Initiation pratique à la constitution et à l'exploitation de corpus électroniques en langue arabe (IV^{e} partie)

Djamel Eddine Kouloughli[1]

0. Préambule

Dans le précédent article de cette série, nous avons présenté, en nous fondant essentiellement sur l'ensemble logiciel TACT[2], quelques-unes des fonctionnalités les plus importantes d'un système d'analyse de corpus. Nous avons traité notamment de la génération d'une base de données textuelles à partir d'un texte (au format TXT), et de certaines des possibilités que cela offrait pour l'exploration de ce texte, en particulier la production d'une « carte d'identité » détaillée du texte : nombre et fréquence des occurrences, nombre et fréquence des formes, contexte d'apparition des occurrences et leur distribution dans le texte. Nous avons terminé en montrant comment, en utilisant le langage des « expressions régulières », on pouvait identifier dans un texte arabe tous les mots appartenant à une même racine.

Dans le présent article, nous allons poursuivre notre survol des possibilités qu'offrent les logiciels d'analyse de corpus pour l'exploration des textes. Nous porterons ici notre attention non plus sur le repérage de mots isolés mais sur celui des groupes de mots. Bien entendu, nous nous intéresserons non aux groupements uniques et, en quelque sorte, accidentels de mots[3], mais à ceux qui présentent une certaine régularité et par suite une certaine fréquence, et que l'on désigne souvent, dans la littérature spécialisée, sous le nom de « collocations ».

1. CNRS, HTL (Histoire des théories linguistiques), UMR 7597.

2. TACT (Text Analysis Computing Tools) : ensemble de programmes d'analyse de corpus diffusé par l'université de Toronto, entre autres sur http://www.chass.utoronto.ca/tact/ (site consulté le 3 avril 2009). Voir Lancashire *et al.* (1996).

3. Les groupes singuliers de mots peuvent certes être porteurs d'une importante charge sémantique (en raison même de leur singularité) et être très révélateurs des trouvailles stylistiques d'un auteur. Mais en raison même de leur unicité, ils échappent aux procédures de recherche essentiellement fondées sur la fréquence. Par contre, un groupement qui ne se répète même qu'une fois (c'est-à-dire qui apparaît seulement deux fois dans un texte) sera repéré par de telles procédures, quelle que soit la longueur du texte.

1. À PROPOS DES COLLOCATIONS[4]

Bien que l'ensemble des collocations d'un texte ne constitue pas toujours une partition nette, mais plutôt des entités aux contours flous, avec divers chevauchements, on peut en principe y distinguer cinq grands types :

• Les collocations grammaticales, qui sont des groupements à statut syntagmatique ou des séquences de mots fonctionnels se succédant à ordre fixe dans les textes, comme par exemple *qabla 'an* ou *baHaca &an*. Les collocations grammaticales présentent des degrés divers de variabilité morphologique, comme le suggèrent les deux exemples ci-dessus[5]. Compte tenu de leur statut, ces groupements sont très fréquents dans tout texte, mais ils sont aussi, pour les mêmes raisons, « peu intéressants » en termes d'analyse textuelle dans la mesure où ils ont les mêmes chances d'apparaître quel que soit le contenu du texte. Par contre, ils sont intéressants pour le grammairien et le lexicographe car ils manifestent des régularités importantes dans la structure de la langue et/ou du lexique.

• Les composés, qui sont en fait des unités lexicales complexes, formées de deux mots ou plus se succédant dans un ordre fixe et fonctionnant comme une seule unité. Par exemple *Tabiyb 'asnaan* ou *elwilaayaat elmut*aHida#*. Elles présentent généralement un certain degré de variabilité morphologique.

• Les collocations lexicales qui sont, elles aussi, des entités lexicales complexes, mais présentent une grande latitude de variation dans l'ordre linéaire et dans la morphologie. Les collocations lexicales présentent en outre la caractéristique que chacune des unités qui les composent conserve une valeur sémantique de base, ce qui permet, dans la majorité des cas, de déduire la signification globale de la collocation de la connaissance du contenu lexical de ses composants. Ainsi par exemple, le verbe arabe *'alqaY*, qui signifie primitivement « jeter », cooccurre, dans l'une de ses acceptions, avec des objets comme *xiTaab*, *muHaaDara#*, *qaSiyda#*, termes très variables dans leur contenu sémantique mais qui tous impliquent le sème [communication orale]. Les traductions du groupe résultant varient selon l'objet (faire un discours, donner une conférence, déclamer un poème), mais conservent le sème commun. Un bon dictionnaire devrait indiquer cette acception du verbe et l'illustrer par des exemples représentatifs.

---

4. La littérature relative aux collocations est relativement abondante, mais peu de travaux donnent une bonne synthèse sur les divers faits de cooccurrence de formes dans un texte, et la terminologie des auteurs présente un grand degré de variabilité qui complique encore les choses. On se contentera ici de renvoyer à Benson (1985) et Liang (1991) pour les collocations en général et à Foley (1985) pour les formules.

5. En effet, le premier exemple est pratiquement invariable (hormis la « variante hypocoristique » *qubayla 'an*), alors que le second peut non seulement être conjugué, mais aussi nominalisé en *elbaHc &an*. Cependant le degré de variation des collocations grammaticales reste faible et peut rapidement être résumé dans une bonne grammaire ou un bon dictionnaire.

• Les expressions figées, qui se différencient des collocations lexicales non seulement par leur relative absence de variabilité mais aussi par le fait que leur valeur sémantique n'est généralement pas déductible de celle de leurs composants (on parle à ce propos de non-compositionnalité). Par exemple le sens de l'expression *xabaTa xabTa &a$waa'*, à savoir « agir au hasard », n'est pas déductible du sens compositionnel de ses constituants et offre comme seule latitude de variation la conjugaison du verbe initial.

• Les formules, que l'on ne trouve que dans certains types de textes[6], se caractérisent par une longueur variable et ne présentent qu'une latitude de variation limitée et largement contrainte. Par exemple, dans le texte coranique, la séquence de longueur 4 « *yaa 'ay*uhaa el*a@iyna 'aamanuwe* » se répète 89 fois, et la séquence de longueur 6 « *maa fiy elsamaawaati wamaa fiy el'arDi* » 28 fois. On voit immédiatement que le sens de ces séquences est parfaitement compositionnel et qu'aucune contrainte ne semble s'y exercer[7]. Il s'agit là de manifestations du « style formulaire » qui caractérise de nombreux textes anciens à statut initialement oral.

