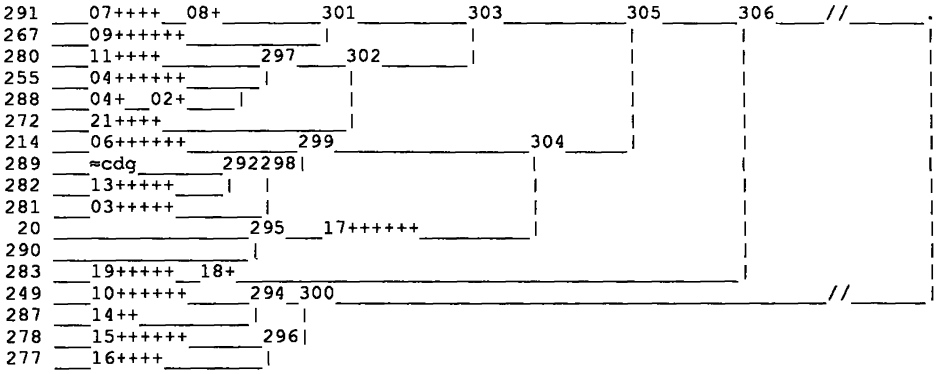


Le chapitre 17, prière sacerdotale (ainsi nommée parce que le Christ, comme souverain prêtre, y prie le Père pour l'universalité des fidèles) est également proche des discours: mais l'originalité de ce texte apparaît clairement dans le plan (2,3): le mot le plus caractéristique est i20, δεδωκας, "tu as donné": car le Christ désigne les fidèles comme "ceux que tu m'as donnés". Les auteurs cités, trouvent de même le chapitre 17 très à l'écart sur l'axe 2, et associé au verbe διδωμι, "donner".

```
c | Partition en 17 classes: formes de la classe numéro c
291| αιωνα ημων εχει οιδα λεγουσιν αληθησ ελθειν δυνασθε
   | πεμψασ εμαυτου λαλω απεκριθησαν ιουδαιοι οιδατε
   | φαρισαιοι ελεγον παρρησια ανθρωπον ποιει αποκτειναι
   | χριστοσ λεγων προφητησ
267| εποιησεν λεγοντεσ ημεισ ειπαν οιδαμεν οφθαλμουσ
280| μνημειον ημερασ μαριαμ
255| γυνη υδατοσ υδωρ
288| μαθηται πασχα επιστευσαν γαλιλαιασ ωρα ηλθεν ερχεται
   | απηλθεν πολλοι παλιν ειπεν ιησουσ κυριε λεγει
272| ειπων μαθητησ κυριοσ σιμων πετροσ
214| ουρανου αρτον(+σ)
289| ελεγεν δοξαν εχετε οχλοσ σημεια ημερα πεμψαντοσ
   | θελημα απεστειλεν δεδωκεν εργα ποιειν ανθρωποσ ρηματα
   | μενει ανθρωπου ηλθον λογοσ λεγεισ ζητειτε μαθητων
   | απεκριθησαν ειπον πνευμα
282| λαμβανει λεγω αμην ποδασ
281| ιωαννησ φωσ ραββι ερχομενοσ θεου αιωνιον ζωην δει
   | θεοσ υιον πιστευων δυναται
 20| δεδωκασ
290| σοι υιοσ εδωκεν ονομα συ ιωαννου σε εμε κοσμον ινα
   | σου αληθεια μοι κοσμοσ κοσμω κοσμου
283| πιλατοσ γραφη εξουσιαν ιδε τοπον αρχιερεισ ιουδαιων
   | ιουδαιοισ εξηλθεν ιησου ιησουν βασιλευσ
249| ψυχην προβατα
287| εστε λογον υμεισ εμου με εγω πατηρ ποιω πιστευετε
   | ερχομαι ημιν θεον χαγω πατροσ μου
278| καρπον πληρωθη εμη εμοι υμασ
277| λελαληκα μικρον υπαγω υμων υμιν ονοματι πατερα
```

4.2 Classification sur l'ensemble des formes (mots)

On a retenu une partition de l'ensemble des 154 formes en 17 classes, qu'on a étiquetées, d'après le listage iVacorj, selon leurs affinités avec les 21 chapitres.



Sans prétendre tenir le rôle d'un helléniste familier de la langue du Nouveau Testament, nous proposerons quelques réflexions susceptibles d'intéreser un statisticien qui a lu Saint Jean en quelque langue que ce soit.

Après i20, c'est i290 qui est le plus étroitement associée à la prière sacerdotale (ch. 17): i290 contient, notamment, les formes de la déclinaison du mot κοσμοσ, "monde". Le mot ιωαννου, génitif de "Jean" ne se trouve dans i290 que du fait de son association avec le pronom σε, "toi", dans d'autres chapitres.

