Ecole Nationale Supérieure
des Sciences de l'Information
et des Bibliothèques

DEA en Sciences de l'information
et de la communication

option : Systèmes d'Information Documentaire

MEMOIRE DE DEA

Conception et réalisation d'un prototype de correcteur orthographique de l'arabe

Réalisé par Nabil GADER

sous la direction de M. Mohamed HASSOUN,

Ecole Nationale Supérieure des Sciences de l'Information
et des Bibliothèques

- 1992 -

Conception et réalisation d'un prototype de correcteur orthographique de l'arabe

auteur : Nabil GADER

RESUME : L'objet de ce travail et un système de vérification et de correction automatique de textes arabes. La vérification lexicale est basée sur une analyse du mot, ayant pour tâche d'en identifier les différents constituants (morphèmes). Le dictionnaire utilisé, permet de contrôler la structure morphologique du mot, et de vérifier la compatibilité de ses constituants.

DESCRIPTEURS : correction automatique; erreur; orthographe; typographie; vérification; traitement de texte; analyse morphologique; arabe;

ABSTRACT : In this work, we present a system of automatic verification and correction of arabic texts. The lexical verification is based on an analysis in which word is split up into its component parts (morpheme). The used vocabulary controls the morphological structure of the word and checks the various agreements between its main components.

KEYWORDS : automatic correction; error; orthography; typography; verification; text processing; morphological analysis; arabic;

REMERCIEMENTS

Je tiens à exprimer toute ma gratitude à Monsieur **M. HASSOUN**, maître de conférences à l'ENSSIB, qui a assuré l'encadrement et la direction de mes travaux.

Je tiens à remercier Monsieur **J. DICHY**, professeur à l'Université Lyon 2, pour sa collaboration et ses précieux conseils, ainsi que Monsieur **X. LELUBRE**, maître de conférence à l'Université Lyon 2, pour l'intérêt qu'il a porté à ce travail.

J'exprime mon profond respect et ma reconnaissance à tous les membres du corps enseignant et administratif de l'**ENSSIB** pour leur chaleureux accueil, et particulièrement à Monsieur **R. BOUCHÉ**, professeur et directeur du C.E.R.S.I (Centre d'Etudes et de Recherches en Sciences de l'Information), pour m'avoir accepter dans son équipe. Qu'ils trouvent ici, l'assurance de ma reconnaissance.

Toute ma reconnaissance et mon amitié aux sympathiques étudiants de la promotion 92 du DEA en sciences de l'information et de la communication et plus particulièrement à **Margarita**, **Nadia**, **Wejdane**, **Ion** et **Malek**.

Que toute personne m'ayant apporté son soutien scientifique, matériel, moral ou amical, trouve ici l'expression de mon amitié et de ma reconnaissance.

TABLE DES MATIERES

PREFACE

L'objet de ce mémoire est la mise au point d'un prototype de correcteur orthographique de l'arabe, assisté par un dictionnaire des bases nominales et verbales de l'arabe comportant les relations présentées dans le projet de DEA de M. Malek GHENIMA.

Ce travail s'inscrit dans le prolongement des recherches effectuées à Lyon 1, Lyon 2 et l'ENSSIB, sur le traitement automatique de la morphologie de l'arabe (thèses de MM. M. HASSOUN en informatique [Hassoun 87], J. DICHY en linguistique [Dichy 90], Vol. des Travaux SAMIA I [SAMIA 89], etc.).

Il prolonge par ailleurs la recherche - donnant lieu à la réalisation d'un prototype et à la rédaction d'un mémoire - que j'ai effectué dans le cadre d'un stage de maîtrise à Lyon 2 (Département d'études arabes). Cette recherche, dirigée par M. J. DICHY, portait sur : "*Un système d'enseignement assisté par ordinateur des formes nominales dérivées de l'arabe, avec simulation*" [Gader 90].

Le produit de notre travail servira dans un premier temps, à la correction des fautes orthographiques susceptibles d'apparaître dans les textes ecrits en graphie arabe non-vocalisée. Cette correction pourra s'étendre par la suite à la correction des requêtes dans l'interrogation des bases d'information. Ceci sera possible par la prise en compte dans le dictionnaire utilisé des termes techniques ainsi que des noms propres de personnes ou de lieux.

Dans une première approche de la vérification et de la correction automatique nous ne traiterons que les fautes orthographiques de niveau lexical (fautes de frappe, etc.).

Le correcteur ainsi obtenu, pourra par la suite être utilisé dans la prise en compte des erreurs syntaxico-sémantiques pouvant affecter les textes ou les requêtes d'interrogation, et dont la correction nécessite l'utilisation de connaissances syntaxiques et sémantiques.

I. *INTRODUCTION A LA CORRECTION ORTHOGRAPHIQUE*

L'étendue et l'utilisation croissante des ordinateurs dans la manipulation des bases de données textuelles par un éventail toujours plus large de personnes pose le problème de la communication homme-machine.

Depuis l'avènement des ordinateurs, la tendance dans les interactions entre l'homme et la machine a été de placer le plus gros de la communication sur la machine plutôt que sur l'homme.

Parmi les moyens d'accès conviviaux opérationnels, la langue naturelle écrite est l'une des plus souples et des plus agréables qui soit pour l'homme. Mais dans le cas d'une utilisation grand public, le problème de la tolérance de certaines erreurs apparaissant dans les requêtes se pose de façon aiguë. Tout système incapable de prendre en compte les fautes de frappe et d'usage risque de décourager rapidement la plupart de ses utilisateurs. Pour améliorer la convivialité d'une interface en langue naturelle, il semble donc important de résoudre cette difficulté.

Les ordinateurs sont capables de détecter les erreurs. Cependant, si une erreur apparaît, le système est souvent incapable de continuer son traitement et demande à ce que l'entrée soit réintroduite correctement (dans le cas des dialogues d'applications et les accès à des bases de données cela se traduit par un échec de la recherche ou un message d'erreur). Mais si l'ordinateur est si adroit dans la détection des erreurs, pourquoi est-il si intolérant ? Quand une personne communique avec une autre, des erreurs fréquentes s'introduisent dans le discours sans pour autant l'interrompre.

Il serait donc souhaitable qu'un système informatique puisse développer une capacité de tolérance aux fautes similaire à celle des hommes.

En effet, pour une communication homme-machine plus évoluée et plus souple, il parait indispensable d'utiliser des algorithmes de correction, et de pouvoir retrouver un mot voisin en cas d'erreur.

De même, après une saisie de textes par lecture optique, la mise en oeuvre d'un accès lexical s'avère indispensable pour détecter, et corriger si possible les erreurs résiduelles après reconnaissance des caractères.

La saisie manuelle n'est elle-même pas exempte d'erreurs, si bien que cette fonctionnalité est aussi utile avant ou pendant tout traitement d'un texte en langage naturel.

Les recherches sur la vérification et la correction automatique ont débuté aux Etats-Unis vers la fin des années 50 et se poursuivent jusqu'à nos jours dans bon nombre de laboratoires aussi bien aux Etats-Unis qu'en Europe.

Les premiers traitements tolérants aux erreurs concernent la recherche lexicographique, ensuite ils se sont étendus à la compilation et l'interrogation des bases d'information, pour aboutir enfin à la compréhension automatique du langage naturel et l'interrogation des systèmes experts.

Nous savons que toutes ces applications ne peuvent se faire sans que s'effectuent à un degré plus ou moins important des traitements linguistiques de tout niveau, tant morphologique et syntaxique que sémantique et pragmatique.

Comme chacun sait, de nombreux travaux ont été développés dans ce domaine pour des langues comme l'anglais, le russe et le français à un point que l'analyse morphologique et syntaxique ne semble plus faire partie des préoccupations essentielles des chercheurs travaillant dans ce domaine aujourd'hui.

Hélas, pour la langue arabe qui pourtant fait partie des cinq ou six langues les plus parlées du monde, on est encore loin de ces résultats. Mais il faut noter que plusieurs équipes de recherches travaillent sur le sujet que ce soit dans les pays arabophones ou non-arabophones et que plusieurs travaux ont déjà été développés ou en cours de développement.

En ce qui concerne la vérification et la correction automatique de l'arabe, la plupart des recherches effectuées l'ont été dans une perspective d'Enseignement Assisté par Ordinateur [Hamadou 89], [Souilem 89].

Le prototype de correcteur orthographique qu'on se propose de réaliser dans le cadre de ce mémoire touchera aussi bien à l'E.A.O qu'aux autres domaines de la vérification et de la correction automatique que nous verrons un peu plus loin.

Il est important de signaler que dans ce qui suit, nous laissons de coté le problème des fautes de nature grammaticales. En effet, leur correction semble nécessiter de modéliser en machine le savoir de linguistes, sous la forme de règles de systèmes experts par exemple. De plus ces correcteurs sont encore souvent cantonnés au traitement de catégories grammaticales limitées comme l'accord des participes passés dans le cas de la langue française [Lapalme 86].

Imaginons par exemple qu'un utilisateur écrive malencontreusement « j'ai allé ». C'est incorrect, nous le savons. Mais notre malheureux correcteur orthographique du français, lui, ne verra que des termes existants, et ne se souciera en aucun cas de vérifier leur bon ordonnancement, ni leur adéquation les uns avec les autres. Même chose pour l'arabe, le correcteur orthographique comme nous l'envisageons dans cette étude, ne décèlera pas d'anomalies dans la phrase «عاقبت المعلّم التلميذ» (l'enseignant a puni l'élève). Ce genre de fautes demande de connaître le rôle syntaxique de chaque mot dans la phrase ainsi que les règles d'accord qui régissent leur écriture correcte dans ce rôle.

Nous nous intéressons donc uniquement aux fautes orthographiques d'usage et dactylographiques. Les premières découlent de la mauvaise transcription de la forme orale, un même phonème ayant plusieurs traductions possibles. Les deuxièmes peuvent résulter de l'utilisation d'un clavier, de la présence de parasites sur la ligne de transmission, d'une reconnaissance imparfaite des caractères lors d'une lecture optique ou d'une combinaison de ces causes. Signalons tout de même qu'une telle correction orthographique est une étape indispensable à franchir avant de pouvoir passer à la correction des erreurs grammaticales.

Nous passerons d'abord en revue les différents domaines de la vérification et de la correction automatiques que nous définirons en général comme suit :

Vérification : Soit un lexique L constitué de chaînes X_r ($r = 1, ..., N_m$, N_m étant le nombre de mots du lexique). Vérifier une chaîne Y consistera à rechercher s'il existe r tel que $Y = X_r$. Si c'est le cas, la vérification est positive. Sinon il y a échec.

Généralement, si la vérification est positive, elle ne donne lieu à aucune action spécifique.

En cas d'échec, il appartient à l'usager de déterminer si *Y* est erronée ou si *L* est incomplet et, en conséquence, si une action doit être entreprise.

Correction : Etant donné le lexique *L* et la chaîne Y, en cas d'échec de la vérification, la correction vise à fournir une ou plusieurs chaînes de *L* voisines de *Y* au sens d'un critère donné.

Le choix de la bonne correction peut être laissé à l'utilisateur. Celle-ci peut également être effectuée impérativement.

Après les différents domaines de la vérification et de la correction automatique, nous définirons les différents types d'erreurs concernées par la correction orthographique, ainsi que les méthodes de vérification et de correction les plus utilisées et celles que nous adopterons pour la vérification et la correction orthographique des textes arabes écrits en langage naturel.

II. *DOMAINES DE LA VERIFICATION ET DE LA CORRECTION AUTOMATIQUE DE TEXTES**

Les situations où l'on peut faire appel à la vérification et/ou à la correction automatique sont très diverses : mots isolés ou en contexte, noms communs, noms propres, étiquettes ou termes divers dans des programmes.

Nous allons illustrer ceci au moyen de quelques exemples qui nous semblent représentatifs.

II.1. ACCES TOLERANT AUX FAUTES DANS LES BASES D'INFORMATION

Lorsque la clé de recherche pour consulter une base de donnée est un terme technique, un nom savant ou un nom propre (de personne ou de lieu), plus généralement lorsque la clé n'est connue qu'avec une certaine imprécision par une partie des utilisateurs, il peut se révéler utile de prévoir un accès tolérant au fautes.

Accessoirement, de tels accès permettent de pallier certaines fautes de codages se produisant lors de la constitution de la base (de telles fautes pourraient interdire l'accès avec une clé juste, sauf s'il y a tolérance aux fautes). L'étude de Bourne [Bourne 77] a montré que ces erreurs de codage ne sont pas à négliger dans certaines banques de données documentaires.

En France, un exemple d'application de ce type est fourni par l'annuaire électronique mis en place en 1981. Il permet l'interrogation du fichier des

* Les chapitres II, III et IV sont une synthèse bibliographique faite à partir de travaux publiés dans le domaine de la vérification et de la correction automatique.

abonnés d'après l'orthographe approximative de leur nom - mais avec la prononciation exacte.

II.2. LECTURE AUTOMATIQUE DE CARACTERES

L'acquisition de données textuelles par lecteurs optiques expose à des erreurs de reconnaissance des caractères même si l'on adopte des polices spécialement étudiées pour faciliter les discriminations.

Leur correction peut intervenir à deux niveaux :

a) au moment de la prise de décision de reconnaissance du caractère par l'utilisation des probabilités de digrammes, trigrammes, etc.(il existe beaucoup de travaux de ce type : [Harmon 62], [Carlson 66], [Riseman 74], [Fisher 76], [Ullman 77]);

b) sur le texte obtenu par consultation d'un lexique (voir § II.4).