2. Recherche de collocations dans les corpus

Ces précisions données, nous allons à présent nous intéresser aux moyens que donnent les logiciels de traitement de corpus, et plus spécifiquement TACT, pour mettre en évidence dans les corpus textuels ces différents types de collocations.

L'ensemble logiciel TACT permet à l'utilisateur deux types d'approches des phénomènes de collocation : l'une, que l'on peut qualifier de « déterministe », permet de rechercher les collocations qui concernent une unité lexicale déterminée ou une liste prédéfinie d'unités. L'autre, plutôt heuristique, permet de rechercher toutes les collocations présentes dans un texte et donc éventuellement d'en découvrir certaines dont on ne soupçonnait pas l'existence.

2.1. Recherche déterministe de collocations[8]

Dans le précédent article, nous avons vu que le logiciel Usebase permettait, notamment grâce à la commande de recherche Query, d'interroger la base de données textuelles (BDT) constituée par le logiciel Makebase, à partir du texte à explorer. C'est également à partir de ce logiciel et de cette commande que l'on peut conduire une recherche de collocations de mots définis.

---

6. Sur les formules en général, voir Foley (1985). Sur les formules en poésie arabe, voir Monroe (1972).

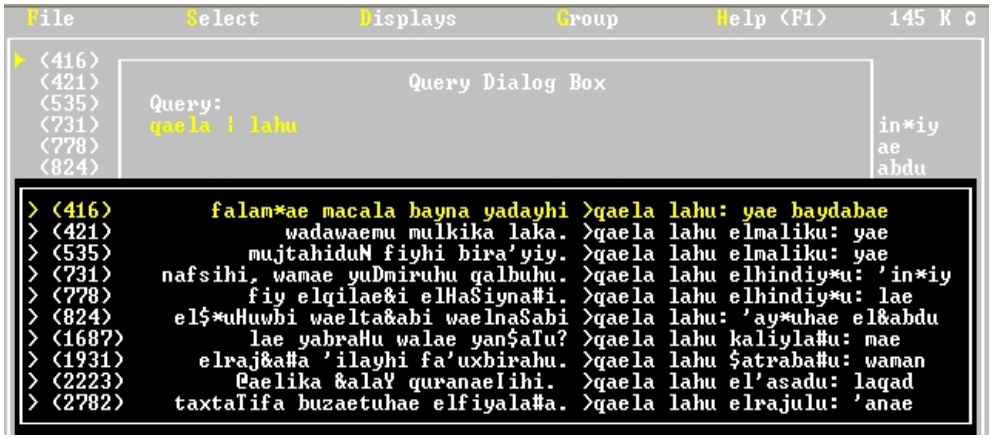
7. Dans certains corpus poétiques, comme *L'Illiade et l'Odyssée* d'Homère ou la poésie arabe ancienne, des contraintes métriques précises pèsent sur les formules (voir les références données dans la note précédente).

8. Tous les exemples d'interrogation du présent article porteront, sauf indication contraire, sur le texte de *kaliyla# wadimna#*, d'*ebn elmuqaf*a&*, texte suffisamment connu de tout arabisant pour qu'il soit inutile de le présenter en détail...

2.1.1. Recherche de syntagmes

On entend ici[9] par syntagme une séquence dans laquelle une unité lexicale définie est suivie par une ou plusieurs autres unités. Il s'agit donc ici de repérer les collocations dont les unités constitutives sont strictement contiguës.

L'opérateur de recherche de telles collocations est, dans TACT, le symbole[10] « | ». Ainsi, en appelant la commande Query dans Usebase, et en tapant par exemple : *qaela* | *lahu*, on obtient le résultat suivant[11] :



On peut ensuite, avec la commande du menu déroulant Displays, demander par exemple l'affichage d'un KWIC[12] donnant la liste en contexte de toutes les occurrences du syntagme :

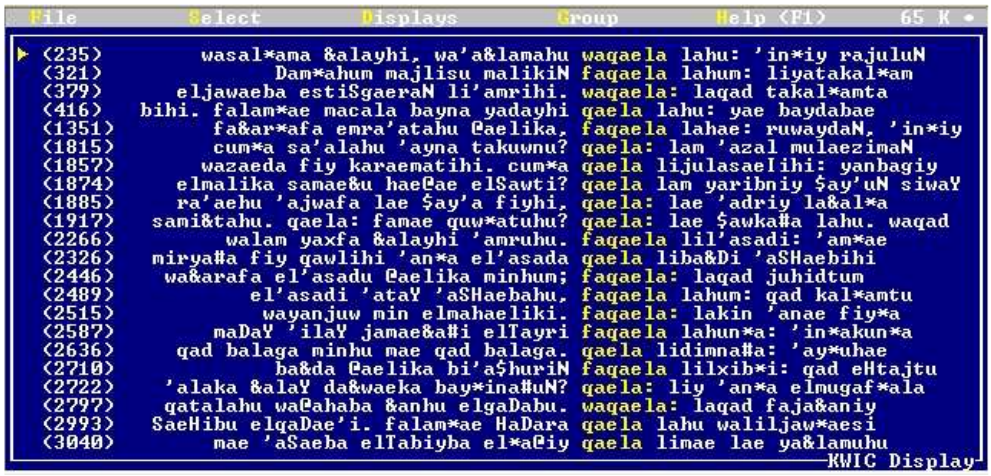
9. Cette conception du syntagme est purement conventionnelle et ne prétend pas se substituer à celle des manuels de linguistique. Elle permet simplement de désigner clairement ce que l'interrogation dont il est question ici restitue. C'est à l'usager d'adapter ensuite l'outil informatique pour obtenir des résultats significatifs.

