الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DR. YAHIA FARES DE MEDEA
FACULTE DES SCIENCES ET DE LA TECHNOLOGIE
Département du Génie Électrique
et de l'Informatique



Mémoire de Fin d'Études

Présenté pour l'obtention du Diplôme
de Master en Informatique
Option : Ingénierie des Systèmes et Technologie de Web

THEME

Adaptation de la langue arabe dans l'outil de la correction grammaticale open source LanguageTool

Proposé et dirigé par :
Mr. HADJIR ISMAIL
Co-encadré par :
Mr. ZERROUKI TAHA

Préparé et présenté par :
- Melle. KALI IMEN
- Melle. TCHOKETCH KARIMA

AnnéeUniversitaire2015-2016

Remerciement

Avant tout, nous remercions ALLAH tout puissant, pour la force, le courage et la patience qu'il nous a donnée durant toutes les années d'étude, et la volonté d'arriver au terme de ce travail.

Nous remercions infiniment nos Parents pour tout ce qu'ils ont fait et ce qu'ils font encore pour nous.

Nous exprimons notre grande reconnaissance et remerciement à nos promoteurs MR. HADJIR Ismail et MR.ZERROUKI Taha pour leur aide, leur conseils et leur orientation et suivi durant notre travail.

Nous remercions les membres de jury d'avoir accepté de juger notre travail.

Un grand merci à toutes nos familles, et nos amies pour leur présence, leur préoccupation et leurs encouragements et leur suivi, avec patience, du déroulement de notre projet.

En fin, nous ne pouvons achever ce projet sans exprimer notre Gratitudes à tous les enseignants de l'Université Dr. Yahia FARES –MEDEA–, pour leur dévouement et leur assistance tout au long de cette année.

Dédicace

Je dédie ce travail à la lumière
de ma vie : Mes chers parents, que nulle dédicace ne peut
exprimer mes sincères sentiments, pour leur patience
Illimitée, leur encouragement contenu, leur aide, en
Témoignage de mon profond amour et respect Pour ses
grands sacrifices.
A mes très chers frères : Mohamed Chérif, Othmane, Achref
Ibrahim et ma sœur Meriem Pour leur grand amour Et
leur soutien qu'ils trouvent ici L'expression de ma haute
gratitude.
A mon grand-père Allah yarhmo et ma grand-mère
Et à toute ma grande famille sans exception.
A mes chères amies et A mon binôme Karima
A toutes personnes Ayant contribué de loin ou de près
À accomplir ce travail.

K.Imen

Septembre 2016

Dédicace

Je dédie ce travail à

Mes chers parents, mon père et ma mère

Pour leur patience, leur amour, leur soutien

Et leurs encouragements.

A mes sœurs, mes frères, mes grands-mères et tous les

membres de la famille TCHOKETCH et de la famille

METIDJI.

A mon binôme Imen

A tous mes chères amis et collègues de la classe ISTW

2015/2016.

A tous ceux qui me connaissent et qui m'aiment surtout.

T.Karima

Septembre 2016

ملخص :

المعالجة الآلية للغات الطبيعية تعتبر مجالا من مجالات اللسانيات و الاعلام الالي، تهدف إلى تطوير برنامج قادر على معالجة البيانات اللغوية آليا بلغة طبيعية معينة وبتطبيق واضح المعالم كالتصحيح النحوي.

يهدف هذا العمل إلى إضافة دعم اللغة العربية في أداة التصحيح النحوي (LanguageTool)حيث التصحيح النحوي التلقائي، و الذي يعتمد على تحليل الجملة من أجل تحديد الكلمات التي يشكل تركيبها خطأ نحويا مصنفا في فئة، يعتبر ميزة مفتقدة إلى حد بعيد في أوساط مستخدمي البرمجيات الحرة.

المراحل المتتبعة تسمح لنا بمراقبة البنية الصرفية للكلمة و التحقق من توافقها مع المركبات الأخرى للجملة.