II.3. ASSISTANCE A LA MISE AU POINT DE PROGRAMMES

Une étude statistique d'erreurs dans les programmes écrits en COBOL par Litecky et Davis [Litecky 76] a montré que 20% des fautes à la compilation (c'est à dire «syntaxique») étaient d'origine orthographique ou typographique.

En fait la question des fautes de frappe dans les programmes s'est posée dès les premiers développements de l'informatique. Ainsi se sont développées des techniques de compilation tolérante au fautes ([Freeman 63], [Morgan 70], [James 76]).

II.4. TRAITEMENT DE TEXTE, BUREAUTIQUE, PRESSE

Dès que l'on a voulu traiter des données textuelles sur ordinateur, la question des logiciels d'aide à la vérification et ou à la correction des fautes s'est posée ([Damerau 64], [Riseman 71], etc.), mais rares sont ceux qui ont abouti à une application commerciale.

En effet, l'utilisation de lexiques étant nécessaire dans le cas de textes généraux, se posent immédiatement les questions de l'occupation de mémoire et du temps de calcul. Deux programmes de vérification disponibles sous UNIX peuvent cependant être cités : SPELL écrit par R. Gorin à Stanford en 1971 et TYPO [Morris 75], [McMahon 78].

Avec l'essor actuel de la bureautique et du traitement de texte, les système de vérification automatique se généralisent sur les ordinateurs personnels. Une tendance actuelle, non sans rapport avec ce nouveau créneau que constitue l'*industrie de la langue*, est la participation des maisons d'édition : ainsi Larousse est l'auteur d'Orthogiciel pour Macintosh et y utilise ses propres *matériaux lexicaux*.

Les performances restent encore insuffisantes, aussi bien pour les temps de réponse que pour l'étendue des vocabulaires de base utilisés. Mais nul doute que des progrès constants seront à enregistrer dans ce domaine. Destinés à des machines puissantes, il faut noter aussi la nouvelle génération de systèmes d'aide à la rédaction de textes comme EPISTLE [Heidorn 82].

A noter que des techniques de systèmes experts liées à celle des linguistes font leur apparition dans le domaine de la correction des fautes grammaticales [Lapalme 86].

Un domaine voisin est celui de la presse : un article de dernière minute, mis rapidement sous presse, contient souvent de nombreuses fautes qui peuvent nuire à la qualité de la rubrique. Dans ce cas, l'utilisation de correcteurs automatiques, même s'il ne corrigent qu'une partie des fautes pourrait se révéler intéressante. Mais malgré l'informatisation poussée des entreprises de presse, il ne semble pas à l'heure actuelle que ce type d'application soit effectivement mis en place à grande échelle.

Un des obstacles réside dans la taille et la complexité des lexiques nécessaires, où doivent figurer, en particulier des noms propres divers, des sigles, etc. Cependant, en cernant bien les applications et les tâches de rédaction, il ne semble pas que ces difficultés soient insurmontable. Il est même envisagé à l'avenir des postes de travail permettant d'automatiser, au moins partiellement, la rédaction des dépêches - voir [Pavard 85] pour une discussion plus approfondie.

II.5. ENSEIGNEMENT ASSISTE PAR ORDINATEUR

L'ordinateur devient de plus en plus un partenaire de la vie quotidienne. A ce titre, il apparaît comme un nouveau média d'information et de formation.

Son introduction dans l'enseignement en tant qu'instrument puissant de rénovation pédagogique a permis d'une part, l'acquisition de connaissances plus ou moins structurées, et d'autre part, l'approfondissement et le perfectionnement des connaissances acquises et ce par la mise en place de nouvelles techniques adaptées aux profils des apprenants.

Contrairement à l'ancienne génération de didacticiels, les nouveaux systèmes d'Enseignement Assisté par Ordinateur ne se contentent plus de

réagir aux réponses des apprenants par un "oui" ou par un "non", mais disposent de plus en plus de nouvelles techniques leur permettant d'analyser les réponses incorrectes.

C'est ainsi qu'un didacticiel d'enseignement de l'orthographe arabe sous forme d'expression libre pourrait, après vérification, dresser la liste des mots erronés avec leurs localisations dans le texte et donner à l'apprenant l'opportunité de corriger ses erreurs. A la suite de l'autocorrection, le système effectue une nouvelle correction et restitue pour chaque erreur persistante la forme correcte correspondante, le type d'erreur et un commentaire résumant la règle orthographique mise en jeu pour la correction.

Un exemple de didacticiel de ce type est le *Système d'Enseignement Assisté par Ordinateur des Verbes Arabes* réalisé par M. Ghenima dans le cadre du programme SAMIA qui, dans sa première version, permet de traiter les erreurs de confusion entre les pronoms [Ghénima 90].

III. *TYPOLOGIE DES FAUTES*

Dans une chaîne de production d'un texte*, les fautes peuvent s'introduire à divers niveaux que nous regrouperons en deux : pour simplifier le *niveau orthographique* et le *niveau typographique*.

III.1. LES FAUTES ORTHOGRAPHIQUES

Sur l'histoire et la pédagogie de l'orthographe, il existe de nombreux ouvrages et articles : voir par exemple [Catach 82] pour une référence sur la langue française.

En ce qui concerne l'arabe, une étude typologique et statistique des erreurs orthographiques parue dans "La Revue Tunisienne des Sciences de l'Education" mérite d'être citée. Son objectif était de prendre connaissance des difficultés de l'orthographe arabe et de disposer de données statistiques sur les erreurs susceptibles d'apparaître dans les testes appréhendés.

Les résultats de cette étude sont résumés dans des tableaux en annexe. Pour plus de détails sur cette étude voir [Hamadou 87].

Lorsqu'ils se risquent à une classification, ce qui ressort le plus nettement chez les auteurs, c'est l'opposition entre les fautes d'usage et celles où dominent les erreurs due à l'ignorance des différentes règles orthographiques, qui font notamment intervenir des relations grammaticales telles que l'accord en genre et en nombre (ainsi le *Grand* dictionnaire encyclopédique *Larousse* (15 vol.) distingue entre *orthographe d'usage* et *orthographe d'accord*).

*Les textes à corriger sont envisagés ici en toute généralité, qu'il s'agisse de textes en langage naturel, de programmes pour ordinateurs ou de mots isolés (clé d'accès à une base, mot usuel ou terme technique).

Les fautes d'usage proviennent essentiellement du manque de concordance entre l'écrit et l'oral.

Ainsi, le même phonème peut avoir plusieurs transcriptions selon les mots considérés, comme par exemple en français le phonème /k/ de «acoustique» de «kiosque» et de «khmer». Ces complication peuvent donner lieu à des fautes classiques telles que «accoustique», «acoustic»,«kiosc» ou «kmer» etc.

En arabe, ce genre de fautes est accentué par plusieurs autres facteurs comme par exemple la difficulté de distinction entre certaines lettres telles que le « ذ » et le « ظ », le « ص » et le « س » et le « ت » et le « ط » qui ne sont distinguées que par le trait phonétique de l'emphase. Ainsi une mauvaise connaissance du vocabulaire ou l'influence de l'arabe dialectal peut donner lieu à des fautes comme l'écriture du mot « مستـرة » à la place de « مسطرة » (règle).

Quant à la mauvaise transcription de la forme orale (transcription trop "phonétisée"), elle peut donner des erreurs comme :

- « ذا لـك » à la place de « ذلـك » (cela).
- « هـا ذا » à la place de « هذا » (ceci).
- « ا بـتعدتـم » à la place de « ا بـتعدتـم » (vous vous êtes éloignés).
- « ا صبـتق » à la place de « ا نبـتق » (il a émergé).

III.2. LES FAUTES TYPOGRAPHIQUES

Dites aussi dactylographiques, ces fautes sont introduites à la saisie des textes et souvent liées à l'usage des claviers de machines à écrire. Nous pouvons classer ici également les fautes matérielles diverses dont l'origine est indépendante des difficultés de l'orthographe : erreurs de reconnaissance

dans les lecteurs optiques, erreurs de transmission télématique, erreurs d'encodage dans les bases de données textuelles, etc.

III.3. CONCLUSION

Ces deux catégories de fautes se traduisent dans les chaînes de caractères par l'apparition des anomalies suivantes :

Type d'erreur	Exemple	Origine de l'erreur
omission d'un caractère	« صفة » à la place de « صيغة »[1]	usage ou dactylographie
insertion d'un caractère	« ذالك » à la place de « ذلك »[2]	usage ou dactylographie
substitution d'un caractère par un autre	« مستـرة » à la place de « مسطرة »[3]	usage ou dactylographie
permutation de deux caractères	« هاذ » à la place de « هذا »[4]	usage ou dactylographie

On constate à la lumière de ces exemples que les fautes d'usage se confondent parfois avec les erreurs purement dactylographiques. La chaîne incorrecte « صفة » peut en effet résulter, soit d'une mauvaise frappe (omission de la lettre «ـيـ»), soit d'une mauvaise connaissance de l'orthographe.

Tous les travaux menés sur le sujet - hors du domaine arabe - confirment que ces quatre anomalies sont largement majoritaires. F.J. Damerau chiffre leur fréquence à 80% des erreurs pouvant affecter un mot [Damerau 64], valeur confirmée par une étude statistique menée sur un quotidien français [Pérennou 86]. En considérant également les combinaisons dans un même mot de ces quatre défauts élémentaires, on couvre plus de 80% des erreurs possibles.

[1] Forme.
[2] Cela.
[3] Règle.
[4] Ceci.

A l'examen des différentes méthodes de correction automatique, il apparaît que les auteurs envisagent le texte comme une suite de chaînes de caractères perturbées par des fautes se ramenant à l'une des catégories déjà citées en y ajoutant les cas de :

- coupure de mot (insertion de blanc)

Exemple : « إ ن بـتـق » à la place de « انـبـتـق » (il a émergé).

- et de soudure de deux mots (omission de blanc).

Exemple : « إ نـشا ء » à la place de « إ ن شا ء » (s'il veut)

IV. *LES APPROCHES DE LA VERIFICATION ET DE LA CORRECTION AUTOMATIQUE DE TEXTES*

Un programme de correction d'erreurs lexicales nécessité deux fonctions de base :

- une fonction de vérification, qui contrôle qu'un mot entré est correct ou non (c'est à dire présent ou non dans le dictionnaire).

- une fonction heuristique, qui propose des voisins plausibles en cas d'erreur (lorsque le mot n'existe pas dans le dictionnaire).

IV.1. LA VERIFICATION AUTOMATIQUE

Elle consiste essentiellement à classer les mots en deux catégories selon qu'ils figurent, ou non, dans le lexique.

Une question importante est alors celle de l'étendue de ce lexique. Elle peut être considérablement réduite en utilisant la décomposition morphologique afin d'éviter le stockage de toutes les formes dérivant d'une même racine (sur ce sujet voir les travaux effectués au Laboratoire d'Automatique Documentaire et Linguistique de l'Université Paris 7 pour la réalisation du dictionnaire informatique du français baptisé DELAS).

D'un point de vue plus fondamental, des études linguistiques ont permis de mettre en évidence que quelques centaines de morphèmes sont suffisants pour rendre compte de 50 000 mots français [Gruaz 86].

Cette question peut aussi être abordée sous l'angle de la structure du vocabulaire des textes. On a depuis longtemps observé sur diverses langues qu'un vocabulaire limité à moins d'une centaine de mots peut couvrir 50% d'un texte; semblablement pour le français : mille mots couvrent 85% et 4 000 mots couvrent 97,5% d'un texte - sur ces questions voir [Guiraud 59].

Ces observations suggèrent que l'on range les mots fréquents dans un lexique à part, de manière à obtenir un traitement et un accès rapide pour la plupart des mots du texte. C'est la stratégie à deux niveaux de Peterson [Peterson 80, 80a] qui prévoit un petit lexique de 258 mots les plus fréquents de l'anglais pour des procédures accélérées.

IV.2. LA CORRECTION AUTOMATIQUE

La correction d'une chaîne de caractères ne figurant pas dans le lexique et, de ce fait, présumée mal écrite, consiste à proposer en remplacement un mot ou une suite de mots la corrigeant au mieux, selon un critère donné. Une variante existe où plusieurs solutions de remplacement sont proposées, les meilleurs au sens du critère; la décision finale de la correction est laissée à l'utilisateur.

La difficulté de la correction réside donc dans la détermination du ou des mots candidats à la correction de la chaîne erronée.

Les méthodes utilisées sont multiples. Nous verrons dans ce qui suit quelques unes de ces méthodes dont l'utilisation dépend des applications que l'on a en vue.

IV.2.1. Méthodes combinatoires

Généralement les auteurs proposent des méthodes corrigeant une faute au plus par mot.

A partir de la chaîne erroné *CE*, on génère toutes les chaînes possibles CP_i dont *CE* pourrait dériver par omission, insertion, substitution d'un caractère. L'intersection entre l'ensemble des CP_i et le lexique *L* fournit une ou plusieurs chaînes candidates à la correction. Si l'intersection se réduit à un mot, celui ci est supposé corriger *CE*; sinon ou bien l'intersection est vide et il n'y à pas de correction possible, ou bien il y a une ambiguïté et l'utilisateur devra intervenir.

L'élaboration des CP_i pose des problèmes d'explosion combinatoire. Afin de les réduire, on ne prend en compte le plus souvent qu'une seule erreur par mot [Pollok 84], [Peterson 80]. Les procédures peuvent être considérablement accélérées dans ce cas.