10. Obtenu sur un clavier français par la séquence « Alt Gr-6 ».

11. Dans ce résultat et les suivants, seuls les débuts de listes seront affichés.

12. Rappelons que cet acronyme signifie « *Keyword in context* ».

On voit d'emblée que ce type d'interrogation ne peut pas, le plus souvent, nous apprendre grand-chose puisque l'on a prédéfini, en quelque sorte, la nature des données à rechercher. Mais il faut savoir que l'on peut, ici encore, faire usage des possibilités du langage des « expressions régulières » (désormais LER) pour élargir la portée de telles interrogations et obtenir de cette façon des résultats plus ouverts. Ainsi, si l'on entre la requête suivante : *.*qaela* | *l.**, qui demande de rechercher toute séquence où *qaela* est précédé par quelque chose (pour permettre l'apparition éventuelle des conjonctions clitiques *wa* et *fa*) et suivi par un mot quelconque commençant par la consonne *l* (pour permettre à tout ce qui peut suivre la préposition clitique *li-*/*la-* d'apparaître), on obtient un inventaire beaucoup plus large de la variation du syntagme recherché. Qu'on en juge plutôt (à partir d'un KWIC dont on a éliminé certaines lignes pour faire apparaître une plus grande variété de résultats) :



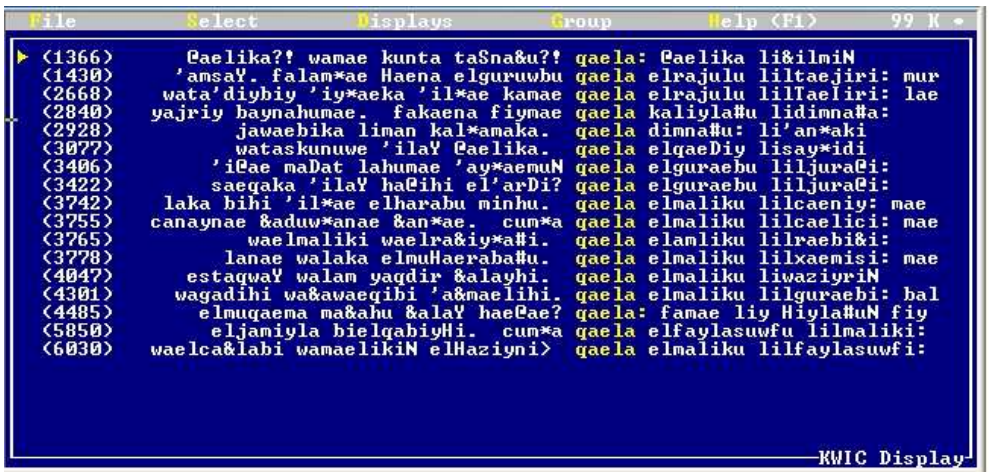
On voit que non seulement on récupère, par l'utilisation du LER, les conjonctions clitiques qui peuvent précéder le verbe et diverses formes de la séquence préposition *li*/*la* + pronom clitique, mais aussi des occurrences du verbe suivi par autre chose que cette séquence.

Les requêtes précédentes portaient sur des successions de deux mots, mais rien n'interdit d'utiliser plusieurs fois l'opérateur d'adjacence « | ». On peut, par exemple, soumettre une requête comme[13] : *za&amuwe* | \'*an*a.** | *.**, qui demande de rechercher après le mot *za&amuwe* toutes les occurrences de *'an*a* (avec ou

13. Noter que, dans cette requête, la lettre qui transcrit la *hamza* arabe, à savoir « ' », est précédée d'un *backslash* « \ ». C'est là une exigence du LER chaque fois que l'on utilise un symbole graphique utilisé par ailleurs comme opérateur par le langage en question. Ce *backslash* indique au programme qu'il ne doit pas considérer le symbole suivant comme un opérateur mais le traiter comme un symbole graphique quelconque. D'autres symboles du système de transcription TRS qui exigent d'être précédés de « \ » pour les mêmes raisons sont « & », « $ » et « @ ». Nous verrons pourquoi plus loin.

sans clitique), puis tout ce qui suit ce complétiviseur. On obtient le résultat présenté ci-dessous :

On peut aussi faire se succéder deux fois dans une requête l'opérateur d'adjacence. Par exemple la requête : *qaela* **||** *li.** demande de rechercher toutes les occurrences de *qaela* suivi de n'importe quel mot suivi lui-même d'un mot commençant par *li*. Le résultat, affiché en KWIC, est le suivant :

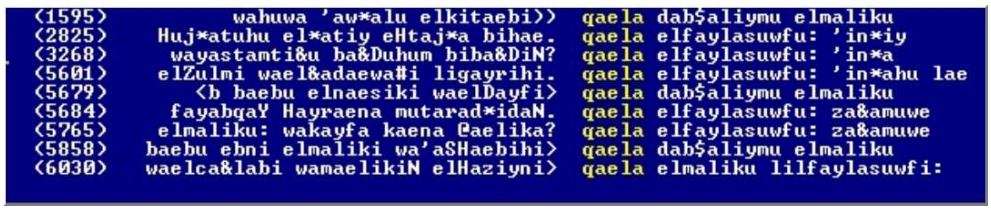


On peut voir ici que, si l'on recherchait « les destinataires du *qawl* » introduits par la préposition *li*, on les a bien tous, mais la requête a « ratissé » un peu trop large et certaines lignes sont superflues. On peut, dans la page de réponse de Usebase, supprimer ces lignes à la main avant d'afficher un KWIC qui ne contiendrait que les résultats pertinents.

2.1.2. Recherche de collocations

Nous avons vu (en 1. ci-dessus) que ce qui différenciait les collocations vraies des mots composés ou des expressions figées c'est, entre autres, la relative liberté des termes cooccurrents, lesquels peuvent apparaître dans des ordres variables l'un par rapport à l'autre. L'opérateur d'adjacence vu ci-dessus (en 2.1.1.), et qui ne peut repérer que des termes qui se suivent, n'est donc pas adapté à la recherche des collocations. Mais le langage d'interrogation de Usebase possède un autre opérateur spécialement conçu pour cette tâche. Il s'agit de l'opérateur de collocation dont le symbole est « & ». Pour bien montrer la différence entre ces deux opérateurs, commençons par soumettre, avec l'opérateur d'adjacence, une requête comme : *qaela* **|** *.*faylasuwf.**. Cette requête va rechercher dans la BDT toutes les occurrences de *qaela* suivies d'occurrences de *faylasuwf*, ce qui va donner, sans grande surprise :

Soumettons à présent une requête qui ne diffère de la précédente que par la substitution de l'opérateur de cooccurrence à celui d'adjacence, soit : *qaela* & *.*faylasuwf.**. On obtient alors une liste dont voici un extrait (toujours au format KWIC) :



À l'examen de cette liste, on pourrait croire qu'il s'agit d'une erreur puisque dans la première ligne, par exemple, on ne voit même pas apparaître le terme *faylasuwf*. Il s'agit là d'une simple limitation de l'affichage KWIC (que l'on peut d'ailleurs modifier) : le mot *faylasuwf* apparaît en réalité « plus loin » dans la ligne, comme le montre, pour cette ligne, l'affichage d'un contexte plus complet :

Cet affichage plus complet montre clairement comment s'est effectuée la recherche dans la requête comportant l'opérateur « & » : le programme a d'abord cherché toutes les occurrences de *qaela*, puis il a recherché en amont et en aval de ce mot s'il trouvait le mot *faylasuwf* dans une limite fixée par défaut[14] à cinq mots avant et après *qaela*. Dans l'exemple ci-dessus, le deuxième composant de la cooccurrence est trouvé quatre mots à droite du premier.