Les formes du pronom υμεισ, "vous", sont dans des classes voisines {287, 278, 277}, toutes incluses dans i300. Mais il n'en est pas de même des fomes du nom de Jésus: le nominatif, ιησουσ (cas du sujet), est dans i288, nettement séparé des cas obliques (accusatif, génitif), ιησουν, ιησου, qui sont dans i283, avec πιλατοσ, Pilate, nom au nominatif du procurateur romain devant qui le Christ comparaît comme accusé, lors de sa passion (chapitres 18, 19). On voit qu'au moins pour un nom propre, il peut n'être pas indifférent de ne pas cumuler les formes de tous les cas.

Les formes {ψυχην, προβατα} constituent la classe i249: on reconnaît la locution "donner sa vie pour ses brebis", caractère propre du Bon Pasteur que présente la parabole contenue dans le chapitre 10. De même i214, {αρτον(σ), ουρανου}, {pain, du Ciel}, est associé au discours Eucharistique inclus dans le chapitre 6; (ici, par exception, αρτον est cumulé avec αρτοσ). On n'en doit pourtant pas conclure que ces associations étroites de mots correspondent ici à des doublets de formes consécutives fréquemment répétées (doublets sur le dénombrement desquels A. Salem a appelé l'attention des linguistes). Mais c'est le cas pour {σιμων, πετροσ}, {Simon, Pierre}, dans i272.

Remarquons encore i255, {γυνη, υδατοσ, υδωρ}, {femme, eau}, associé au ch. 4, rencontre avec la Samaritaine; et i280, {μνημειον, ημερασ, μαριαμ}, {tombeau, jours, Marie}, avec ch. 11, résurrection de Lazare.

```
==========================================================================
c | Partition en   7 classes : Sigles des chapitres de la classe numéro c
--------------------------------------------------------------------------
34| ch@1 ch@3 ch12 ch@6 214++ (289+) 281++++
--------------------------------------------------------------------------
29| ch13 ch21 lavement de pieds, Cène; résurrection 282++ 272+++++ (Pierre)
--------------------------------------------------------------------------
35| ch@7 ch@9 ch11 ch20 ch@2 ch@4  récits,personnages multiples, miracles
--------------------------------------------------- 280++ ---- 288++ --
27| ch18 ch19        la Passion 283++++++
--------------------------------------------------------------------------
17| ch17             prière sacerdotale 20++++ 290++++
--------------------------------------------------------------------------
30| ch10 ch@5 ch@8   Jésus aux prises avec le peuple, discours 287+++
--------------------------------------------------------------------------
31| ch14 ch16 ch15   discours après la Cène 287+(Max)  278++ 277++++
==========================================================================
```

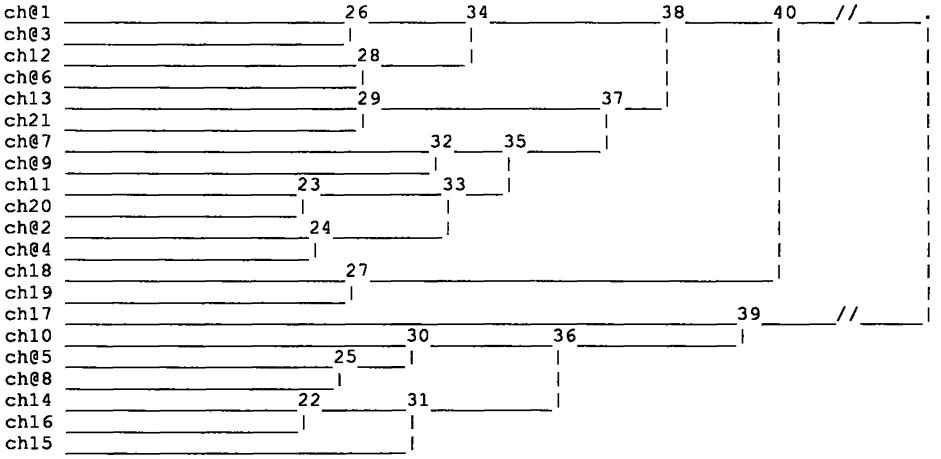
4.3 Classification de l'ensemble des 21 chapitres

De la hiérarchie, on a extrait une partition en 7 classes, où figurent les divisions déjà signalées. L'étiquetage a été fait en terme de classes de formes (mots), d'après le listage jVacoriq: on ne pouvait, en effet, tenter une interprétation prenant en compte, individuellement, les 154 formes. On publie ègalement le tableau de contingence (saisi lors de la création du listage Vacor) croisant les partitions entre les deux ensembles I (mots) et J (chapitres).