Malgré cette importante restriction, la correction d'une chaîne nécessite de nombreux accès au dictionnaire. Un mot de n lettres issues d'un alphabet de m caractères peut dériver de $m(2n + 1) + n - 1$ chaînes, s'il ne présente qu'une seule anomalie [Hall 80].

Des considérations heuristiques peuvent contribuer à diminuer cette valeur; elle reste cependant élevée : Peterson a constaté une moyenne de 200 accès au lexique pour corriger un mot [Peterson 80].

IV.2.2. Méthodes statistiques

Elles sont basées sur les fréquences observées d'apparition de lettres dans les mots de la langue considérée. Il existe des dictionnaires de digrammes (séquence de deux lettres) et de trigrammes (trois lettres) [Angell 83].

Les performances sont inférieures à celles que l'on peut obtenir par utilisation des lexiques, mais revanche les besoins de stockage sont réduits. De plus on évite le risque de se trouver en présence de mots corrects ne figurant pas dans le lexique.

Les algorithmes qui utilisent ces méthodes permettent surtout de corriger des fautes de substitution, qui n'interviennent que dans 20% des cas [Pérennou 86]. De plus, leur efficacité est moindre sur les mots courts puisqu'une simple erreur peut perturber une forte proportion de leurs trigrammes.

Si cette approche est bien adaptée à la lecture optique où dominent les fautes de substitution, en revanche son emploi dans la correction de textes saisis au clavier semble moins indiqué.

Lorsque le lexique est utilisé, le problème central est celui de la comparaison de la chaîne erronée avec les mots du lexique pour en extraire les plus ressemblants. L'utilisation des modèles statistiques dans ce cas n'est pas très répandue. Le travail de Kashyap et Oommen [Kashyap 84] en a pourtant montré l'intérêt pour la correction de multiples fautes typographiques. L'algorithme proposé par les auteurs s'apparente par sa structure à un algorithme de programmation dynamique.

IV.2.3. Méthode des alphacodes

Cette méthode a été mise au point pour le système de documentation automatique SPIRIT. Nous nous référons à la description qui en est faite dans [Andreewsky 78] par les concepteurs de ce système.

L'alphacode d'un mot est la chaîne de caractères constituée de l'ensemble de ses lettres classées par ordre alphabétique :

ALLIANCE et CANAILLE ont pour alphacode AACEILLN;
MASSACRE et SARCASME ont pour alphacode AACEMRSS;
CADEAU et AUDACE ont pour alphacode AACDEU.

L'ensemble des mots relatifs à un alphacode donné est appelé classe de cet alphacode. Les auteurs constatent que la grande majorité des classes se réduit à un seul mot. Les exemples ci-dessus forment donc exception. En français, le cardinal moyen d'une classe est de 1,05. En anglais il est de 1,10.

Le système de correction dispose de l'ensemble des alphacodes correspondant au lexique et, pour chacun, de la classe de mots correspondante. Le correcteur établit tout d'abord l'alphacode du mot erroné. Si cet alphacode est répertorié, les mots de la classe correspondante sont retenus comme candidats. Si l'alphacode est inconnu, le système y ajoute un, puis deux caractères, et vérifie si le nouvel alphacode obtenu existe. Dans ce cas, la classe est retenue. Il procède de même en supprimant puis en substituant un ou deux caractères. Les mots candidats obtenus font alors l'objet d'un calcul de proximité avec la chaîne incorrecte, ce qui autorise un classement de cette liste de mots correctifs.

Ce procédé est donc capable de corriger les mots qui présentent :

- un nombre quelconque de permutations;
- une ou deux omissions, insertions ou substitutions.

L'intervention des alphacodes des alphacodes réduit fortement la combinatoire de la recherche des mots candidats. Ainsi il est possible de prendre en compte deux erreurs de type omission, insertion ou substitution, outre les permutations.

Dans ses premières versions, ce procédé de correction exigeait de nombreux accès au lexique. Des contraintes (similaires à celles des méthodes statistiques) sont utilisées lors de l'élaboration des alphacodes dérivés du mot erroné : on limite ainsi le nombre d'alphacodes générés et donc les accès au lexique.

Ce procédé présente, a nos yeux, l'inconvénient de nécessiter un deuxième lexique qui recense les alphacodes et leurs correspondants dans le lexique.

IV.2.4. Méthodes métriques

Elles consistent à calculer une distance qui traduit les différences entre la chaîne erronée *CE* et une chaîne *C* du lexique.

Pour comparer la chaîne *CE* à une chaîne *C* du lexique, il est possible d'envisager des métriques $d(C, CE)$. Le principe de correction peut se schématiser ainsi : corriger *CE* en *C* si *C* est le mot du lexique le plus proche de *CE* et si $d(C, CE)$ et inférieur à un certain seuil de rejet.

Une métrique classiquement utilisée est celle de Levenshtein [Levenshtein 66], [Okuda 76]; elle est fonction du nombre d'opérations nécessaires à la transformation de *CE* en *C*, chaque type d'opération étant affecté d'un poids particulier basé sur les statistiques d'erreurs.

L'attrait pour la distance de Levenshtein s'est trouvé considérablement accru après que Wagner et Fisher [Wagner 74] aient donné un algorithme de programmation dynamique pour son calcul dans le cas où les transpositions sont exclues - à noter dans [Okuda 76] un circuit spécialisé pour la mise en oeuvre d'un algorithmes du même type. Ces transpositions sont prise en compte dans l'extension donnée par Lowrance et Wagner [Lowrance 75].

Divers travaux ont suivi pour étudier la complexité de ces algorithmes [Wong 76] et pour apporter quelques améliorations : cas de coûts en nombres entiers [Masek 80], organisation de la source lexicale et optimisation des accès.

Cette méthode, particulièrement efficace, est très onéreuse en temps de calcul dès que la lexique est volumineux. Récemment des réalisations à base de circuits intégrés VLSI (Very Large Scale Integration) ont permis des gains de vitesse de l'ordre de 10 à 60 selon les machines considérées [Frison 88].

IV.2.5. Méthodes phonétiques

Le principe en est connu : il est utilisé par l'annuaire électroniques du Minitel. Il équipe également le système de consultation en langue naturelle des pages jaunes [Clémencin 88]. La représentation phonétique des mots permet de tolérer toute faute n'affectant pas leur prononciation : c'est le cas de beaucoup de fautes d'usage. Cependant, elle ne prend pas en compte la

plupart des fautes de frappe car ces dernières compromettent la bonne prononciation des mots.

Dans cette méthode, deux mots sont considérés comme équivalents lorsqu'ils se prononcent de manière identique. Quand l'utilisateur propose une graphie, il accède en fait à tous les noms équivalents phonétiquement.

Les méthodes fondées sur l'équivalence de prononciation ne prennent pas en compte le fait que la probabilité de fautes d'usage est liée à un ensemble complexe de facteurs parmi lesquels interviennent à la fois la (ou les) graphie(s) du dictionnaire, la prononciation et le degré de familiarité avec le mot.

De plus, leur tolérance aux fautes et limitée à celles qui préservent la prononciation; elles sont donc peu appropriées au traitement de texte et à toutes les applications où se produisent des fautes typographiques.

Plus généralement, la démarche consistant à représenter un mot soit par un ensemble de propriétés, soit par un *squelette* supposé résistant aux fautes typographiques et orthographiques, a été envisagée dès les premiers travaux dans ce domaine. Ainsi, Blair [Blair 60] propose un système d'abréviations de quatre à cinq lettres; une pondération leur est attribuée en fonction de leur pouvoir discriminant. Ici encore la tolérance ne s'étend pas au fautes typographiques qui peuvent affecter les lettres retenues dans le squelette (voir aussi [Davidson 62]).

Pour pallier cet inconvénient, certains auteurs ont tenté de combiner les méthodes phonétiques et les algorithmes ci-dessus, plus à même de corriger les erreurs dactylographiques. Van Berkel et De Smedt [Berkel 88] utilisent

des *trigrammes phonétiques*, ou *triphones*, sur le principe des méthodes statistiques (*Cf.* IV.2.2). Ils retrouvent les problèmes de correction des mots courts : une erreur de frappe peut modifier la majorité de leurs triphones.

J. Véronis, lui étend les méthodes métriques au calcul des *distances phonétiques* entre deux mots [Véronis 87]. A cette fin, il a établi des dictionnaires de la langue française, c'est-à-dire des couples différents formés d'un graphème et de son pendant phonématique [Véronis 86]. Afin d'obtenir des temps de réponse raisonnables, il est contraint d'adopter certaines restrictions sur les erreurs prises en compte :

- une seule anomalie dactylographique par mot,
- pas d'erreur de frappe sur la première lettre des mots.

IV.3. L'OPTIMISATION DES NOMBRES D'ACCES LEXICAUX

La question de la vérification et de la correction automatique vue sous l'angle de l'optimisation du nombre d'accès à un fichier lexical est bien analysée dans Hall et Dowling [Hall 80] auquel nous renvoyons pour plus de détails (pour les questions plus classiques de techniques d'accès voir [Knuth 73].

Bornons-nous à indiquer que diverses solutions envisagées utilisent une partition de lexique. En présence d'une chaîne de caractères, un test préalable oriente vers les sous-lexiques où la recherche est opportune.

Parmi les modes de partition utilisées, les plus simples consistent à regrouper soit les mots de même longueur - proposé par Morgan [Morgan 70] et Szanzer [Szanzer 68] -, de même initiale - proposé par Muth et Tharp

[Muth 77] ou ayant les mêmes deux premières lettres - proposé par Morgan [Morgan 70]. Le système SPELL utilise la combinaison du critère de longueur et de digramme initial.

Les défauts de ce type de méthodes sont assez clairs : le manque d'équilibre entre les classes, certaines ayant de faibles effectifs, le manque de tolérance à diverses fautes - en particulier si l'on classe selon les initiales, il faut prévoir des procédures pour rattraper les fautes sur les débuts des mots, fautes dont l'effet est l'orientation erroné vers une classe. De plus elles demandent un nombre de consultations de mots proportionnel à la taille N du lexique.

Lorsque le temps d'examen d'un mot consulté n'est pas négligeable, par exemple s'il y a calcul de la distance de Levenshtein, il est possible d'accepter les procédures plus complexes permettant de rendre le nombre de consultations proportionnel à $\log(N)$.

Un tel résultat peut être obtenu après une classification hiérarchique des mots, chaque niveau de la hiérarchie consistant en une partition du lexique affinant celle du niveau supérieur. Idéalement, à un niveau donné k, une classe $C_j(k)$ regroupe les éléments X dont la distance à un mot $c_j(k)$ est inférieure à $r_j(k)$ donné. Le principe de la consultation consiste à descendre dans la hiérarchie par utilisation de la distance aux centres des sous-classes. Dans chaque classe l'appartenance à une sous-classe donnée exclut celle à d'autres sous-classes en vertu de l'inégalité triangulaire, ce qui permet précisément la réduction du nombre de consultations.

V. *LA VERIFICATION ORTHOGRAPHIQUE DE L'ARABE*

V.1. INTRODUCTION

La vérification d'un texte sur le plan lexical est l'opération qui consiste à classer ses différents mots selon le critère d'appartenance ou de non appartenance au vocabulaire.

Le principal problème que pose cette opération est le choix du vocabulaire et de sa représentation en machine qui se heurte au problème de l'étendue de la mémoire.

On a déjà vu dans le paragraphe IV.1 que plusieurs approches peuvent être utilisées afin d'optimiser l'espace mémoire nécessaire pour loger un important vocabulaire. L'approche que nous avons retenue pour notre étude est basée sur la structure linguistique du vocabulaire. En effet, le vocabulaire de la langue arabe est représenté en se référant à sa structure redondante à savoir la possibilité de décomposer ses mots en leurs éléments premiers.

Nous parlerons dans ce qui suit du contenu et de l'organisation du dictionnaire que nous utiliserons pour notre correcteur. Nous détaillerons par la suite, les différentes étapes de la vérification orthographique.

V.2. CONTENU ET ORGANISATION DU DICTIONNAIRE

Le vérificateur et le correcteur orthographique de l'arabe, objets du présent travail, font appel à un dictionnaire informatisé de l'arabe.

La conception est la réalisation de ce dictionnaire font l'objet d'un mémoire de D.E.A en sciences de l'information et de la communication, en cours de réalisation à l'ENSSIB par M. GHENIMA pour l'année 91/92. Ils prolongent par ailleurs les travaux de recherche effectués à Lyon 1, Lyon 2 et l'ENSSIB sur le traitement automatique de la morphologie de l'arabe : thèses de M. HASSOUN en informatique [Hassoun 87], J. DICHY en linguistique [Dichy 90], etc.

Ce travail à permis de mettre au point un prototype de dictionnaire des bases nominales et verbales de l'arabe comportant :

- Un ensemble de relations entre les bases relevant d'une même racine.