Une autre occurrence identifiée par le programme dans le cadre de la requête utilisant l'opérateur « & » est très instructive : la voici affichée dans un contexte élargi :

On voit qu'ici le mot *faylasuwf* occurre avant *qaela*, ce qui montre bien que « & » recherche en amont et en aval du premier cooccurrent. Mais ici, le programme est allé chercher le second cooccurrent en sautant la balise qui introduit le chapitre où apparaît *qaela* parce que Query, par défaut, ignore les balises. On peut, bien sûr, spécifier que l'on souhaite que cette frontière (ou toute autre spécifiée au programme sous forme de balise) bloque la recherche des collocations.

2.1.3. Recherche de « non-collocations »

On peut vouloir rechercher une unité dans un contexte où l'on veut explicitement en exclure une autre. Query dispose pour ce faire de l'opérateur[15] « ~ » qui fonctionne comme l'exact opposé de « & ». Ainsi la requête : *qaela* ~ *.*faylasuwf.** va rechercher dans la BDT toutes les occurrences de *qaela* qui ne sont pas précédées ou suivies d'occurrences de *faylasuwf*, par défaut dans une limite de cinq mots (ici encore modifiable).

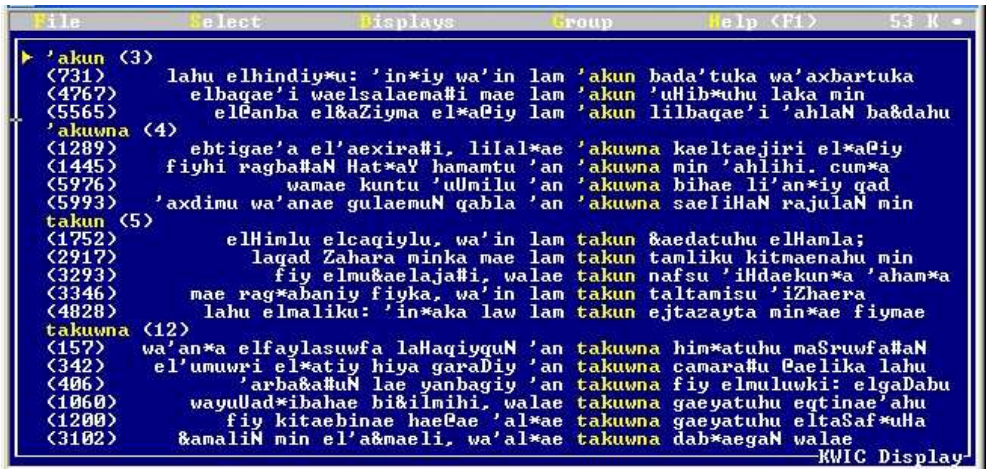
14. Cette limite peut être modifiée par l'utilisateur dans le champ *Span Context* de la fenêtre Query.
15. Obtenu sur un clavier français par la séquence « Alt Gr-2-Espace ».

2.1.4. Recherche d'associations

Supposons que nous souhaitions rechercher dans un texte arabe toutes les occurrences de toutes les formes du verbe *kaana/yakuwn*. Il est aisé de prévoir qu'une recherche portant sur les formes de la racine √*kwn* ne fera pas l'affaire : en effet, outre qu'elle passera à côté de formes comme *kaana*, *kaanat*, ou *kaanuwe*, elle extraira une flopée de formes dérivées (noms et verbes) qui ne correspondent pas à ce qui est recherché. La solution la plus commode est de recenser soi-même, judicieusement, la liste des formes de base de ce verbe et de soumettre cette liste à la recherche.

Les formes de base en question pourraient être *kaan.*, ..kuwn.*, ..kun.** et *kun.** : la première pour les formes à voyelle longue de la conjugaison à suffixe, la seconde pour les formes non apocopées de la conjugaison à préfixe, la troisième pour les formes apocopées, et la quatrième pour les formes à voyelle brève de la conjugaison à suffixe et de l'injonctif.

Query nous permet de concaténer toutes ces formes en une seule demande grâce à l'opérateur d'association noté « , ». La requête relative à toutes les formes du verbe *kaana/yakuwn* est donc : *kaan.*, ..kuwn.*, ..kun.*, kun.**. Voici les premières lignes du résultat de cette requête appliquée à notre BDT, affichées en KWIC :



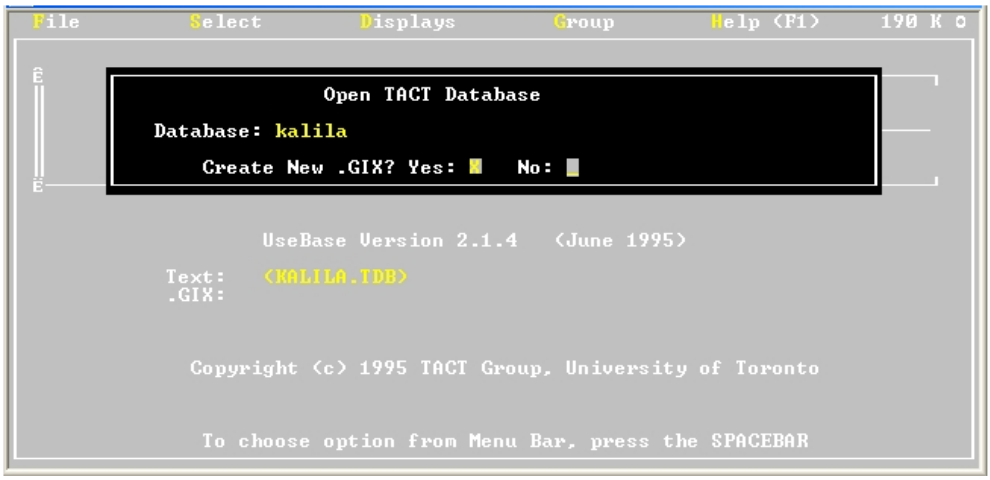
L'examen de l'ensemble de la liste montre que l'on a bien toutes les formes recherchées[16], même si l'on doit à l'honnêteté de dire que cette requête génère aussi un peu de « bruit » puisque l'on trouve des formes indésirables comme

16. Noter que cette recherche laisse de côté les conjonctions comme *wa-* et *fa-* et les particules comme *li-* qui pourraient venir se cliticiser aux formes verbales (voire les pronoms clitiques car des formes comme *kuntuhu* existent !). C'est par souci de simplification que nous procédons ainsi : il suffirait de faire précéder et suivre les formes de la requête du joker « .* » pour élargir les résultats.

kunuwzanae, qui répondent formellement à la requête sans être des formes du paradigme verbal recherché et qu'il faut « désélectionner » à la main[17].