الكلمات المفتاحية :

التصحيح النحوي، التدقيق النحوي، اللسانيات، مصدر مفتوح ، أدوات اللغة ، المعالجة الالية للغات الطبيعية، الغموض الصرفي ، التقسيم ، الوسم النحوي .

Résumé :

Le Traitement Automatique des Langues Naturelles (TALN) est un domaine à la frontière de la linguistique et l'informatique, il a pour objectif de développer des logiciels capables de traiter de façon automatique des données linguistiques exprimées dans une langue naturelle donnée et pour une application bien définie telle que la correction grammaticale

L'objet de ce travail est d'ajouter le support de la langue arabe à l'outil de correction grammaticale initialement développé "languagetool".

La correction grammaticale automatique de l'arabe est une fonctionnalité qui fait cruellement défaut à la communauté des utilisateurs de logiciels libres basée sur une analyse de la phrase ayant pour tâche d'en identifier les mots mal organisés pour détecter une erreur de grammaire classée dans telle catégorie.

Les étapes suivies permettent de contrôler la structure morphologique du mot, et de vérifier la compatabilité avec les autres constituants de la phrase.

Mots-clés :

Correcteur grammaticale, vérification grammaticale, linguistique, code source ouvert, language-tool, traitement automatique des langues naturelles, ambiguïté morphologique, segmentation, Etiquetage Morphosyntaxique.

Abstract:

The Natural Language Processing (NLP) is a field in the linguistic border and computers; it aims to develop software capable of processing automatically linguistic data expressed in a given natural language and for application well defined as grammatical correction.

The purpose of this work is to add support for the Arabic language to grammatical correction tool originally developed "languagetool".

Automatic correcting grammar of Arabic is a feature that is sorely lacking in the free software user community based on an analysis of the sentence whose task is to identify the badly organized words to detect a grammatical error in classified such a category.

The steps taken to control the morphological structure of the word, and to check the compatibility with the other components of the sentence.

Key words:

Grammatical corrector, grammar checking, linguistic, open source, language-tool, Automatic Natural Language Processing, morphological ambiguity, segmentation, labeling Morphosyntactic