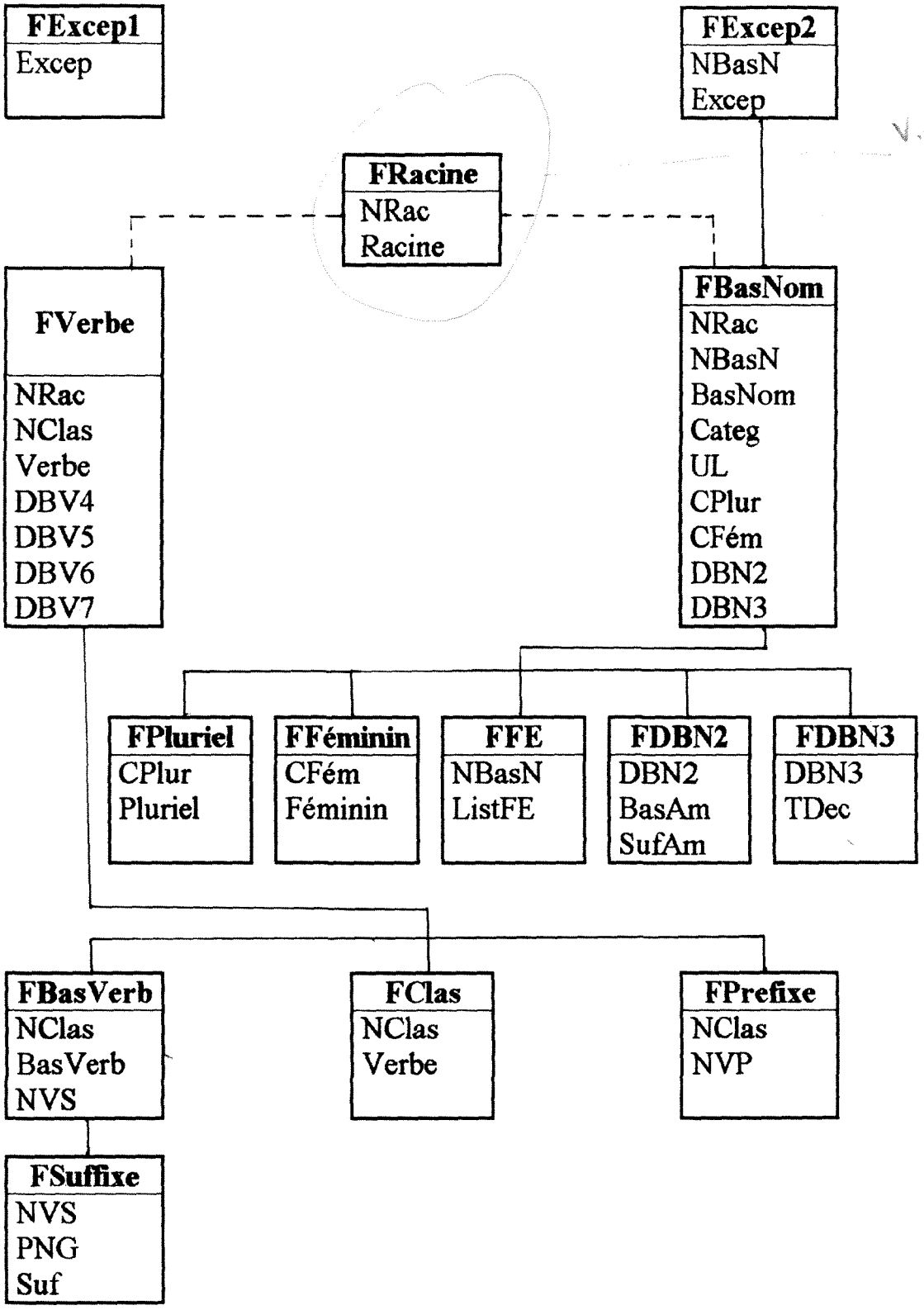
- Un ensemble de relations entre bases corrélées d'un point de vue morphologique telles que singulier-pluriel, masculin-feminin, etc.

- Un ensemble de traits morphosyntaxique associés aux bases et validant ou invalidant des relations entre les bases nominales ou verbale et les autres morphèmes du mot (préfixes ou suffixes, proclitiques ou enclitiques).

Ce dictionnaire, implémenté sous forme d'une base de données relationnelle, contient dans sa première version une quinzaine d'entités ou fichiers repartis suivant deux branches principales : une branche verbales et une branche nominale.

Nous emprunterons une brève présentation de ce dictionnaire aux travaux de M. GHENIMA [Ghénima 92], auxquels nous renvoyons pour plus de détails.

Les relations entre les différentes entités du dictionnaire peuvent être schématisées comme suit :



Contenu et organisation du dictionnaire

V.2.1. FExcep1 : Fichier des Exceptions 1

Ce fichier regroupe des mots outils (champ Excep) composés de clitiques, de prépositions et de pronoms affixes. Exemples : « لـه » (il a), « فـي » (dans), « كـي » (dans le but de), etc.

Ce sont des mots qui n'ont pas de descripteurs et dont l'utilisation dans l'écriture des textes arabes est assez fréquente. Leur regroupement dans le fichier exception 1 permet d'accélérer le processus de vérification.

V.2.2 FExcep2 : Fichier des Exceptions 2

Ce fichier contient des mots (champ Excep) non assimilés par le système morphologique dérivationnel de l'arabe tels que les emprunts : « كمبيوتر » (computer), « تكنولوجيا » (technologie), ou les noms propres « تونس » (Tunisie), « فرنسا » (France), etc.

On trouve aussi dans ce fichier exception 2 des mots tels que « سفرجل » (coing) ou « بوش » (Bush) dont la base n'est par représentable en schème et en racine mais qui peuvent avoir des descripteurs linguistiques. C'est le cas du mot « سفرجل » dont le pluriel est « سفارج ». Le champ NBasN (Numéro de la Base Nominale) permet de rattacher ces mots à leurs descripteurs.

V.2.3 FRacine : Fichier Racine

Ce fichier est considéré comme une méta-entrée du dictionnaire, il contient les racines des bases verbales ou nominales des mots pour lesquels une représentation par un schème et une racine est possible.

Il faut noter qu'un élément de ce fichier peut être la racine de plusieurs bases verbales et nominales. C'est d'ailleurs pour cette raison que le champ NRac (Numéro de Racine) figure dans le fichier des verbes et le fichier des bases nominales.

V.2.4 FVerbe : Fichier Verbe

Ce fichier contient tous les verbes du dictionnaire ainsi que leurs descripteurs respectifs. Son exploration ainsi que les autres fichier qui lui sont rattachés (FBasVerb : Fichier des bases Verbales, FClas : Fichier des Classes, FPrefixe : Fichier des préfixes, FSuffixe : Fichier des suffixes) permet en synthèse d'avoir la conjugaison de tous les verbes dans tous les aspects et modes.

De même, la présence des descripteurs DBV4, DBV5, DBV6 et DBV7 [Dichy 90] permet respectivement :

- de savoir si le verbe est transitif, transitif non-humain ou non-transitif pour décider de sa compatibilité ou de son incompatibilité avec certains pronoms enclitiques (DBV4).
- de savoir si le verbe peut avoir un complément d'objet de la première ou de la deuxième personne si son sujet est de la même personne (DBV5).
- de savoir si le verbe admet ou pas un double complément d'objet (le premier étant toujours humain) (DBV6).
- de déterminer le nom verbal (« مصدر ») pour les verbes qui en possèdent un (DBV7).

Les trois premiers descripteurs cités permettent en particulier, dans un contexte de correction orthographique, de "gérer" la relation entre la base verbale et les éléments pré- ou siffixés.

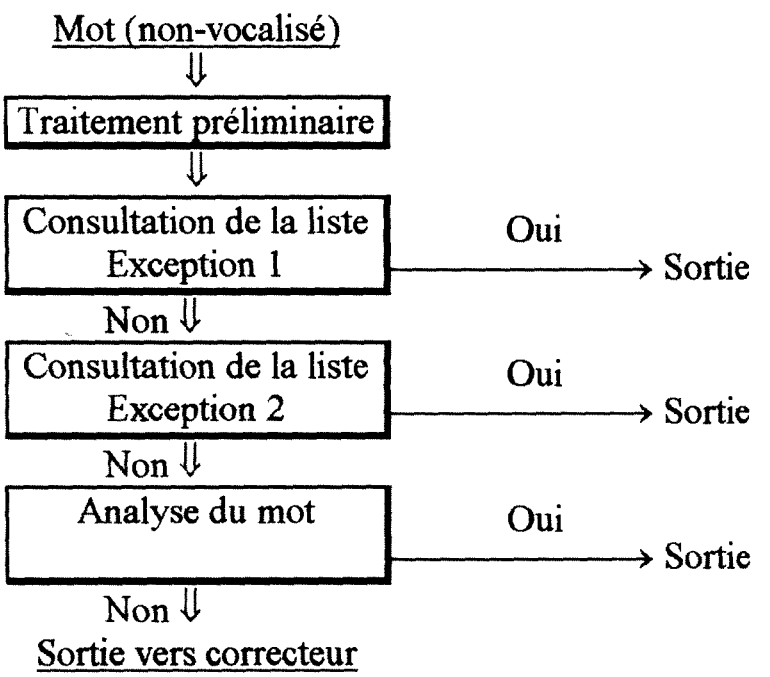
V.2.5 FBasNom : Fichier des Bases Nominales

C'est le fichier des bases verbales du dictionnaire. Il permet en synthèse d'avoir toutes les formes nominales possibles pour chacune de ses bases.

Pour cela, le synthétiseur utilisera tous les descripteurs nominaux du fichier des bases nominales ainsi que les fichiers qui lui sont rattachés (FPluriel : Fichier du Pluriel, FFéminin : Fichier du Féminin, FFE : Fichier Formant Extension, FDBN2 et FDBN3 : Fichier Descripteur Base Nominale).

V.3. LES DIFFERENTES ETAPES DE LA VERIFICATION

L'entrée du vérificateur orthographique est un mot écrit en graphie arabe non-vocalisée. Le but de la vérification est de dire si ce mot tel qu'il est écrit figure ou non dans le dictionnaire. Pour cela, le vérificateur procède par plusieurs étapes que l'on peut schématiser comme suit :



V.3.1. Le traitement préliminaire :

Certaines lettres de l'alphabet arabe peuvent avoir plusieurs écritures possibles dans le mot selon qu'elles se trouvent au début, au milieu ou à la fin de celui-ci. Ainsi, la lettre « ع » dans les mots « عنب » (raisin), « معلّم » (enseignant), « ربيع » (printemps) et « باع » (il a vendu), s'écrit respectivement « ـع », « ـعـ », « عـ » et « ع ».

Pour les données du dictionnaire, une seule représentation est retenue pour chaque lettre de l'alphabet. Ainsi la lettre « ع » sera toujours représentée par le caractère « ـع » indépendamment de sa position dans le mot.

Le traitement préliminaire permet donc de remplacer chaque caractère du mot en entrée du système par le caractère retenu dans la représentation des données du dictionnaire (un traitement inverse permet de retrouver la bonne graphie du mot pour son édition).

La présence d'un caractère spécial dans l'écriture du mot en arabe justifie aussi le recours à ce traitement préliminaire. En effet, le mot « باع » peut aussi être écrit « بـــاع » en utilisant le caractère de prolongement « ـ » (« ـّـ ») pour rendre l'écriture plus lisible et plus jolie. Ce traitement aura donc pour but de compacter le mot pour ne garder que les lettres de l'alphabet.

V.3.2. Consultation de la liste Exception 1 :

La consultation de cette liste est une étape facile à franchir car elle consiste à voir si le mot à vérifier figure parmi ses mots ou pas. Le nombre des mots de cette liste étant fini et pas très grand, on peut envisager de la charger en mémoire au début du traitement pour accélérer la consultation.

Cette liste, une fois dressée et bien définie, peut nous donner la taille du plus long mot la constituant. Ainsi un test sur la longueur du mot à vérifier peut nous éviter une recherche inutile (si la taille de ce dernier est supérieure à la taille maximale).

Le succès de cette consultation met fin à la vérification (le mot appartient au vocabulaire).

V.3.3. Consultation de la liste Exception 2 :

La consultation de cette liste se déroule de la même manière que la consultation précédente (du moins la première fois car cette liste sera sollicitée une seconde fois dans l'étape suivante). Par contre, la taille de cette liste étant indéterminée, son chargement en mémoire nous parait difficile à envisager.

De même, le succès de cette consultation met fin à la vérification (le mot appartient aussi au vocabulaire).

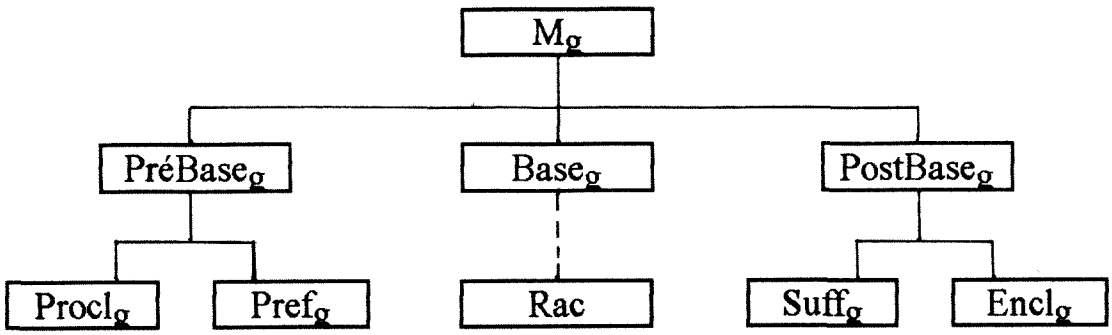
V.3.4. Analyse du mot :

Cette étape de la vérification est basée sur un modèle d'analyse du mot graphique qui est celui du programme de recherche SAMIA présenté dans la thèse de M. HASSOUN [Hassoun 87], et dans le premier volume des Travaux SAMIA par J. DICHY [SAMIA 89].