Une fois le groupe constitué, on peut le sauvegarder pour des utilisations ultérieures. Pour ce faire, il faut d'abord constituer, à partir de la page d'accueil de Usebase, un « fichier GIX » (Group Index file) qui servira pour le stockage de toutes les listes que l'on est amené à constituer au cours de l'interrogation de la BDT et que l'on souhaite conserver.

Pour créer ce fichier, il suffit de répondre positivement à la question dans la page d'accueil de Usebase concernant la création d'un tel fichier. Ce fichier aura le nom de la BDT suivi de l'extension « .GIX » :

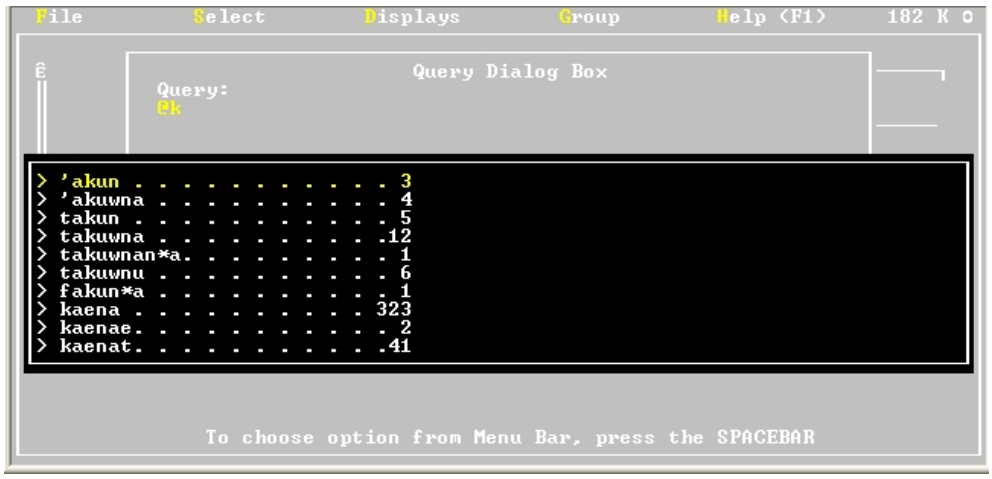


Une fois ce fichier créé, il suffit, lorsqu'on désire conserver une liste quelconque, de l'y stocker[18]. Cette opération se réalise aussi dans Usebase, à partir du menu déroulant *Group* et du choix *make* (touche de raccourci Ctrl-F3).

Revenons à notre liste des formes du verbe *kaena* : nous avons demandé sa sauvegarde sous le nom de « k ». Si nous voulons la réutiliser, il suffit de la demander à Query en utilisant la formule « @ k », formule où le « @ » indique au programme que ce qui suit est, précisément, un nom de liste stocké dans le fichier GIX. Query nous restitue alors la liste demandée :

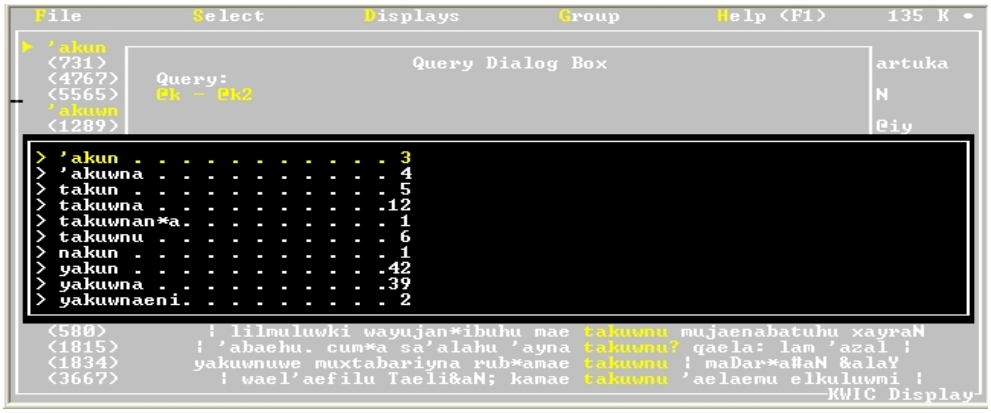
17. Alternativement, on pourrait formuler la requête d'une manière plus restrictive pour qu'elle ne génère plus les formes indésirables. Ici, c'est la composante « *kun.** » qui devrait être révisée.

18. Il faut, pour pouvoir utiliser le fichier GIX, demander son ouverture en même temps que celle du fichier TDB de même nom. Cela se fait au démarrage de Usebase, dans le menu « File ».



Supposons à présent que nous souhaitions avoir, outre la possibilité de repérer toutes les occurrences du verbe *kaena*, une possibilité équivalente concernant soit les formes à préfixe, soit les formes à suffixe. On pourrait, bien sûr, procéder comme ci-dessus en fabriquant deux nouvelles listes, une pour chaque paradigme de conjugaison. Mais une possibilité plus économique existe grâce à la présence, dans le langage d'interrogation de Query, de l'opérateur de « soustraction de liste », dont le symbole est le trait d'union « - ».