TABLE DES MATIERES

Introduction générale……..…1

Partie I : Etat de l'art

Chapitre I : La vérification grammaticale

I.1 Introduction……3

I.2 Traitement automatique de langage (TAL)……3

I.3 Vérification grammaticale……..……4

I.3.1 Définition de la grammaire……..…4

I.3.2 Définition : Les erreurs de grammaire……4

I.3.3 Définition de correction grammaticale……..4

I.4 Correcteurs grammaticaux……..…….……5

I.4.1 Historique…….…….…5

I.4.2 Evolution par génération……...…….…….....5

I.4.2.1 Génération1 : le mot…….……..5

I.4.2.2 Génération 2 : le mot et son contexte immédiat……6

I.4.2.3 Génération 3 : la phrase complète……...6

I.4.3 Définition de correcteur grammatical……..7

I.4.4 Rôle de correcteur grammatical…….……...7

I.4.5 Architecture de correcteur grammatical…….……..7

a) Segmentation (Tokenisation) ……..….....8

b) Etiquetage morphosyntaxique (Tagging)……..9

c) Détection d'erreurs……..…9

I.5 Problème général de la correction grammaticale…….….11

I.6 Conclusion…….……11

Chapitre II : Les systèmes de correction des erreurs grammaticales

II.1 Introduction…….….12

II.2 Vue générale sur les systèmes open source/close source.....12

II.2.1 Différents types de logiciels.....12

II.2.1.1 code ouvert (open source).....12

II.2.1.2 code fermé (Source fermée)…….13

II.2.1.3 Logiciel libre.....13

II.2.1.4 Logiciel propriétaire.....13

II.2.1.5 Logiciel commercial.....14

II.2.2 Avantages du Logiciel Libre et Open Source.....14

II.3 Correcteur open source …….14

II.3.1 LightProof ……..14

II.3.1.1 Définition de light proof……..14

II.3.1.2 Caractéristiques ….....14

II.3.1.3 Licence ……15

II.3.1.4 LightProof pour la langue arabe……15

II.3.2 Language Tools…….…..16

II.3.2.1 Définition de language Tools……16

II.3.2.2 Rôle …….....16

II.3.2.3 Licence et code source……....16

II.3.2.4 Historique……...16

II.3.2.5 Avantages (points forts) de LanguageTool ……19

II.3.2.6 Evaluation de LanguageTool……...19

II.3.2.7 LanguageTool VS LightProof……19

II.4 Conclusion ……20

Chapitre III : Les aspects des erreurs en langue arabe

III.1 Introduction ……21

III.2 La langue arabe, histoire et variantes ……..21

III.3 Caractéristique de la langue arabe……22

III.3.1 Structure du mot ……22

III.3.2 Catégories de mot ……..23

III.3.2.1 Nom ……23

III.3.2.2 Verbe ……...23

III.3.2.3 Particules ……...……..24

III.4 Grammaire de la langue arabe ……25

III.4.1 Morphologie, الصرف ……..25

III.4.2 Syntaxe, النحو ……..25

III.4.2.1 Phrase nominale ……25

III.4.2.2 Phrase verbale ……26

III.4.2.3 Règles d'accord ……26

a) Accord du verbe ……26

b) Accord de l'adjectif ……26

III.5 Analyse des erreurs grammaticales communes en langue arabe ……...27

III.6 Conclusion ……..30

Partie II : Conception et réalisation

Chapitre IV: Conception

IV.1 Introduction…….….31

IV.2 Morphologie : Principe et Objective……...31

IV.2.1 Principe……..31

IV.2.2 Objective……32

IV.3 Segmentation……..…..32

IV.3.1 Types de segmentation……..……..…..32

IV.3.2 Module de segmentation…….....33

IV.4 Etiquetage Morphosyntaxique…….34

IV.5 Différentes étapes/phases du prototype……..…….34

IV.5.1 Masculin au lieu de féminin……..……..35

IV.5.2 Féminin au lieu de masculin……..……..36

IV.5.3 Prépositions……..……..……36

IV.5.4 Accord de genre entre le verbe et le sujet……..……37

IV.5.5 Pluriels……..……..…..38

IV.5.6 L'accord entre les nombres et les noms qu'ils suivent……..38

IV.6 Conclusion……..39

Chapitre V: Implémentation et expérimentation

V.1 Introduction……..…….…40

V.2 Implémentation de l'approche……..……...…...40

V.2.1 Environnement de développement……..…40

V.2.1.1 Langage de programmation JAVA……..…40

V.2.1.2 Langage XML…….40

V.2.1.3 Environnement de développement ECLIPSE…….....41

V.3 Apache Maven……..41

V.3.1 Définition……..……...41

V.3.2 POM……..…….....42

V.3.3 Plugins……..….43

V.4 Encodage……..……..44

V.5 Présentation de l'application ……..……44

V.5.1 Création du projet et l'ajout du support « ar »…….45

V.5.2 Création des fichiers nécessaires java et xml……...46

V.5.2.1 Arabic.java…….....46

V.5.2.2 language-module.properties…….…...46

V.5.2.3 grammar.xml……47

V.5.2.3.1 Element <rule>……47

V.5.2.3.2 Element <example>…….....47

V.5.2.3.3 Element <message>…….……48

V.5.2.3.4 Element <pattern>……..…….48

V.5.2.3.5 Element <token>…….48

V.5.3 Integration de l'Arabe et résultat…….……..50

V.6 Développement des règles d'erreurs ……...51

V.6.1 Masculin au lieu de féminin……...51

V.6.2 Féminin au lieu de masculin…….……..52

V.6.3 Prépositions…….54

V.6.4 Accord de genre et nombre entre l'accusatif circonstanciel et le sujet qu'elle modifie……..……...55