La classe 39 comprend ce qu'on a appelé les "discours": la prière sacerdotale y est agrégée au discours après la Cène, discours dont elle est la conclusion. Les discours du Christ aux prises avec le peuple sont caractérisés par une fréquence élevée des mots de la classe i287 {moi, et moi, Père (nom. et gén.), croyez, vous (nom.), ...}. La fréquence cumulée de ces mots est encore un peu plus élevée dans la classe j31 (discours après la Cène): mais celle-ci est plus particulièrement caractérisée par les classes i278 et i277: {Père (accus.), vous (cas fléchis), ελαληκα (j'ai dit), μικρον (un peu, c'est-à-dire: un peu de temps), υπαγω (je m'en vais)}.

La classe j27 du récit de la passion est à part, caractérisée par i283: on a noté que, dans cette classe, Pilate est au nominatif (ainsi que les grands prêtres); tandis que Jésus est à l'accusatif ou au génitif. De même, "juifs" est au génitif ou au datif: cf. roi des juifs, Pilate dit aux juifs...

À la classe j27, s'agrège, au sein de la classe j40, la classe 38 qui comprend les classes {j35, j29, j34}. Dans j35, on a des chapitres ayant généralement le caractère de récits; avec plusieurs épisodes, comme pour la rencontre avec la Samaritaine (ch. 4), la guérison de l'aveugle-né (ch. 9) ou la résurrection de Lazare (ch.11); épisodes éventuellement séparés: miracle de l'eau changée en vin à Cana avec ensuite les marchands chassés du Temple (ch. 2), ou des rencontres avec Jésus après sa résurrection (ch. 20).



mots de dij x chapitres de B:Jea:Jx
cariq = 17 ; cariq = 7 ; trace = 2.022

7	j34	j29	j35	j27	j17	j30	j31
i291	41	9	124	14	1	87	13
i267	18	3	64	8	3	10	5
i280	1	0	25	2	0	0	0
i255	1	1	21	1	0	7	1
i288	136	89	273	66	4	82	36
i272	4	32	14	18	0	1	1
i214	29	3	0	0	0	0	0
i289	137	13	90	33	4	73	36
i282	30	30	6	0	0	27	14
i281	119	4	41	2	3	38	5
i20	0	0	0	1	9	0	0
i290	122	45	103	45	84	66	59
i283	28	9	36	90	3	12	3
i249	2	4	2	0	0	12	1
i287	94	43	82	27	32	200	157
i278	12	3	9	10	3	8	47
i277	27	18	24	9	2	49	111

La classe j29 contient, avec le récit de la Cène, précédé de celui du lavement des pieds des apôtres par le Seigneur (ch. 13) un dernier chapitre (21-ème) d'épisodes après la résurrection: le personnage de Pierre étant présent ici et là.

Enfin, j35 comprend l'entrée à Jérusalem (ch. 12,rameaux), le discours Eucharistique (ch. 6), l'entrevue avec Nicodème (ch. 3) et le chapitre 1, composite lui aussi, avec la vocation des apôtres, précédée de ce sublime prologue qui a valu à l'évangéliste Saint Jean le titre de Théologien (Θεολόγος).

Satisfaisante dans son ensemble, la classification ne laisse pas d'appeler notre attention sur le fait que la division en chapitres n'est sans doute que le produit de l'activité contingente de copistes et d'éditeurs; ce qui suggère au statisticien d'adopter plutôt une division du texte en épisodes. Celle-ci pourrait d'ailleurs être introduite sans modifier directement le texte déjà saisi, à l'aide d'un programme qui, en une seule lecture, substituerait, aux adresses par chapitre et verset, des adresses par chapitre et épisode; ou, plus simplement encore, en créant un fichier auxiliaire définissant la subdivision du texte en épisodes et en ajoutant au programme 'tridic§' une troisième option de "cumul par épisodes", qui se baserait sur ce fichier auxiliaire pour ventiler les occurrences des formes.

5 Conclusion

Sans prétendre proposer une bibliothèque de programmes achevée, nous croyons légitime d'appeler l'attention des statiticiens férus de linguistique sur l'élégance des traitements que permet l'application systématique de l'algorithme de fusion. Dès maintenent, les temps de calculs nous semblent acceptables; et ils pourraient être encore réduits si, disposant d'une mémoire centrale ample et facile à gérer, on effectuait les fusions sans recourir incessamment à des entrées et sorties.

Bibliographie

L'article [CLASS. CORPUS], de M. Reinert, publié dans ce même cahier, donne, sur le thème du présent travail, une bibliographie de base.