En effet, compte tenu du fait que nous possédons déjà la « liste k », une procédure possible consistera d'abord à en supprimer (à l'aide de la touche « Inser » du clavier) le symbole « > » devant les formes d'un des paradigmes de cette liste, par exemple les formes à préfixe. Nous pouvons maintenant sauvegarder la nouvelle liste, par la même procédure que pour la première, en la nommant par exemple « k2 ». À l'issue de ces manipulations, nous possédons les deux listes : « k » qui contient toutes les formes du verbe *kaena*, et « k2 » qui contient les seules formes de la conjugaison à suffixe. Nous pouvons à présent, grâce à ces deux listes, sélectionner soit l'ensemble des formes du paradigme verbal (liste k), soit celui des formes suffixées (liste k2), soit enfin celle des seules formes non suffixées. Pour demander à Query de nous sortir du texte toutes les occurrences de la conjugaison à préfixe du verbe *kaena*, il suffit d'utiliser l'opérateur de soustraction de liste. La requête sera donc : @ k - @ k2, qui demande de retenir tout élément de la liste k qui ne soit pas aussi dans la liste k2. Le résultat sera :



2.1.5. Recherche paramétrée

Les divers types d'opérateurs de recherche que nous avons vus peuvent être spécifiés par cinq paramètres qui permettent d'en affiner les résultats. Ces paramètres sont : *when*, *freq*, *simil*, *pos* et *span*. La syntaxe commune à l'utilisation de ces paramètres est la suivante : requête ; paramètre <valeur>. Les exemples qui suivent vont illustrer l'usage de cette syntaxe ainsi que la fonction de chacun des paramètres.

Le paramètre *when* permet de limiter la recherche à un segment défini du texte, segment identifié par une des balises déclarées dans le fichier à extension « .MKS » que l'on doit obligatoirement spécifier au moment de la constitution d'une base de données textuelle (fichier « .DBT ») sous Makebase[19]. Par exemple, sachant que l'un des types de balises déclarés dans le fichier « KALILA.MKS » permet de repérer les divers chapitres de l'ouvrage[20], la requête *@k ; when b = baebu elbuwmi waelgirbaeni* limitera la recherche des formes du verbes *kaena* au chapitre de l'ouvrage précédé par la balise : *<b baebu elbuwmi waelgirbaeni>*.

Le paramètre *span* est particulièrement utile lorsqu'on travaille sur un texte dont certaines balises identifient les prises de parole de locuteurs différents, comme dans une pièce de théâtre. On peut alors aisément analyser la langue de chacun des personnages en assortissant chaque requête du paramètre *span* idoine.

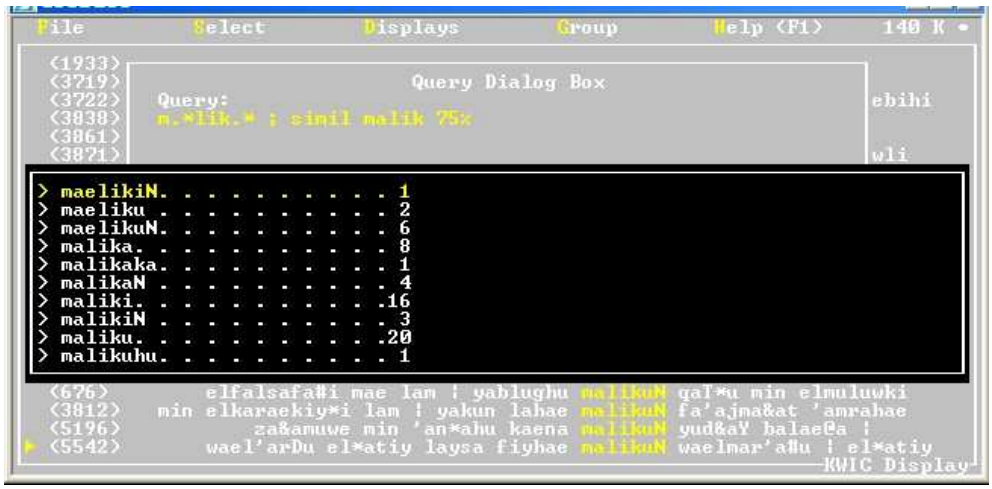
On peut en outre additionner les paramètres span dans une même requête grâce à l'opérateur « & » qui équivaut à la conjonction « et ». On pourra donc avoir une requête comme *love ; when (personnage Juliette & acte = 1, 2, 3)*, que le lecteur ne devrait pas avoir de mal à interpréter[21]...

Le paramètre *freq* permet de limiter la recherche aux unités présentant un fréquence absolue définie. Par exemple la requête *@k ; freq > 5* ne retiendra les

19. Voir sur ce point l'article précédent de cette série, dans LLMA n° 7, p. 81 : http://icar.univ-lyon2.fr/llma/sommaires/LLMA_7_05_Kouloughli.pdf
20. Balise référencée par « b » (comme *baeb*) dans le fichier MKS.
21. Noter la parenthèse qui réunit les deux termes de la conjonction sur laquelle porte *when*.

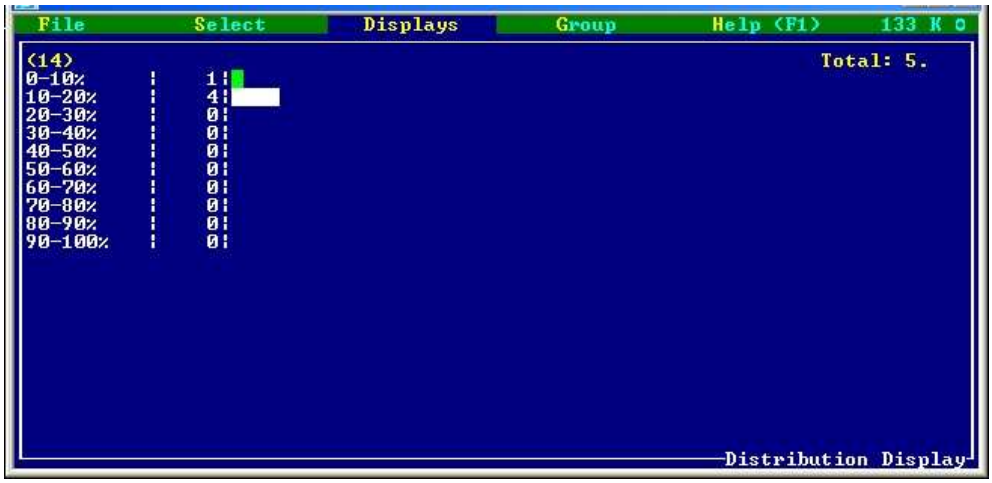
formes du verbe *kaena* que si elles ont une fréquence supérieure à cinq dans le texte.

Le paramètre *simil* met en jeu un algorithme de recherche capable d'identifier des mots présentant un certain degré de ressemblance avec le mot recherché. Ce degré est spécifié par un pourcentage donné en argument au paramètre ***simil.*** Ainsi, avec la requête *m.*lik ; simil malik 75 %* on obtient le résultat suivant :



On peut voir, même sur cette liste partielle, que les résultats obtenus sont tous de la racine √*mlk*, mais qu'ils sont assez différents les uns des autres. Si l'on abaisse un peu le pourcentage de ressemblance (à 70 % par exemple), la liste s'allonge et on y voit apparaître des formes qui ne relèvent plus de la racine √*mlk*, comme par exemple *manzilika*.