V.7 Conclusion et perspective……..……56

Conclusion générale……..……59

Glossaire……60

Bibliographie……62

Webographie.……....65

LISTE DES FIGURES :

Chapitre I

Figure I.1: structure générale en couche d'un correcteur grammatical…….…….8

Chapitre II

Figure II.1 : Structure de LanguageTool à l'époque…….……17

Figure II.2 : Structure actuelle de LanguageTool…….…18

Chapitre IV

Figure IV.1 : Les types de segmentation.....33

Figure IV.2 : exemple d'extraction des segments.....33

Figure IV.3 : exemple d'étiquetage morphosyntaxique.....34

Figure IV.4 : Démarche de correction « catégorie Masculin au lieu de féminin ».....35

Figure IV.5 : Démarche de correction « catégorie Féminin au lieu de masculin ».....36

Figure IV.6 : Démarche de correction « catégorie Prépositions ».....36

Figure IV.7 : Démarche de correction « catégorie Accord de genre entre V et S ».....37

Figure IV.8 : Démarche de correction « catégorie Pluriels ».....38

Figure IV.9 : Démarche de correction « catégorie Autres erreurs communes ».....38

Chapitre V

Figure V.1 : Fichier pom.xml du projet « language-ar »…….....43

Figure V.2 : Création du projet « ar » BUILD SUCCESS……45

Figure V.3: Arabic.java…….….46

Figure V.4 : Interface de LanguageTool « Arabic Algérie ».....51

Figure V.5 : Résultat de la règle «هذا (هذه) المراجع »....52

Figure V.6 : Résultat de la règle «وقود كافية (كافي)» -1....53

Figure V.7 : Résultat de la règle «وقود كافية (كافي)» -2……..……..…53

Figure V.8 : Résultat de la règle « برهنت عن (على)» -1……..54

Figure V.9 : Résultat de la règle « برهنت عن (على)» -2……..55

Figure V.10 : Résultat de la règle «جاءت السيدات تحملن (يحملن) أطفالهن»....56

Figure V.11 : Proposition d'une structure pour un nouvel outil....58

LISTE DES TABLEAUX

Chapitre I

Tableau I.1. Tableau Récapitulatif des différentes générations de correcteurs……...6

Chapitre II

Tableau II.1. Exemple des erreurs testées en LightProof……..15

Tableau II.2 : Comparaison entre languagetool et lightproof……19

Chapitre III

Tableau III.1. Les erreurs grammaticales communes de l'Arabe……..29

LISTE DES ACRONYMES

Ar : Arabe

ASM: Arab Standard Modern

CC: coordinating conjunction

GRAC: Grammar Checker

HTML: Hyper Text Markup Language

IBM: International Business Machine

IDE: Integrated Development Environment

IN: preposition/ subordinate conjunction

ISO: International Organization for Standardization

JAR: Java Archive

JJ: adjective

JVM: Java Virtual Machine

LGPL: Lesser General Public License

LT: Language Tool

MSA: Modern Standard Arab

NN: Noum, singular or mass

POM: Project Object Model

SGML: Standard Generalized Markup Language

TAL : Traitement Automatique de la langue

TALA: Traitement Automatique de la Langue Arabe

TALN : Traitement Automatique de la Langue Naturelle

UTF-8: Unicode Transformation Format 8 bits

UTF-16: Unicode Transformation Format 16 bits

UTF-32: Unicode Transformation Format 32 bits

VRB: Verb present tense

War: Web application Archive

XML: eXtended Markup Language



Problématique



Problématique :

Le monde compte plus de 300 millions d'arabophones dont l'Arabe est l'une des langues les plus parlées. Langue officielle à l'Organisation des Nations Unies, à la Ligue arabe, à l'Organisation de la Conférence islamique et à l'Union africaine, elle devient une langue référent d'échanges, de commerce, de culture et de sciences.