Le mot graphique en arabe (M_g*) pouvant comprendre plusieurs morphèmes, l'analyse aura pour tache d'en identifier ses constituants en de

* La lettre g en indice veut dire non-vicalisé.

décomposant en : proclitique(s), préfixe, base, suffixe(s) et enclitique(s) selon le schéma suivant :



La décomposition du mot en prébase, base et postbase peut donner lieu à plusieurs combinaisons de ces derniers. Nous verrons dans ce qui suit les différentes étapes de l'analyse qui vont permettre de retrouver celle qui assurera le succès de la vérification.

Etape 1 : Décomposition du mot

Pour faciliter le déroulement de cette étape, nous disposons de la liste des combinaisons possibles entre proclitiques et préfixes (voir liste des prébases en annexe) et celle des combinaisons possibles entre suffixes et enclitiques (voir liste des postbases en annexe).

La comparaison de chaque élément de ces listes respectivement avec le début et la fin de mot à vérifier nous permet d'obtenir différentes combinaisons de prébases ($Procl_g$ + $Pref_g$) et de postbases ($Suff_g$ + $Encl_g$). Seules seront retenues les combinaisons dont la prébase et la postbase sont compatibles (voir tableau de compatibilité en annexe).

A la fin de cette étape nous aurons donc la liste des combinaisons candidates formée de quintuplés ($Procl_g$, $Préf_g$, $Base_g$, $Suff_g$, $Encl_g$).

La suite de la vérification consiste à prendre ces quintuplés un à un pour essayer de retrouver $Base_g$ dans le dictionnaire et ainsi valider sa compatibilité avec les autres constituants identifiés dans la décomposition.

Etape 2 : Consultation de la liste Exception 2

Contrairement à la première consultation de cette liste, ce deuxième accès ne se fait pas avec le mot M_g complet mais uniquement avec sa base présumée $Base_g$. Si cette base existe (Excep = $Base_g$), un test sur le champ NBasN permet de savoir si des descripteurs linguistiques lui sont associés dans le dictionnaire (NBasN <> 0). Si c'est le cas, on peut savoir à l'aide du dictionnaire si $Pref_g$ et $suff_g$, associés à $Base_g$ dans le mot lui sont compatibles ou non.

Si la base n'a pas de descripteurs linguistiques (NbasN = 0), elle ne doit avoir ni préfixes ni suffixes, ce qui veut dire que $Pref_g$ et $Suff_g$ doivent être vides pour que $Base_g$ soit acceptée.

Si ces différents tests réussissent, le mot est correct et on arrête la vérification sinon, on passe à l'étape suivante.

Etape 3 : Détermination des racines candidates

Le recours à la détermination des racines candidates permet de restreindre la recherche. En effet, si on ne considère que les bases nominales ou les verbes dont la racine peut être celle de $Base_g$, plusieurs recherches inutiles seront évitées.

Pour cela, il faut faire attention à certaines bases dans lesquelles, une consonne ou une partie des consonnes de la racine est présentée (cas de « يرى » (il le regarde) dont la racine est « رأى »).

Pour résoudre ce problème, une racine est retenue comme candidate si elle contient au moins une lettre de $Base_g$. Un classement préalable de la liste des racines candidates, permettra d'utiliser en premier lieu celle qui contient le plus grand nombre de lettres de $Base_g$

<u>Etape 4 :</u> Consultation de la base nominale

L'étude de différents corpus de textes arabes montre que les formes nominales sont plus fréquentes que les formes verbales. C'est pour cette raison qu'on a choisi de consulter la base nominale en premier car on a plus de chance de retrouver notre base dans celle-ci plutôt que dans la base verbale.

La consultation consiste à filtrer la liste de toutes les bases nominales dont la racine est égale à la première racine candidate. Pour la première base nominale de cette liste, on synthétise toutes les formes nominales possibles (voir algorithme de synthèse dans les travaux de M. GHENIMA) et on compare chacune d'elles avec la partie de M_g formée de $Pref_g$+$Base_g$+$Suff_g$. Si on trouve une égalité, la vérification s'arrête et M_g et correct, sinon on passe à la base nominale suivante.

Quand toutes les bases nominales de la liste filtrée sont épuisées sans succès, on prend la racine candidate suivante et on refait le même travail.

Etape 5 : Consultation de la base verbale

Comme l'étape précédente, cette consultation commence par le filtrage de la liste des verbes dont la racine est égale à la première racine candidate. Par contre avant de procéder à la synthèse de toutes les formes verbales de chaque verbe de la liste, celle-ci peut être réduite à l'aide d'un test permettant de confronter les descripteurs DBV4, DBV5 et DBV6 au clitiques $Procl_g$ et $Encl_g$ du mot M_g (cf. § V.2).

Une fois la liste réduite, on procède à la synthèse des formes verbales du premier verbe de la liste restante. Chaque forme est comparée à la partie de M_g constituée de $Pref_g$ + $Base_g$ + $Suff_g$ jusqu'à ce qu'on trouve une égalité. Si c'est le cas, la vérification s'arrête avec succès sinon on passe au verbe suivant jusqu'à l'épuisement de tous les verbes de la liste.

Si aucun verbe de la liste ne permet d'identifier M_g, on reprend cette étape dès le début avec la racine candidate suivante.

Quand la liste des racines candidates est épuisée sans succès, on reprend la recherche à l'étape 2 avec la décomposition suivante du M_g c'est à dire un nouveau quintuplé ($Procl_g$, $Préf_g$, $Base_g$, $Suff_g$, $Encl_g$) jusqu'à épuisement de toute les décompositions.

A ce niveau de la vérification, si aucun résultat n'est obtenu, la vérification s'arrête et on passe à la correction avec un mot M_g présumé inconnu ou mal orthographié.

V.4. CONCLUSION

On a vu dans les deux dernières étapes de l'analyse du mot qu'on fait appel aux synthétiseurs des formes nominales et verbales objets des travaux de M. GHENIMA. L'intégration de ces synthétiseurs dans le processus de vérification pose un problème de temps de réponse qui ne peut pas être négligé.

En effet, il serait préférable d'interrompre le processus de synthèse si le mot recherché figure dans les formes déjà synthétisées. Pour cela, des modifications de ces synthétiseurs sont envisagées pour leur permettre de s'arrêter soit quand le mot recherché figure dans les formes déjà synthétisées, soit quand on s'aperçoit que la continuation ne mènera pas au mot recherché.

VI. *LA CORRECTION ORTHOGRAPHIQUE DE L'ARABE*

VI.1. INTRODUCTION

Le but de la correction est d'essayer d'identifier dans le dictionnaire, le ou les mots susceptibles de corriger la chaîne erronée M_g.

Nous disposions à la sortie de la vérification, du mot M_g ainsi que de la liste de ces décompositions possibles en proclitique(s), préfixe, base, suffixe(s) et enclitique(s).

Pour éviter les répétitions dans les paragraphes qui suivent, nous considérerons que M_g est formé d'une prébase ($Prébase_g$) suivie d'une base ($Base_g$) et d'une postbase ($Postbase_g$) avec $Prébase_g = Procl_g + Préf_g$ et $Postbase_g = Suff_g + Encl_g$. Ainsi nous ne parlerons plus de quintuplés mais de triplets ($Prébase_g$, $base_g$, $Postbase_g$).

Notre approche de la correction du mot M_g consiste à dire qu'il contient peut être des erreurs dans ses constituants, erreurs ayant empêché sa bonne décomposition et par suite l'échec de la vérification.

Nous verrons dans ce qui suit comment on essayera de corriger chaque constituant du mot à part, pour ensuite confronter les résultats obtenus et proposer des chaînes candidates à la correction de M_g.

Ces différentes chaînes candidates seront classées de telle sorte que la première soit la correction la plus plausible de M_g.

VI.2. PRINCIPE DE LA CORRECTION

La correction de M_g ($Prébase_g$, $Base_g$, $Postbase_g$) se déroule sur trois étapes principales. La première permet de corriger M_g quand il ne contient ni prébase ni postbase ($M_g = Base_g$). La seconde étape permet de corriger la prébase et la postbase. La troisième et dernière étape corrige la base.

VI.2.1. Correction de M_g :

La correction de M_g quand ce dernier est égal à $Base_g$ est prise à part dans notre approche de la correction car elle ne nécessite, à première vue, qu'un accès à une partie du dictionnaire qui est la liste Exception 1. Cet accès est conditionné par la taille de M_g. En effet si cette taille est supérieure de plus d'un caractère à la taille du plus long mot de cette liste alors cette étape est sautée, sinon la correction consiste à parcourir cette liste pour essayer de trouver les mots qui peuvent corriger une faute d'omission, d'insertion, de permutation ou de substitution commise dans l'écriture de M_g. Chaque mot trouvé est retenue comme une correction possible de M_g.

VI.2.2. Correction de $Prébase_g$ et $Postbase_g$:

A partir du mot M_g on essaye de voir si la correction d'une erreur d'omission, d'insertion, de substitution ou de permutation sur ses caractères de début permet de reconnaître une prébase ($Prébase_p$) de la liste des prébases possibles (voir liste en annexe). Pour cela il est plus facile de procéder inversement, c'est à dire, prendre les prébases de la liste et pour chaque $Prébase_p$ de taille *n* (n inférieur à la taille de M_g) vérifier les conditions suivantes :

- si les *n* - 1 premiers caractères de M_g figurent dans $Prébase_p$ avec le même ordre d'apparition alors $Prébase_p$ est retenue comme correction d'une erreur d'omission dans la prébase de M_g.

- si les *n* caractère de $Prébase_p$ figurent dans les *n* + 1 premiers caractères de M_g et avec le même ordre alors $Prébase_p$ est retenue comme correction d'une erreur d'insertion dans la prébase de M_g.

- si les *n* caractères de $Prébase_p$ figurent dans les *n* premiers caractères de M_g avec une seule et unique permutation de deux caractères voisins alors $Prébase_p$ est retenue comme correction d'une erreur de permutation dans la prébase de M_g.

- si *n* - 1 caractères de $Prébase_p$ figurent dans les *n* premiers caractères de M_g et avec le même ordre alors $Prébase_p$ est retenue comme correction d'une erreur de substitution dans la prébase de M_g.

Quand toute la liste des $Prébases_p$ est épuisée, nous disposerons d'une liste formée de deux éléments, le premier étant la $Prébase_p$ retenue, le second étant le type de l'erreur qu'elle corrige (O : omission, I : insertion, P : permutation et S : substitution) ce qui nous permettra par la suite de reconstituer $Base_g$.

On procède de la même manière sur les derniers caractères de M_g mais pour dresser cette fois la liste des $Postbases_p$ qui peuvent corriger des fautes d'omission, d'insertion, de permutation ou de substitution commises dans la postbase de M_g. Ainsi nous disposerons d'une seconde liste identique à la première du point de vue structure mais contenant des postbases.

Avec ces deux listes, on essaye de dresser une liste de triplets ($Prébase_g$, $Base_g$, $Postbase_g$) dans lesquels :

- $Prébase_g$ et $Postbase_g$ proviennent respectivement de la première et de la deuxième liste dressées auparavant et ayant satisfait au critères de compatibilité entre prébases et postbases (voir liste de compatibilité en annexe).

- $Base_g$ est ce qui reste de M_g après élimination des caractères de début et de fin corrigés respectivement par $Prébase_g$ et $Postbase_g$ (le triplet est éliminé s'il y a chevauchement entre la prébase et la postbase).

Chaque fois qu'un triplet est constitué, on vérifie qu'il ne s'agit pas d'une combinaison ayant servit à la vérification. Si c'est le cas, le triplet est rejeté sinon il fera l'objet d'une vérification dans le dictionnaire.

Si la vérification se termine par un succès, on procède à la concaténation des différents composants du triplet pour former un mot qui sera retenu comme une correction possible de M_g et on passe au triplet suivant.

Il est important de signaler qu'à ce niveau de la correction, le vérification des triplets n'est pas pareille à la vérification de M_g étudiée dans le paragraphe V, bien qu'il s'agisse du même vérificateur. En effet, quand il est appelé par le correcteur, le vérificateur saute ses deux premières étapes et commence directement par la consultation de la liste Exception 2.

VI.2.3. Correction de $Base_g$:

Pour la correction de $Base_g$, on dispose de la liste des triplets résultant de la correction de $Prébase_g$ et de $Postbase_g$ et n'ayant pas satisfait à la vérification. A cette liste on ajoute la liste des triplets obtenus par l'étape de décomposition dans le processus de vérification de M_g (vérification qui s'est soldée par un échec).

Pour chaque triplet de cette liste, on essaye de trouver dans le dictionnaire un mot pouvant corriger une erreur d'omission, d'insertion, de permutation ou de substitution commise dans $Base_g$ (la liste Exception 1 n'est pas consultée du fait que ses mot n'acceptent ni prébases ni postbases).

Pour chaque mot trouvé on vérifie sa compatibilité avec la $Prébase_g$ et la $Postbase_g$ du triplet. S'il est compatible on procède à sa concaténation avec ces derniers pour former un mot qui sera retenu comme une correction possible de M_g. On passe ensuite au triplet suivant jusqu'à la fin de la liste.

V.3. CLASSIFICATION DES SOLUTIONS CANDIDATES

Le but de cette étape est de classer les chaînes candidates à la correction de façon à ce que la plus plausible soit en premier.

Pour cela nous utiliserons le coefficient de similarité introduit par Angell [Angell 83]. La chaîne candidate ayant le plus fort coefficient de similarité avec M_g sera classée la première.