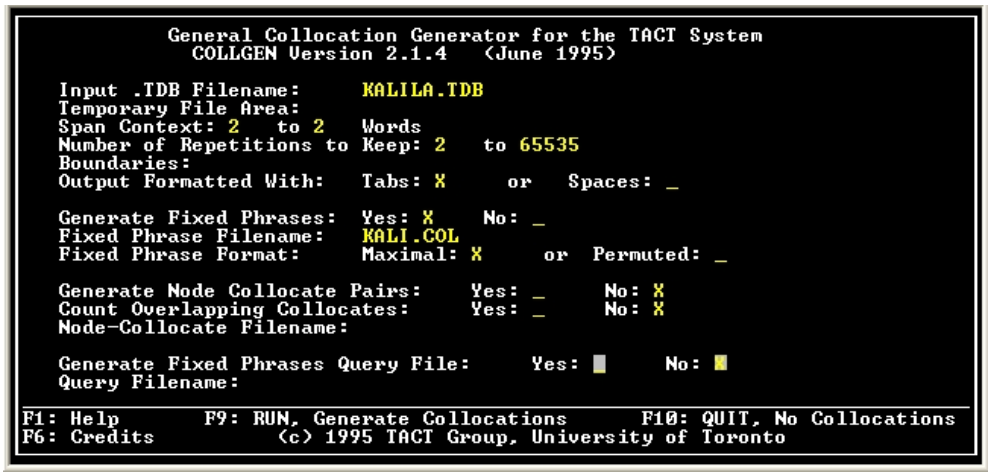
Le paramètre *pos* permet de rechercher toutes les positions où apparaît l'unité qui fait l'objet de la requête. Ainsi la requête *kisraY ; pos* génère la liste de toutes les occurrences du nom du grand empereur perse. Si l'on demande alors à faire afficher ces résultats sous la forme d'un histogramme de distribution, on constate qu'il n'en est fait mention que dans le début de l'ouvrage :



2.2. Recherche heuristique de collocations

C'est le logiciel Collgen qui permet de rechercher systématiquement les collocations d'un texte sans qu'il soit besoin de lui désigner un terme particulier. Pour comprendre le mode de fonctionnement de Collgen, il faut se souvenir qu'il y a deux grands types de collocations : les collocations « fixes », dans lesquelles les termes impliqués apparaissent toujours dans le même ordre et à distance fixe les uns des autres, et les collocations « libres », dans lesquelles les termes impliqués peuvent apparaître non seulement dans des ordres variables, mais également à des distances variables les uns des autres.

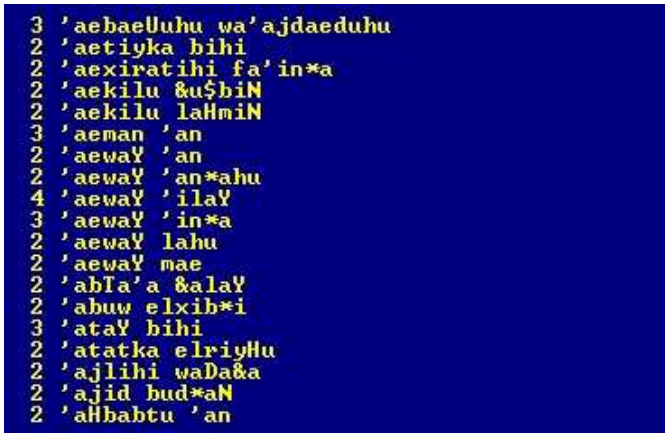
Les collocations fixes d'un texte peuvent être identifiées systématiquement et exhaustivement par Collgen. La seule chose à spécifier au programme est la longueur des contextes droit et gauche dans lesquels il doit rechercher des « segments répétés », selon la terminologie d'André Salem et de son équipe[22], et le seuil de fréquence des répétitions à retenir. Voici à quoi ressemble une page d'interrogation de Collgen avec ces paramètres :



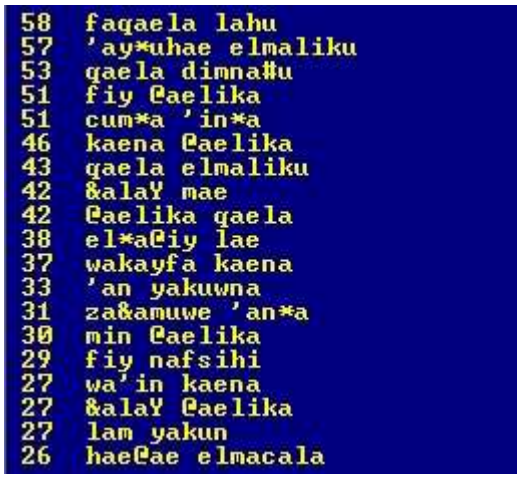
Cette interrogation de Collgen (en cochant « Maximal » pour l'option « Fixed Phrase Format ») génère deux fichiers : l'un, ayant l'extension « .col », donne la liste alphabétique des collocations fixes trouvées ; l'autre, ayant l'extension « .frq », en donne la liste fréquentielle. Pour chaque élément de la liste, la fréquence est données en début de ligne.

Voici le début du fichier « .col » de résultats (qui compte un total de 3 123 paires de mots) :

22. Voir Lafon et Salem (1983). La longueur de ces segments est fixée par défaut à 2. La fréquence des répétitions aussi. Ces valeurs peuvent être modifiées par l'usager.