La langue arabe possède plusieurs caractéristiques qui demandent des traitements particuliers pour qu'elle soit implémentée dans un programme, intégrée dans les équipements informatiques.

Les ressources linguistiques sont spécifiques à la langue et diffèrent donc d'une langue à l'autre, pour l'arabe, et malgré les nombreux travaux et recherches, ces ressources restent moins présentes, pour cela le traitement automatique de la langue arabe nécessite plus de recherches surtout dans les études qui traitent les lois et les règles régissant de cette langue tels que le système vocal, le système morphologique, système grammatical, système sémantique et le système écrit.

Le nombre des programmes spécialisées dans le traitement automatique de la langue arabe sur le marché des programmes informatiques ou sur le web, est très petit, et parfois même incomparable par rapport au nombre et à la qualité des programmes qui concernent d'autres langues, comme le français et l'anglais et même d'autres langues moins parlé et utilisé dans le monde par rapport à la langue arabe. Nous pouvons citer Hunspell comme correcteur d'orthographe spécialisé lors de sa première apparition à la langue hongroise et le languagetool; qui est très utilisé sur le web et qui traite plusieurs langues sauf la langue arabe.

La recherche de solutions au problème de correction grammaticale de texte est restée, depuis longtemps, un défi. Plusieurs chercheurs se sont penchés sur le problème et, grâce à leurs efforts, diverses techniques et de nombreux algorithmes ont vu le jour. La détection d'erreurs consiste à trouver les mots grammaticalement incorrects dans un texte.

La plupart des correcteurs grammaticaux sont payant et ils sont propriétaires, fermés, et on ne peut pas voir ses codes sources et connaitre leurs méthodes de travail. Pour l'Arabe, y a un seul correcteur grammaticale (libre) « LightProof » qui est intégré dans OpenOffice/LibreOffice. Ce programme propose de corriger la ponctuation et quelques erreurs connues. Ceci est une version de démonstration et dans la phase de développement.

Donc, tout au long de ce travail, nous allons travailler sur l'ajout du support de la langue arabe au système open source de correction des fautes grammaticales et linguistiques "LanguageTool". Les ressources linguistiques du "languagetool" resteront accessibles et modifiables, et dont le formalisme générique le rendra adaptable à d'autres recherches et améliorations.



Introduction générale



Introduction générale :

Avec la diffusion de la langue arabe sur le Web et la disponibilité des moyens de manipulation de textes arabes, les travaux de recherche ont abordé des problématiques plus variées comme la syntaxe, la grammaire, la traduction automatique, l'indexation automatique des documents, la recherche d'information, etc.

Le traitement automatique des langues et l'accès aux technologies modernes de l'information, et en particulier à Internet, a entraîné le développement d'outils spécialisés (logiciel de traitement de texte, correcteurs orthographiques et grammaticaux, dictionnaires en ligne, outils de recherche d'informations notamment sur le Web, outils d'aide à la traduction, synthèse et reconnaissance de la parole,…). Si ces outils s'imposent comme la condition nécessaire de la survie des langues, ils participent par ailleurs à un nouveau mode d'écriture où il apparaît, ces outils informatiques permettent à la fois de revenir facilement sur ce qui a été écrit, de le corriger, de l'améliorer et en même temps offre des outils d'aide à la rédaction et à la correction qui laissent à penser à l'utilisateur que l'ordinateur fera à sa place ce travail laborieux.

Objectif :

L'objectif de ce travail consiste à l'ajout d'un outil de correction grammaticale pour l'Arabe à un programme open source surnommé "languagetool", déjà existant sur le web depuis quelques années, et pouvant être intégré à l'environnement bureautique, ou utilisé indépendamment de celui-ci, et aussi pour objectif d'enrichir la langue arabe par des logiciels open source et enrichir les correcteur grammaticaux par des différentes langues. Dans ce domaine, les outils existants sont généralement des logiciels propriétaires aux coûts d'intégration élevés, qui sont peu ou pas décrits dans la littérature, et fermés à toute amélioration externe. Notre travail de recherche vise donc, de manière plus générale, à développer un outil de correction grammaticale générique, dont les ressources linguistiques seront accessibles et modifiables.