Le calcul du coefficient de similarité est basé sur la décomposition en trigrammes du mot erroné M_g (de taille n) et du mot candidat à la correction M_c (de taille n'). Cette décomposition se fait de la façon suivante :

Deux caractères blancs sont insérés au début et à la fin de M_c pour s'assurer que chaque caractère figure exactement dans 3 trigrammes. Le mot M_c peut ainsi être représenté par le vecteur $(c_1, c_2, ..., c_{n'+2})$ où chaque c_i est une suite de trois caractères.

Le mot M_g est aussi décrit par un vecteur similaire $(g_1, g_2, ..., g_{n+2})$.

Si k est le nombre des trigrammes communs à M_g et M_c ($c_i = g_j$) alors une mesure de similarité S_c entre M_g et M_c est donnée par la formule :

$$S_c = \frac{2 * k}{n + n' + 4}$$

VI.3. CONCLUSION

L'approche que nous avons adoptée dans la correction des mots erronés permet de corriger au plus trois erreurs par mot, la première affectant la prébase, la seconde la base et la troisième la postbase.

Par contre, pour chaque constituant du mot, un seul type d'erreur est pris en compte. Une amélioration de ce correcteur consiste à prévoir la correction des combinaisons d'erreurs ainsi que l'omission et l'insertion de blancs (coupure et soudure de mots).

VII. *REALISATION INFORMATIQUE*

VII.1. INTRODUCTION

Ce travail a été effectué sur micro-ordinateur PC MicroSpot ayant les caractéristiques suivantes :

- Microprocesseur : 386 SX à 16 Méga Hertz (MHZ).
- Mémoire Vive (RAM) : 2 Méga Octets (MO).
- Capacité Disque Dur : 40 Méga Octets (MO).
- Système d'Exploitation : DOS Version 5.0.

Nous avons opté pour la version 4.0 de turbo-Pascal comme langage de programmation, pour plusieurs raisons. En effet, notre méthode de correction alterne la manipulation de listes, de tableaux, de fichiers et de chaînes de caractères, il était donc nécessaire d'utiliser un langage de programmation ayant une bibliothèque de procédures et de fonctions capables d'assurer la manipulation de toutes ces structures de données.

Notre choix a été aussi motivé par la riche bibliothèque, de procédures et de fonctions graphiques, disponible avec cette version qui, nous a donné la possibilité d'élaborer une interface d'entrée/sortie en caractères arabes. En effet, des raisons de portabilité et de coût excluaient l'utilisation du MS-DOS arabisé.

D'autre part, la puissance expressive de ce langage constitue un atout supplémentaire pour la réalisation d'un prototype de correcteur orthographique.

VII.2. L'INTERFACE D'ENTREE/SORTIE

Cette interface a nécessité en premier lieu, le dessin des caractères arabes dans différents formats pour la constitution des fichiers fontes, et ensuite, l'écriture de différentes fonctions et procédures pour remplacer celles de Turbo-Pascal relatives aux entrées/sorties en caractères latins.

Notre premier objectif était d'avoir une interface compatible avec toutes les cartes vidéo disponibles (**CGA**, **EGA**, **VGA**, **HERCULES**, etc.), et permettant dans tous les cas d'avoir un écran de 25 lignes sur 80 colonnes.

Il est important de signaler que, l'utilisation de cette interface ne nous fait pas perdre la possibilité d'affichage des caractères latins.

Pour avoir plus de détails sur les procédures et les fonctions de cette interface voir [Gader 90] et [Ghénima 90].

VII.3. LE PROGRAMME PRINCIPAL

VII.3.1. Algorithme :

```
Début Programme
  Lire(TEXTE_A_CORRIGER)
  Tant que non fin de TEXTE_A_CORRIGER faire
    Former(Mg)
    Correct ← VERIFIER(Mg, Liste_Décompositions)
    Si Correct
     alors
      Ecrire(TEXTE_CORRIGE, Mg)
     sinon
      Afficher('Mot incorrect ou mal orthographié : ', Mg)
      CORRIGER(Mg, Liste_Décompositions, Sol_Candidates)
      Si non Vide(Sol_Candidates)
       alors
        Afficher('Suggestions : ')
        Afficher(Sol_Candidates)
        Mot_Correct ← Sélectionner(Sol_Candidates)
       sinon
        Mot_Correct ← Mg
       fin Si
      Mot_Correct ← Saisir(Mot_Correct)
      Ecrire(TEXTE_CORRIGE, Mot_Correct)
     fin Si
  fin Tant que
fin Programme
```

VII.3.2. Commentaires :

Le programme que nous proposons ici accepte en entrée un texte écrit en graphie arabe non-vocalisée et renvoi en sortie une copie corrigée de ce même texte.

Le programme commence par former un mot. Le mot étant une suite de caractères délimitée par le caractère blanc (espace) ou un séparateur similaire (".", ",", "?", etc.).

Pour chaque mot formé on fait appel au vérificateur qui nous indiquera si le mot figure dans le dictionnaire ou pas. Le vérificateur renvoi aussi la liste des décompositions possibles du mot en prébase, base et postbase.

S'il y a échec de la vérification, le mot ainsi que sa liste de décompositions sont fournis au correcteur qui nous fournira si possible une liste de solutions candidates à la correction du mot supposé incorrect ou mal orthographié.

Cette liste, si elle n'est pas vide, est présentée à l'utilisateur pour qu'il choisisse la bonne correction. Ce dernier pourra sélectionner la bonne correction dans la liste, l'introduire lui même si elle n'y figure pas ou indiquer que le mot est bien correct ce qui voudra dire que sa non reconnaissanse par le vérificateur est due à son absence du dictionnaire.

Le mot correct obtenu après l'intervention de l'utilisateur sera écrit dans le texte résultat et le programme passe à la formation du mot suivant jusqu'à la fin du texte en entrée.

VII.4. LE VERIFICATEUR ORTHOGRAPHIQUE

VII.4.1. Algorithme de vérification :

```
Fonction VERIFIER(Mg, Liste_Décompositions) : Booléen
Début Vérifier
  Mg ← Traitement_Préliminaire (Mg)
  Trouve ← Consultation(FExcep1, Mg)
  Si non Trouve
   alors
    Trouve ← Consultation(FExcep2, Mg)
    Si non Trouve
     alors
      Trouve ← Analyser(Mg, Liste_Décompositions)
     fin Si
   fin Si
  VERIFIER ← Trouve
fin Vérifier
```

```
Fonction Consultation(FExcep, Mot) : Booléen
Début Consultation
  Trouve ← faux
  Tant que non fin de FExcep et non Trouve faire
    Lire(FExcep, Art_FExcep)
    Trouve ← (Art_FExcep.Excep = Mot)
  fin Tant que
  Consultation ← Trouve
fin Consultation
```

```
Fonction Analyser(Mg, Liste_Décompositions) : Booléen
Début Analyser
  Décomposer(Mg, Liste_Décompositions)
  Trouve ← faux
  Tant que non fin de Liste_Décompositions et non Trouve faire
    Lire(Liste_Décompositions, Procl_g, Pref_g, Base_g, Suff_g, Encl_g)
    Trouve ← Recherche_Dico( Procl_g, Pref_g, Base_g, Suff_g, Encl_g)
  fin Tant que
  Analyser ← Trouve
fin Analyser
```

```
Fonction Recherche_Dico(Procl, Préf, Base, Suff, Encl) : Booléen
Début Recherche_Dico
  Si Consultation(FExcep2, Base)
   alors
    Recherche_Dico ← Compatible_N(Art_FExcep2.NBasN, Préf, Suff)
   sinon
    Racines_Candidates(FRacine, Base, List_NRac)
    Si Rech_FBasNom(List_NRac, Préf, Base, Suff)
     alors
      Recherche_Dico ← vrai
     sinon
      Trouve ← Rech_FVerbe(List_NRac, Procl, Préf, Base, Suff, Encl)
     fin Si
   fin Si
fin Recherche_Dico
```

```
Fonction Rech_FBasNom(List_NRac, Préf, Base, Suff)
Début Rech_FbasNom
  Trouve ← faux
  Tant que non fin List_NRac et non Trouve faire
    Lire(List_Nrac, Numéro)
    List_Bas_Nom ← Sélection(FBasNom, Numéro)
    Tant que non fin de List_Bas_Nom et non Trouve faire
      Lire(List_Bas_Nom, Art_FBasNom)
      List_Form_Nom ← Synthèse_Nom(Art_FBasNom)
      Trouve ← Recherche(List_Form_Nom, Préf + Base + Suff)
    fin Tant que
  fin Tant que
fin Rech_FBasNom
Fonction Rech_FVerbe(List_NRac, Procl, Préf, Base, Suff, Encl)
Début Rech_FVerbe
  Trouve ← faux
  Tant que non fin List_NRac et non Trouve faire
    Lire(Liste_Nrac, Numéro)
    Liste_Verbes ← Sélection(FVerbe, Numéro )
    Tant que non fin de Liste_Verbes et non Trouve faire
      Lire(Liste_Verbes, Art_FVerbe)
      Si Compatible_V(Art_FVerbe, Procl, Encl)
       alors
        List_Form_Verb ← Synthèse_Verb(Art_FVerbe)
        Trouve ← Recherche(List_Form_Verb, Préf + Base + Suff)
       fin Si
    fin Tant que
  fin Tant que
fin Rech_FVerbe
```

VII.4.2. Commentaires :

La vérification d'un mot commence par le traitement préliminaire qui a pour but de compacter le mot au maximum et de remplacer chacun de ces caractères par la graphie de ce même caractère retenue pour la représentation des données dans le dictionnaire.

La consultation des fichiers exceptions FExcep1 et FExcep2 se fait par comparaison de leurs mots avec le mot obtenu après le traitement préliminaire. Si ce mot figure dans l'un des deux fichiers, la vérification s'arrête et la fonction VERIFIER prend la valeur vrai sinon elle se poursuit par l'appel de la fonction Analyser à qui on fournit le mot à analyser est qui renvoi la liste des décompositions possibles du mot ainsi qu'une valeur logique vrai ou faux suivant le succès ou l'échec de l'analyse.

L'analyseur commence par dresser la liste des décompositions possibles du mot en proclitique(s), préfixe, base, suffixe(s) et enclitique(s), ceci et fait grace aux listes de leurs combinaisons possibles ainsi que leurs listes de compatibilité dont nous disposons en mémoire.

La suite de l'analyse consiste à essayer de valider l'une des décompositions ainsi obtenues.

La fonction **Compatible_N** permet de vérifier la compatibilité entre le mot trouvé dans le fichier FExcep2 et le préfixe et le suffixe de la décomposition en cours de vérification. Ceci n'est fait que quand le mot trouvé possède un pointeur vers le fichier des bases nominales.

La procédure **Racines_Candidates** permet à partir de FRacine de retenir une liste (List_NRac) des numéros des racines, dont chacune contient au moins une consonne radicale de la base (Base) de la décomposition en cours de vérification. Cette liste sert à minimiser le temps de recherche dans le fichier des bases nominales et celui des verbes.

La fonction **Sélection** permet d'extraire une liste de bases nominales à partir du fichier FBasNom pour lesquels le numéro de racine (NRac) est égal à Numéro. Pour le fichier FVerbe, cette même fonction donne la liste des verbes dont le numéro de racine est égal à Numéro.

La fonction **Compatible_V** permet de ne pas traiter les verbes dont les descripteurs DBV4, DBV5 ou DBV6 ne sont pas compatibles avec le proclitique et l'enclitique (Procl et Encl) de la décomposition en cours de vérification.

Pour une base nominale donnée, la fonction **Synthèse_Nom** donne la liste de toutes les formes nominales de celle-ci. Cette synthèse est faite moyennant les fichier rattachés au fichier FBasNom.

Quant à la fonction **Synthèse_Verb** elle permet d'avoir toutes les formes verbales d'un verbe donné par l'utilisation des fichiers rattachés au fichier FVerbe.

Pour avoir plus de détails sur le principe et les algorithmes de ces synthétiseurs nominaux et verbaux nous vous renvoyons aux travaux de M. GHENIMA [Ghénima 92].

VII.5. LE CORRECTEUR ORTHOGRAPHIQUE

VII.5.1. Algorithme de correction :

Module CORRIGER (M_g, Liste_Décompositions, Sol_Candidates)
Début Corriger
  Si Taille (M_g) <= Taille (Plus_Long_Mot_De_FExcep1) + 1
   alors
    Correction_M_g(M_g, Sol_Candidates)
  fin Si
  Détermine_Prébases_Corrigées(M_g, ListeA)
  Détermine_Postbases_Corrigées(M_g, ListeB)
  Combiner(M_g, ListeA, ListeB, Liste_Décompos_Corrigées)
  Tant que non fin de Liste_Décompos_Corrigées faire
    Lire(Liste_Décompos_Corrigées, $Procl_c$, $Préf_c$, $Base_c$, $Suff_c$, $Encl_c$)
    Si Recherche_Dico($Procl_c$, $Préf_c$, $Base_c$, $Suff_c$, $Encl_c$)
     alors
      Ajouter(Sol_Candidates, $Procl_c$+$Préf_c$+$Base_c$+$Suff_c$+$Encl_c$)
    fin Si
  fin Tant que
  Concat_Listes(Liste_Décompositions, Liste_Décompos_Corrigées)
  Tant que non fin de Liste_Décompositions faire
    Lire(Liste_Décompositions, $Procl_g$, $Préf_g$, $Base_g$, $Suff_g$, $Encl_g$)
    Corriger_$Base_g$($Procl_g$, $Préf_g$, $Base_g$, $Suff_g$, $Encl_g$, Sol_Candidates)
  fin Tant que
  Trier(Sol_Candidates)
fin Corriger

```
Module Correction_Mg(Mg, Sol_Candidates)
Début Correction_Mg
  n ← Taille(Mg)
  Tant que non fin de FExcep1 faire
    Retenir ← faux
    Lire(FExcep1, Art_FExcep1)
    Cas Taille(Art_FExcep1) de
      n - 1  : Retenir ← Test_Insertion(Mg, Art_FExcep1)
      n + 1 : Retenir ← Test_Omission(Mg, Art_FExcep1)
      n      : Retenir ← Test_Permutation(Mg, Art_FExcep1)
               Si non Retenir
               alors
                 Retenir ← Test_Substitution(Mg, Art_FExcep1)
               fin Si
    fin Cas
    Si Retenir
     alors
       Ajouter(Sol_Candidates, Art_FExcep1)
    fin Si
  fin Tant que
fin Correction_Mg
```

VII.5.2. Commentaires :

Le module **Correction_M_g** permet d'effectuer une tentative de correction rapide du mot erroné en supposant que ce dernier n'est pas décomposable. Pour cela il est comparé aux mots du fichier FExcep1 quand sa taille n'excède pas celle du plus long mot de ce fichier, de plus d'un caractère.