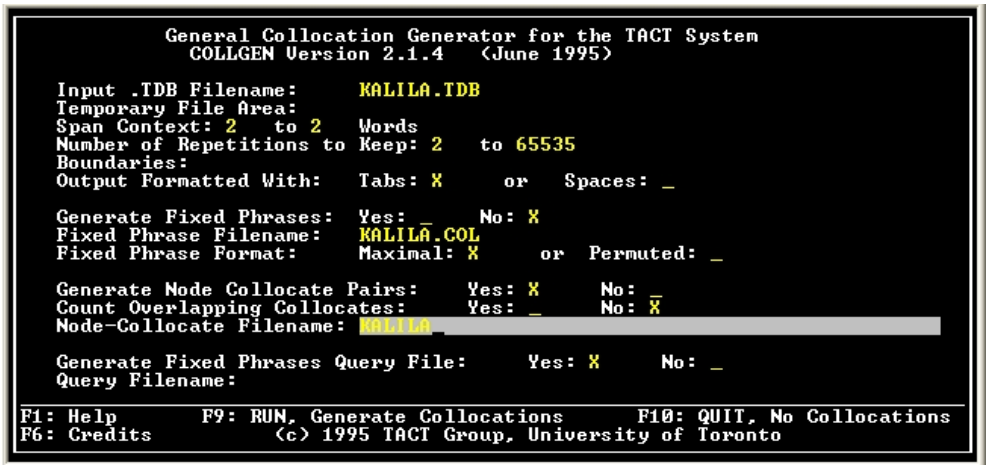


Et voici le début du fichier « .frq » de résultats (le nombre total de paires est évidemment le même) :



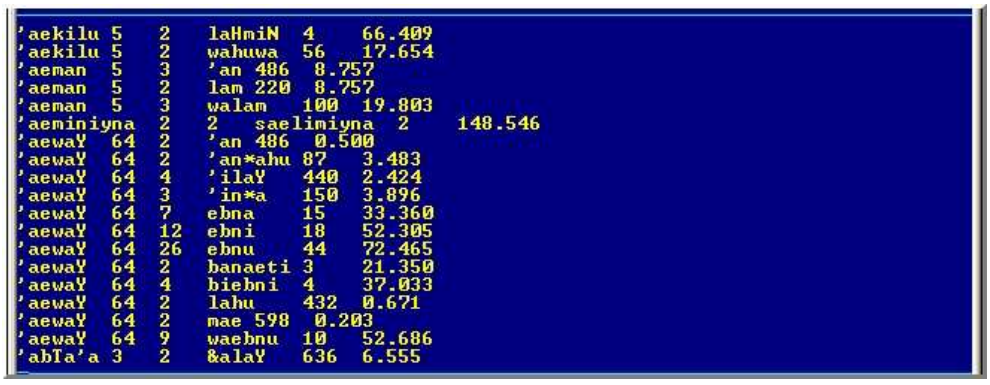
On peut, bien entendu, demander à Collgen de rechercher des collocations fixes de longueur quelconque, sachant que, tendanciellement, plus la longueur est grande, plus courte sera, en principe, la liste des collocations trouvées. Ainsi, à titre indicatif, le fichier des collocations fixes de six mots qui se répètent au moins deux fois dans notre texte est de « seulement » 75 lignes.

Si Collgen sait identifier les collocations « fixes » quelle que soit leur longueur, il ne peut identifier les collocation « libres » que si elles ont une longueur de deux mots. Pour obtenir la liste complète des collocations libres de deux mots, il faut utiliser Collgen, mais en répondant cette fois positivement à la proposition « Generate Node Collocate Pairs : » et en choisissant un nom pour le fichier des résultats (qui aura par défaut une extension « .NOD ». Noter que, bien que la capacité de Collgen à identifier les collocations libres soit limitée à deux mots, on peut, ici encore, choisir la distance à laquelle les deux mots peuvent se trouver (elle est par défaut de 2).



Le fichier résultat généré comporte six colonnes : « mot foyer » de la collocation, fréquence du mot foyer dans le texte, fréquence de la cooccurrence avec le collocat, identité du collocat, fréquence totale du collocat dans le texte et enfin, le Z-score, une valeur statistique qui mesure le degré selon lequel la fréquence constatée de la collocation des deux mots s'écarte de ce qu'elle serait si les mots du texte étaient distribués au hasard.

Examinons un extrait du fichier généré pour notre texte :



On y voit, par exemple, que le mot *'aekilu* qui figure cinq fois dans le texte, y cooccurre 2 fois avec *laHmiN*, lequel figure lui-même quatre fois dans le texte. Le Z-score de cette paire est assez élevé, sanctionnant le fait que le mot *laHmiN* apparaît, pour la moitié de ses occurrences, avec *'aekilu*. Par contraste, la cooccurrence de *'aekilu* avec *wahuwa*, bien qu'ayant la même valeur absolue, est beaucoup moins significative car ce dernier mot apparaît 56 fois dans le texte. Le Z-score plus faible correspond à ce constat.

Si, instruit de ces informations relatives à l'interprétation des Z-scores, on recherche, dans ce fragment de liste, le Z-score le plus élevé, on constate qu'il concerne la cooccurrence de *'aeminiyna* et de *saelimiyna*. Il a en effet, pour cette paire, une valeur de 148 546. La raison en apparaît d'emblée quand on regarde les valeurs respectives de fréquences d'occurrence de ces deux mots dans le texte et

celle de leur collocation : ils apparaissent chacun deux fois et les deux fois ensemble.

On pourra vérifier aussi que la cooccurrence des deux mots *'aewaY* et *ebnu* obtient un Z-score relativement élevé en raison de leur fréquente apparition dans le syntagme *ebnu 'aewaY*.

Signalons, pour finir, que l'on peut fournir à Collgen un fichier d'exclusion lui indiquant les mots dont il doit négliger les collocations (par exemple les mots outils).

Nous en resterons là pour le présent article.

Éléments de bibliographie

BENSON M., 1985, « Collocations and idioms », *Dictionaries, Lexicography and Language Learning*, R. Ilson éd., Oxford, Pergamon Press, p. 61-68.

CRYSTAL D., 1991, *A Dictionary of Linguistics and Phonetics*, Oxford, Cambridge, Blackwell Reference.

FOLEY J. M., 1985, *Oral-Formulaic Theory and Research : An Introduction and Annotated Bibliography*, New York, Garland Publishing.

HABERT B., NAZARENKO A. et SALEM A., 1997, *Les linguistiques de corpus*, Paris, Armand Colin.

LAFON P. et SALEM A., 1983, « L'inventaire des segments répétés d'un texte », *Mots. Les langages du politique,* n° 6, p. 161-177.

LANCASHIRE I. *et al.*, 1996, *Using TACT with Electronic Texts*, New York, The Modern Language Association of America.

LEBART L. et SALEM A., 1994, *Statistique textuelle,* Paris, Dunod.

LIANG S. Q., 1991, « À propos du dictionnaire français-chinois des collocations françaises », *Cahiers de lexicologie*, n° 59, p. 151-167.

MONROE J. T., 1972, « Oral Composition in Pre-Islamic Poetry », *Journal of Arabic Literature*, n° 3, p. 1-54.

OOSTDJIK N. et DE HAAN P. éd., 1994, *Corpus-Based Research into Language,* Amsterdam, Rodopi.