Organisation du mémoire :

Ce travail est organisé de la présente introduction générale, de deux partie état de l'art et la partie conception et réalisation, et d'une conclusion générale dans laquelle nous présentons les principales conclusions ainsi que les perspectives de nos travaux.

La première partie qui concerne l'état de l'art et une étude détaillée sur le contexte de notre travail. Elle se compose en trois chapitres :

Chapitre I : Le premier chapitre est réparti en trois sections La première est destinée à définir le traitement automatique de la langue (TAL) et ses applications. La seconde étudie brièvement la vérification grammaticale par la définition de la grammaire, les erreurs de grammaire, la correction grammaticale. La troisième section est consacrée à la présentation des correcteurs grammaticaux qui fournissent un banc d'essai pour l'évaluation de notre système de reconnaissance.

Chapitre II : Le deuxième chapitre est consacré essentiellement aux systèmes open / close source, nous allons aborder dans un premier temps une vue générale sur les systèmes open source/close source où nous décrivons les principaux types de logiciels. Ensuite nous présentons une petite description sur « lightproof » comme la définition et les caractéristiques et quelques erreurs testées, nous terminons par une étude détaillée sur language tool.

Chapitre III : Dans le troisième chapitre nous étudions brièvement la langue arabe comme son origine et ses caractéristiques…etc, ensuite nous détaillons la Grammaire de la langue arabe et nous terminons par analyse des erreurs grammaticales communes en langue arabe.

La deuxième partie concerne notre contribution et la présente en détails. Elle comporte deux chapitres.

Dans le premier chapitre de cette partie, nous présentons notre approche par une conception. Nous commençons par présenter le principe de la morphologie et ces objectifs, La segmentation et l'étiquetage Morphosyntaxique. Nous donnons ensuite la démarche de correction de différentes catégories de cette conception.

Dans un dernier chapitre, nous présentons l'implémentation de notre approche après avoir définit l'environnement de développement et les déférents outils utilisés. Ce chapitre comporte aussi les résultats des tests effectués afin de l'évaluer et d'en mesurer les performances.

PARTIE 1 :



Etat de l'art



CHAPITRE I:



I.1 Introduction :

L'étude du langage naturel est un domaine d'études fascinant et riche en applications potentielles. Cependant de nombreux progrès restent à accomplir pour mieux comprendre cette faculté et pour bâtir des systèmes capables de soutenir la comparaison avec l'humain, mais l'état des connaissances permet aujourd'hui de proposer de nombreuses solutions efficaces à des problèmes et à des demandes réelles.

Aujourd'hui, le champ du traitement automatique du langage naturel (TAL) est devenu un domaine à la fois industriel en plein effervescence dû à l'émergence d'un nombre incalculables d'applications dont la correction grammaticale est l'une de ses technologies les plus utilisées du grand public. En effet, elle est intégrée à des outils informatiques utilisés chaque jour par des millions de personnes à travers le monde.

Au cours de ce chapitre, nous commençons par définir quelques notions de TAL puis nous passons à la vérification grammaticale : définition de la grammaire, les erreurs de grammaire, la correction grammaticale et enfin nous présentons la définition des correcteurs grammaticaux et ses divers titres.

I.2 Traitement automatique de langage (TAL) :

Le traitement automatique des langages (TAL) est l'application des mathématiques et de l'informatique à la linguistique. Il est aussi une technologie qui comprend l'ensemble des applications informatiques portant sur les langues humaines. [P. GUY, 14]

Les principales applications du TAL :

- La traduction assistée par ordinateur.
- La correction orthographique et grammaticale.
- La recherche d'informations textuelles, les moteurs de recherche.
- La fouille de textes, l'indexation de documents.
- Le résumé automatique.
- La génération automatique de textes.
- La synthèse de la parole, la reconnaissance vocale.
- La reconnaissance de l'écriture manuscrite.