Le module **Détermine_Prébases_Corrigées** dresse la liste (ListeA) des prébases sesceptibles de corriger des erreurs d'omission, d'insertion, de substitution ou de permutation commises sur les caractères de début du mot M_g. Le même travail est fait sur les caractères de fin de M_g par le module **Détermine_Postbases_Corrigées**, dans le but de dresser la liste des postbases (ListeB).

Une fois les deux listes dressées, le module **Combiner** permet de former la liste des décompositions corrigées, dont chaque combinaison contient une prébase est une postbase compatibles issues respectivement de ListeA et de ListeB, le troisième élément étant ce qui reste de M_g après son amputation du couple prébase et postbase erronées.

Les fonctions **Test_Insertion**, **Test_Omission**, **Test_Permutation** et **Test_Substitution** acceptent comme données deux mots et renvoient comme résultat une valeur logique vrai si le deuxième mot corrige respectivement une erreur d'insertion, d'omission, de permutation ou de substitution commise dans le premier (premier argument).

Le module **Corriger_Base$_g$** permet de trouver des corrections possibles de la base du mot et de valider ces corrections avec le reste des constituants de la décomposition. Ceci est fait pas la consultation du fichier FBasNom et du fichier FVerbe. Chaque correction obtenue est ajoutée à la liste des solutions candidates.

La fonction du module **Trier** est de classer les solutions candidates en utilisant le coefficient de similarité décrit dans le paragraphe V.3.

VII.6. CONCLUSION

Nous avons essayé de décrire dans cette partie, le principe et les algorithmes de quelques procédures et fonctions principales du programme de vérification et de correction orthographique que nous avons réalisé.

Pour des raisons de brièveté et de clarté dans la description faite de ces algorithmes, des traitements complexes ont été remplacés par des appels de procédures ou de fonctions dont les algorithmes n'ont pas été détaillés. En effet, la procédure **Détermine_Prébases_Corrigées** par exemple, dont le principe est pourtant simple, nécessite le développement d'un algorithme d'une trentaine de lignes.

CONCLUSION GENERALE

Pour conclure sous forme d'un résumé mettant en évidence les points importants de notre travail, nous pouvons dire que nous avons dégagé et montré la faisabilité de l'automatisation de la vérification et de la correction des fautes dans les textes arabes non-vocalisés. Il s'agit, en fait, d'un travail pluridisciplinaire qui montre bien tout ce que peut apporter une collaboration entre informaticiens et linguistes.

A son état actuel, le système auquel nous avons abouti, permet de vérifier les textes sur le plan lexical par décomposition de ses mots en prébases, bases et postbases, et de corriger un type d'erreur par constituant, ce qui permet donc, de corriger au plus trois erreurs par mot.

Les perspectives d'avenir consistent à améliorer la version actuelle pour tenir compte des combinaisons de types d'erreurs et des cas de soudure ou de coupure de mots, qui, pour des contraintes de temps, n'ont malheureusement pas été traités.

La méthode de vérification et de correction que nous avons adopté n'exclut pas la possibilité d'avoir plusieurs lexiques. Ceci est intéressant en pratique, car il est ainsi possible d'ajouter des lexiques propres à chaque domaine d'application. Par exemple, le traitement des sigles ou des autres termes spéciaux dans la correction des requêtes d'interrogation pourra se faire au niveau du lexique de l'application qui sera mis à jour par l'utilisateur.

D'autre part, la structure et l'organisation du dictionnaire utilisé par notre système, nous permet, avec quelques légères modifications, de traiter les textes vocalisés et aussi de ne pas se contenter de trouver des mots ou formes

exacts, et des voisins en cas d'erreurs, mais de délivrer avec chaque forme trouvée le ou les codes grammaticaux correspondants, ce qui permet d'envisager des traitements supérieurs au simple niveau lexical.

En effet, si au stade actuel les fautes de nature grammaticale n'ont pas été traitées, une application visée consiste à développer autour de ce correcteur un environnement de connaissances syntaxico-sémantiques, pour passer de la vérification et la correction du mot à la vérification et la correction de la phrase.

Pour finir, nous nous excusons auprès de nos chers lecteurs de l'absence de données statistiques sur les performances de notre système de vérification et de correction. Ceci sera possible dès la fin des tests que nous sommes en train de réaliser sur des corpus de textes arabes tirées de quelques journaux tunisiens écrits dans cette langue.

ANNEXE

Résultats de l'étude typologique et statistique des erreurs orthographiques [Hamadou 87]

Cette étude a été effectuée dans un contexte de dictée. Elle a concerné une population de 1200 élèves répartis sur les classes de 1ière, 2ième et 6ième année de l'enseignement secondaire.

Tableau des fréquences moyennes et relatives

أنماط الأخطاء / درجات التواتر	ألف واو الجماعة	رسم		الخلط بين				المجموع
		الهمزة	التاء	المقصور والممدود	الظاء والصاد	الحركات الطويلة والقصيرة	القطع والوصل	
معدل التواتر	4،12	14،45	2،20	14،18	16،43	5،02	18،36	معدل التواتر : 110
التواتر النسبي	3،43	12،04	1،83	11،81	13،69	4،18	15،30	التواتر النسبي : 100
أنماط الأخطاء / درجات التواتر	حذف		الأصوات المتقاربة	تأثير الجوار الصوتي	الكتابة الاصطلاحية	الادغام	فك السلسلة الكلامية	
	اللام الشمسية	الهمزة الوصلية						
معدل التواتر	0،34	18،42	3،42	2،26	6،87	2،87	11،09	
التواتر النسبي	0،29	15،35	2،85	1،88	5،73	2،39	9،24	

Influence du niveau de connaissances

أنماط الأخطاء / السنوات	ألف واو الجماعة	رسم		الخلط بين				المجموع
		الهمزة	التاء	المقصور والممدود	الظاء والضاد	الحركات الطويلة والقصيرة	القطع والوصل	
التلميذ س 1	0،05	1،04	0،74	1،51	2،37	1،23	1،61	التلميذ س 1 : 12،77
التلميذ س 2	0،04	0،83	0،26	1،16	1،86	0،87	1،91	التلميذ س 2 : 9،85
التلميذ س 3	0،03	0،58	0،20	1،03	1،37	0،52	1،11	التلميذ س 3 : 7،45
أنماط الأخطاء / السنوات	حذف		الأصوات المتقاربة	تأثير الجوار الصوتي	الكتابة الاصطلاحية	الادغام	فك السلسلة الكلامية	
	اللام الشمسية	الهمزة الوصلية						
التلميذ س 1	0،05	0،67	1،48	0،58	0،43	0،30	0،71	
التلميذ س 2	0،03	0،51	0،91	0،30	0،44	0،22	0،51	
التلميذ س 3	0،04	0،39	0،30	0،14	0،32	0،10	0،32	

Listes des Proclitiques et Préfixes

Proclitiques
أ ، و ، ف ، ك ، ل ، س
ال ، أل ، وكال
أو ، أف ، أب ، أك
أول ، أفل
أس
وب ، وس ، ول ، وك
فل ، فب ، فك ، فس
وال ، فال ، بال ، كال
لل ، ولل ، فلل
أوب ، أفس ، أفب
فلال ، فكال ، وبال

Préfixes
أ
ن
ي
ت

Listes des Suffixes et Enclitiques

Suffixes
ت ، ة
ون
تا ، تم
و ، وا
ات ، تان
ن ، تين
ين ، تن
ا ، ان
نا
تمو ، تما
و ، يه

Enclitiques
ها
هما
ه
هم
هن
كم
كن
ي
ك
نو
نا

Liste de compatibilité Proclitiques-Préfixes

Préf. / Procl.	Vid	أ	ن	ب	ت
Vide	+	+	+	+	+
أ	+	+	+	+	+
و	+	+	+	+	+
ف	+	+	+	+	+
ب	+				
ك	+				
ل	+	+	+	+	+
س		+	+	+	+
ال	+				
أل	+	+	+	+	+
فل	+	+	+	+	+
فس		+	+	+	+
لل	+				
أوب	+				
أفس		+	+	+	+
أفب	+				
أول	+	+	+	+	+
أفل	+	+	+	+	+
ولل	+				

Préf. / Procl.	Vid	أ	ن	ب	ت
أو	+	+	+	+	+
أف	+	+	+	+	+
أب	+				
أك	+				
أس		+	+	+	+
وب	+				
وك	+				
ول	+	+	+	+	+
وس		+	+	+	+
فب	+				
فك	+				
فلل	+				
وال	+				
فال	+				
بال	+				
كال	+				
بكال	+				
وكال	+				
فبال	+				

Liste de compatibilité Suffixes-Enclitiques

Suf / Encl	Vid	جا	جما	ن	هم	جن	كم	كن	و	ك	نو	نا
Vide	+	+	+	+	+	+	+	+	+	+	+	+
ت	+	+	+	+	+	+	+	+	+	+	+	+
ون	+	+	+	+	+	+	+	+	+	+	+	+
تا	+	+	+	+	+	+	+	+	+	+	+	+
و	+	+	+	+	+	+	+	+		+	+	+
ات	+	+	+	+	+	+	+	+	+	+		+
ن	+	+	+	+	+		+	+	+	+		+
ين	+	+	+	+	+	+			+		+	+
ا	+	+	+	+	+		+	+		+	+	+
تا	+	+	+	+	+		+	+		+		
تم	+	+	+	+	+	+					+	+
و	+	+	+	+	+	+					+	+
تما	+	+	+	+	+	+					+	
تن	+	+	+	+	+	+						
ان	+	+	+	+	+	+	+	+			+	
تين	+	+	+	+	+	+						
ة	+											
يه	+											
تان	+											
وا	+											
تم	+											

Liste des Prébases

(Proclitiques + Préfixes compatibles)

(1) [ ]

(2) [أو ف]

(3) [وأ فأ]

(4) [أأ أوأ أفأ ألأ أسأ أفلأ أفسأ ولأ وسأ فلأ
فسأ أ لأ سأ]

(5) [أن أون أفن ألن أسن أفلن أفسن ون ولن وسن
فن فلن فسن ن لن سن]

(6) [أي أوي أفي ألي أسي أفلي أفسي وي ولي وسي
في فلي فسي ي لي سي]

(7) [أت أوت أفت ألت أست أفلت أفست وت ولت وست
فت فلت فست ت لت ست]

(8) [ل ول نل]

(9) [لل ولل نلل]

(10) [ال وال وكال فال فكال كال]

(11) [بال فبال]

(12) [ك وك فك]

(13) [ب وب فب]

(14) [و بكال أل أف أب أك أوب أول أولأ أولن
أولي أولس أفل]

Liste des Postbases

(Suffixes + Enclitiques compatibles)

(1) [ ]

(2) [ك كن كم ه هم ها هن هما]

(3) [نه نكن نكم نهم نها ن]

(4) [اني انا ني نني وني ونه وننني وننا ونهم ونها
ونهن ونا تنا]

(5) [ون ان انهم انها انهن انهما وا وه وهم وها
وهن وهما]

(6) [ا اه اهم اها اهما]

(7) [ي]

(8) [ين]

(9) [نك اك اكم]

(10) [وك وكم]

(11) [ونك ونكن ونكم]

(12) [ينه يننني يننا ينهم ينها ينهن ينهما يني ينا
يه يهم يها يهن يهما]

(13) [تني ة يه ات]

(14) [تان]

(15) [نا تا تاك تاه تاني تانا تاكن تاكم تاهم
تاها تاهن تاهما]

(16) [تن تنه تنهم تنها تنهن تنهما تاه تاهم تاك
تاكن تاكم تاها تاهما ة تي تك ته تني تنا]

(17) [تكن تكم تهم تهن تهما تما تماه تماني تماهم
تماهن اتها تم اته اننا اتكم اتهم تماهما]

(18) [تها وتهما وته وني وننني وننا تاو و وكن وي
اتهما اتكن اتي اتك اتني اكن تهو تهوها]

(19) [تهوه تهوهما تهوهم تهوهن تهوني تهونا انه انكم
اكن انني تينها تينهما تينه تينهم تينهن]

Liste de compatibilité Prébases-Postbases

12	11	10	9	8	7	6	5	4	3	2	1	Prébases / Postbases
+	+	+	+	+	+	+	+	+	+	+	+	1
+	+				+	+	+	+	+	+	+	2
						+	+	+		+	+	3
						+	+			+	+	4
	+		+			+	+			+	+	5
+	+					+	+			+	+	6
+	+	+	+		+	+				+	+	7
+	+	+	+	+	+	+					+	8
							+			+	+	9
+	+						+				+	10
							+				+	11
						+				+	+	12
+	+	+	+	+	+						+	13
	+		+								+	14
+	+					+					+	15
											+	16

REFERENCES BIBLIOGRAPHIQUES

[Andreewsky 78] A. ANDREEWSKY, F. DEBILI, C. FLUHR : *Une propriété remarquable du lexique des langues naturelles et son utilisation dans la correction automatique des erreurs typographiques*; Rapport du Commisariat à l'Energie Atomique, référence CEA-N-2067, décembre 1978.