Le TAL est guidé par deux paradigmes :

Paradigme symbolique : vise à modéliser les connaissances linguistiques à l'aide de systèmes symboliques.

Paradigme stochastique : vise à extraire l'information linguistique de corpus à l'aide de méthodes stochastique, ou tous qui sont statistique. [P. GUY, 14]

Le TAL est organisé selon deux sens :

Analyse : qui va des énoncés à leur interprétation pragmatique.

Génération : qui mène des buts pragmatiques aux énoncés représentant leur réalisation linguistique. [P. GUY, 14]

I.3 Vérification grammaticale :

I.3.1 *Définition de la grammaire :*

La grammaire peut être présentée comme l'ensemble des règles de la langue, comme l'ouvrage qui les contient, ou de manière plus globale comme l' « étude systématique des éléments constitutifs et du fonctionnement de la langue »Grevisse 1993. [A. Souque, 14]

I.3.2 *Définition : Les erreurs de grammaire :*

Les erreurs de grammaire comprennent à la fois les erreurs d'orthographe grammaticale (أنت يضع) et les erreurs syntaxiques (انشاء الله) .Elles sont appelées également « erreurs morphosyntaxiques » puisqu'elles concernent la morphologie de la syntaxe. [A. Souque, 14] , Ce sont sur les erreurs de ce type nous focalisons notre travail.

I.3.3 *Définition de correction grammaticale:*

La correction grammaticale vise à détecter les erreurs pour lesquelles la correction orthographique n'est pas compétente. Elle est ainsi chargée d'une part des erreurs concernant l'organisation de la phrase (ex : l'ordre des mots) et les relations entre les mots (ex : accords),

et d'autre part des erreurs de graphie qui conduisent à un mot existant mais incorrect dans le contexte de la phrase. [A. Souque, 14]

I.4 Correcteurs grammaticaux

I.4.1 *Historique :*

La correction orthographique et grammaticale par ordinateur est surement la technologie du traitement automatique des langues (TAL) la plus répandue à travers le monde en tant que composante des outils de traitement de texte.

Dans les années 90, la correction de texte assisté par ordinateur est passé du laboratoire au terrain commercial ce qui explique en partie qu'elle semble négligée par les recherches en TAL. En effet les chercheurs industriels publient très peu pour des raisons de secret commercial. De plus, le monopole de la suite Office de Microsoft dans le monde du traitement de texte entraine un manque d'intérêt pour ce type d'application, qui est perçu comme la chasse gardée de la multinationale. Enfin, la performance des systèmes commerciaux actuels est satisfaisante compte tenu de l'effort qui serait requis pour l'améliore d'une manière significative. [VIENNEY& BIOUD, 04]

Les correcteurs orthographiques existent depuis plusieurs décennies, tandis que les correcteurs dits « grammaticaux » n'arrivent véritablement sur le marché des Industries de la langue que vers les années 1990. [VIENNEY& BIOUD, 04]

I.4.2 *Une évolution par génération :*

Une séparation des différentes technologies en « générations » permet de préciser simplement les apports technologiques particuliers décelables dans plusieurs logiciels de correction et d'en extraire les principes de correction propres à chacun.

I.4.2.1 *Génération1 : le mot*

La première génération de correcteurs, appelés avec justesse correcteurs orthographiques, se contentait de vérifier si un mot d'un texte faisait partie d'un lexique de référence le plus souvent stocké sur disque. L'idée de départ consistait à supposer que si un mot était absent du lexique, il devait être mal écrit.

Le critère principal de la qualité de ces outils est la couverture lexicale : le nombre de mots courants, la prise en compte des variantes orthographiques, le traitement des locutions, les noms propres, les emprunts aux langues étrangères, les abréviations, les sigles, les acronymes, et les autres éléments du lexique comme les nombres, les symboles, les unités de mesure, les unités monétaires, les dates, les adresses, les équations, etc.