[Angell 83] R. C. ANGELL, G. E. FREUND, P. WILLET : *Automatic spelling correction using a trigram similarity measure*; Information Processing Management, vol. 19, n° 4, 1983, pp. 255-261.

[Berkel 88] B. VAN BERKEL, K. DE SMEDT : *Triphone analysis : a combined method for the corrections of orthographical and typographical errors*; Second Conference on Applied Natural Language Processing (Austin), 9-12 february 1988, pp. 77-83.

[Blair 60] C. R. BLAIR : *A program for correcting spelling errors*; Information Control, vol. 3, 1960, pp. 60-67.

[Bourne 77] C. P. BOURNE : *Frequency and impact of spelling errors in bibliographic data bases*; Information Processing and Management, vol. 13, n° 1, 1977, pp. 1-12.

[Carlson 66] G. CARLSON : *Techniques for replacing characters that are garbled on input*; Proceeding of the 1966 Spring Joint Computer Conference, 1966, pp. 189-192.

[Catach 82] N. CATACH : *L'orthographe*; coll. , Presses Universitaires de France, 2e édition, 1982.

[Clémencin 88] G. CLEMENCIN : *Querying the French yellow pages : natural language access to the directory*; International Conference on the user-oriented contents-based text and image handling, RIAO88 (Cambridge MIT), march 1988.

[Damerau 64] F. J. DAMERAU : *A technique for computer detection and correction of spelling errors*; Connunication of the ACM, vol. 7, n° 3, march 1964, pp. 171-176.

[Davidson 62] L. DAVIDSON : *Retrieval of misspelled names in an airline's passenger record system*; Communication of the ACM, vol. 5, n° 3, march 62, pp. 169-171.

[Dichy 90] J. DICHY : *L'écriture dans la représentation de la langue : la lettre et le mot en arabe*; Thèse pour le doctorat d'état, Université Lyon 2, 1990.

[Fisher 76] E. G. FISHER : *The use of context in character recognition*; COINS TR 76-12, Department of Computer and Information Sciences, University of Massachussetts, Amherst, july 1976, 189 pages.

[Freeman 63] D. N. FREEMAN : *Error correction in CORC*; the Cornell Computing Language, Ph. D. Thesis, Department of Computer Science, Cornell University, september 1963.

[Frison 88] P. FRISON, D. LAVENIER : *A fast machine for prototyping string correction algorithms*; International Conference on the user-oriented contents-based test and image handling, RIAO88 (Cambridge MIT), march 1988.

[Gader 90] N. GADER : *Un système d'Enseignement Assisté par Ordinateur des formes nominales dérivées en arabe, avec simulation*; Mémoire de maîtrise, Institut Supérieur de Gestion de Tunis, C.R.T.T Lyon 2, octobre 1990.

[Ghénima 90] M. GHENIMA : *Conception d'un système d'Enseignement Assisté par Ordinateur de la conjugaison des verbes arabes avec simulation*; Mémoire de maîtrise, Institut Supérieur de Gestion de Tunis, C.R.T.T Lyon 2, octobre 1990.

[Ghénima 92] M. GHENIMA : *Conception et réalisation d'un dictionnaire informatisé de l'arabe, compatible avec les besoins de l'analyse morphologique et de la correction orthographique*; Mémoire de DEA, Ecole Nationale Supérieure des Sciences de l'Information et des Bibliothèques (ENSSIB Lyon), septembre 1992.

[Gruaz 86] C. GRUAZ : *Forme et fonction d'un composant morphémique en intelligence artificielle*; Actes du séminaire Gréco-Galf de la communication parlée CNRS : , Toulouse 1986.

[Guez 85] S. GUEZ, S. SABBAGH : *INTERIX : système d'aide à l'utilisation d'Unix*; Congrès de l'AFCET : , AFCET (Paris), mars 1985, pp.167-176.

[Guiraud 59] P. GUIRAUD : *Problèmes et méthodes de la statistique linguistique*; D. Reidel Pub. Co. Dordrecht-Holland, 1959.

[Hall 80] A. V. HALL, G. R. DOWLING : *Approximate string matching*; Computer Syrveys, vol. 12, n° 4, december 1980, pp. 381-402.

[Hamadou 87] A. BEN HAMADOU, M. CHARFI, A. DJEMAIL, Y. BORCHANI : *Contribution à l'étude quantitative des erreurs orthographiques*; Revue tunisienne des sciences de l'éducation, publiée par le Ministère de l'éducation, de l'enseignement et de la recherche scientifique, vol. 13, n° 16, 1987.

[Hamadou 89] A. BEN HAMADOU, M. CHARFI, J. FEKIH : *MOUID 1: un didacticiel multilingue d'enseignement de l'orthographe arabe*; Actes du IV colloque international de linguistique : Linguistique arabe et informatique, Tunis 9 - 12 novembre, 1989.

[Harmon 62] L. D. HARMON : *Automatic reading of cursive script*, dans : Proc. Symp. on Optical Character Recognition, Spartan Books. Washington, D.C., 1962, pp. 151-152.

[Hassoun 87] M. O. HASSOUN : *Conception d'un dictionnaire pour le traitement automatique de l'arabe dans différents contextes d'application*; Thèse pour le doctorat d'état ès sciences, Université Lyon 1, 1987.

[Heidorn 82] G. E. HEIDORN, K. JENSEN, L. A. MILLER, R. J. BYRD, M. S. CHODOROW : *The EPISTLE text-critiquing system*, IBM System Journal, vol. 21, n° 3, 1982.

[James 76] E. B. JAMES, D. P. PARTRIDGE : *Tolerance to inaccuracy in computer programs*; Computer Journal, vol. 19, n° 3, august 1976, pp. 207-212.

[Kashyap 84] R. L. KASHYAP, B. J. OOMMEN : *Spelling correction error using probabilistic methods*; dans : , ed. par S. Srihari, IEEE Computer Society Press, 1984, pp. 272-279.

[Knuth 73] D. E. KNUTH : *The art of the computer programming*; vol. 3 : Sorting and Searching, Addison-Wesley, Reading, Mass., 1973.

[Lapalme 86] G. LAPALME, D. RICHARD : *Un système de correction automatique des accords des participes passés*; TSI, vol. 5, n° 4, 1986, pp. 307-319.

[Lefévre 91] Ph. LEFEVRE, N. CAILLAUD : *Logiciel d'accès par voisinage à un dictionnaire informatique du français courant*; Electricité De France, bulletin de la direction des études et recherche, serie C : Mathématiques et Informatique, n° 1, 1991.

[Levenshtein 66] V. I. LEVENSHTEIN : Binary codes capable of correcting deletions, insertions and reversals; Sov. Phys. Dokl., vol. 10, n° 8, february 1966, pp. 707-710.

[Litecky 76] R. C. LITECKY, G. B. DAVIS : *A study of errors, error-proneness, and error diagnosis in COBOL*; Communication of the ACM, vol. 19, n° 1, january 1976, pp. 33-37.

[Lowrance 75] R. LOWRANCE, R. A. WAGNER : *An extension of the string-to-string correction problem*; J. Assoc. Comput. Mach., vol. 22, n° 2, april 1975, pp. 177-183.

[Masek 80] W. J. MASEK, M. S. PATERSON : *A faster algorithm computing string edit distances*; J. Comput. and System Sciences, vol. 20, 1980, pp. 18-31.

[McMahon 78] L. E. McMAHON, L. L. CHERRY, R. MORRIS : *Statistical text processing*; The Bell System Technical Journal; vol. 57, n° 6, part 2, july-august 1978, pp. 2137-2154.

[Morgan 70] H. L. MORGAN : *Spelling correction in systems programs*; Comm. Assoc. comput. Mach., vol. 13, n° 2, february 1970, pp. 90-94.

[Morris 75] R. MORRIS, L. L. CHERRY : *Computer detection of typographical errors*; IEEE Transactions on Professional Communications, vol. PC-18, n° 1, march 1975, pp. 54-64.

[Muth 77] F. MUTH, A. L. THARP : *Correcting human error in alphanumeric terminal input*; Information Proc. and Manage, vol. 13, n° 6, 1977, pp. 329-337.

[Okuda 76] T. OKUDA, E. TANAKA, T. KASAI : *A method for the correction of garbled words based on the Levenshtein metric*; IEEE Trans. Comput., vol. C-25, february 1976, pp. 172-177.

[Partyko 82] Z. V. PARTYKO : *Analysis of distortions arising at the input of texts in the assistent system*; Nauchno-Tekhnicheskaya Informatsiya, seria 2, n° 1, 1982, pp. 21-26.

[Pavard 85] B. PAVARD : *La conception des systèmes de traitement de textes*; Intellectica, vol. 1, n° 1, 1985, pp. 37-68.

[Pérennou 86] G. PERENNOU, F. LAHENS, P. DAUBEZE : *La vérification et la correction automatique de textes : le système VORTEX*; TSI, vol. 5, n° 4, juillet-août 1986, pp. 285-305.

[Peterson 80] J. L. PETERSON : *Computer programs for detecting and correcting spelling errors*; Comm. Ass. Comput. Mach., vol. 23, 1980, pp. 676-687.

[Peterson 80a] J. L. PETERSON : *Computer programs for spelling correction : an experiment in program design*; Lectures Notes in Computer Science, Springer-Verlag, 1980.

[Pollock 84] J. J. POLLOCK, A. ZAMORA : *Automatic spelling correction in scientific and scholarly text*; Communication of the ACM, vol. 27, n° 4, april 1984, pp. 358-368.

[Richard 86] D. RICHARD, G. LAPALME : *Un système de correction automatique des accords des participes passés*; TSI, vol. 5, n° 4, juillet-août 1986, pp. 307-319.

[Riseman 71] E. M. RISEMAN, R. W. EHRICH : *Contextual word recognition using binary digrams*; IEEE Trans. Comput., vol. C-20, n° 4, april 1971, pp. 397-403.

[Riseman 74] E. M. RISEMAN, A. R. HANSON : *A contextual postprocessing system for error correction using binary n-grams*; IEEE Trans. Comput., vol. C-23, n° 5, may 1974, pp. 480-493.

[Sabah 85] G. SABAH, A. VILNAT : *Le dialogue dans système de questions-réponses*; Congrès de l'AFCET : , AFCET (Paris), mars 1985, pp. 167-176.

[Salmina 86] N. Y. SALMINA, I. A. KHODASHINSKII : *Methods and tools of automatic spelling correction*; Automatic Documentation and Mathematical Linguistics, vol. 20, n° 5, 1986, pp. 105-112, traduction de Nauchno-Tekhnicheskaya Informatsiya, seria 2, vol. 20, n° 10, 1986, pp. 25-28.

[SAMIA 89] J. DICHY, M. O. HASSOUN : *Simulation de modèles linguistiques et Enseignement Assisté par Ordinateur de l'arabe*; Travaux SAMIA I, Paris, Fondation Postuniversitaire interculturelle (Conseil international de la langue française - CILF), 1989.

[Souilem 89] D. SOUILEM, M. TRUQUET, B. CAUSE : *Un système d'Enseignement Assisté par Ordinateur de la grammaire arabe «S.E.A.G.A»*; Actes du IV colloque international de linguistique : Linguistique arabe et informatique, Tunis 9 - 12 novembre, 1989.

[Szanzer 68] A. J. SZANZER : *Error-correcting methods in natural language processing*; Information Proc. 68, North-Holland, Amsterdam, 1968, pp. 1412-1416.

[Ullman 72] J. R. ULLMAN : *A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words*; Computer Journal, vol. 20, n° 2, 1977, pp. 141-147.

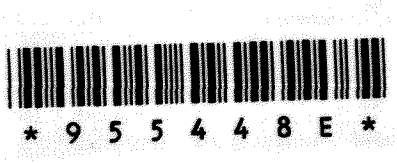
[Véronis 86] J. VERONIS : *Etude quantitative sur le système graphique et phonographique du français*; Cahier de Psychologie Cognitive, vol. 6, n° 5, 1986, pp. 501-531.

[Véronis 87] J. VERONIS : *Phonographic vs typographical correction in natural language interfaces*; Fifth International Symposium in Applied Informatics (Grindelwald Suisse), february 1987, pp. 180-183.

[Wagner 74] R. A. WAGNER, M. J. FISCHER : *The string-to-string correction problem*; J. Assoc. Comput. Mach., vol. 21, n° 1, january 1974, pp. 168-173.

[Wilensky 84] R. WILENSKY, Y. ARENS, D. CHIN : *Talking to UNIX in english : an overview of UC*; Communication of the ACM, vol. 27, n° 6, june 1984, pp. 574-593.

[Wong 76] C. K. WONG, A. K. CHANDRA : *Bounds for the string editing problem*; J. Assoc. Comput. Mach., vol. 23, n° 1, january 1976, pp. 13-16.


955448E