I.4.2.2 Génération 2 : le mot et son contexte immédiat

La deuxième génération de correcteurs s'appuie sur une analyse syntaxique locale. Le correcteur prend en compte l'environnement immédiat du mot à vérifier. La faute type recherchée est la faute d'accord et principalement l'erreur de nombre ou de genre. La liste de formes orthographiques présente dans la première génération fait place à un dictionnaire de mots contenant des informations supplémentaires permettant d'étiqueter la catégorie grammaticale de chaque mot et de définir le genre et le nombre des noms, pronoms, des adjectifs et des déterminants ainsi que la ou les personnes pour les verbes.

I.4.2.3 *Génération 3 : la phrase complète*

L'analyse de la phrase complète est l'élément le plus visible de la transition vers la troisième génération. Cette génération fait intervenir un analyseur morphosyntaxique à large couverture. Le dictionnaire de mots devient alors une véritable base de données de référence comprenant des informations syntaxiques servant à valider la construction du groupe ainsi que sa dépendance. [VIENNEY& BIOUD, 04]

	1re génération	2e génération	3e génération
Niveau d'analyse	Mot	mot + contexte local	Phrase
Méthode de détection	Lexique de formes fléchies	Lexique étiqueté+ règles d'analyse locales	Lexique étiqueté + règles locales + analyse profonde de la phrase
Principales erreurs détectées	Orthographe lexicale	Orthographe, accords (surtout genre et nombre)	Orthographe, accords (SN,verbes, part.passé), conjugaison,syntaxe,etc.
Taux d'erreurs détectées	≈ 40%	≈ 60%	≈ 80%
Exemples de logiciels	Vérificateurs intégrés aux suites bureautiques		
	Vérificateurs d'orthographe intégrés aux systèmes d'exploitation	LanguageTool, An Gramadoir	Antidote, Cordial, Correcteur 101

Tableau I.1 : Tableau Récapitulatif des différentes générations de correcteurs [A.SOUQUE,16]

I.4.3 *Définition de correcteur grammatical :*

Un correcteur grammatical est un logiciel qui permet de vérifier et de corriger un texte en langue naturelle, saisi sur format électronique, par exemple, au sein d'un traitement de texte. Comme son nom l'indique, il doit franchir l'étape de la vérification orthographique, pour atteindre et intégrer des aspects relevant de la vérification grammaticale. La «détection » d'erreurs et l'intervention d'une « correction » s'avèrent cruciales et justifient la raison d'être de ce type d'application. [CHARNET & PANCKHURST, 98]

I.4.4 *Rôle de correcteur grammatical :*

- Permettre d'améliorer la qualité du document réalisé.
- Produire un texte "sans fautes".
- Utiliser un correcteur comme une aide à la réflexion, comme un outil, un soutien dans le cadre de l'enseignement d'une langue.
- Cibler les informations de base pour les apprenants.
- Fournir une explication des grandes catégories grammaticales, des règles grammaticales, des exceptions principales et des fautes à éviter.
- Identifier les informations essentielles pour éviter les fautes qui se trouvent fréquemment dans les compositions d'apprenants.
- Réduire le nombre de fautes présentes dans un travail écrit. [C. Jacquet-Pfau, 01]

I.4.5 *Architecture de correcteur grammatical :*

Le fonctionnement d'un correcteur grammatical est plus complexe que le correcteur d'orthographe. Il se décompose classiquement en plusieurs étapes successives de traitement, effectuées généralement après la rédaction. En effet, les différentes étapes réalisent soit des analyses ascendantes des phrases (approche bottom-up), en partant des mots graphiques pour arriver aux syntagmes par exemple, soit au contraire des analyses descendantes (approche top-down), allant de la phrase vers le mot. Dans un cas comme dans l'autre, il est alors nécessaire que les phrases soient terminées pour que les différents traitements soient fonctionnels. [A. Souque, 14]

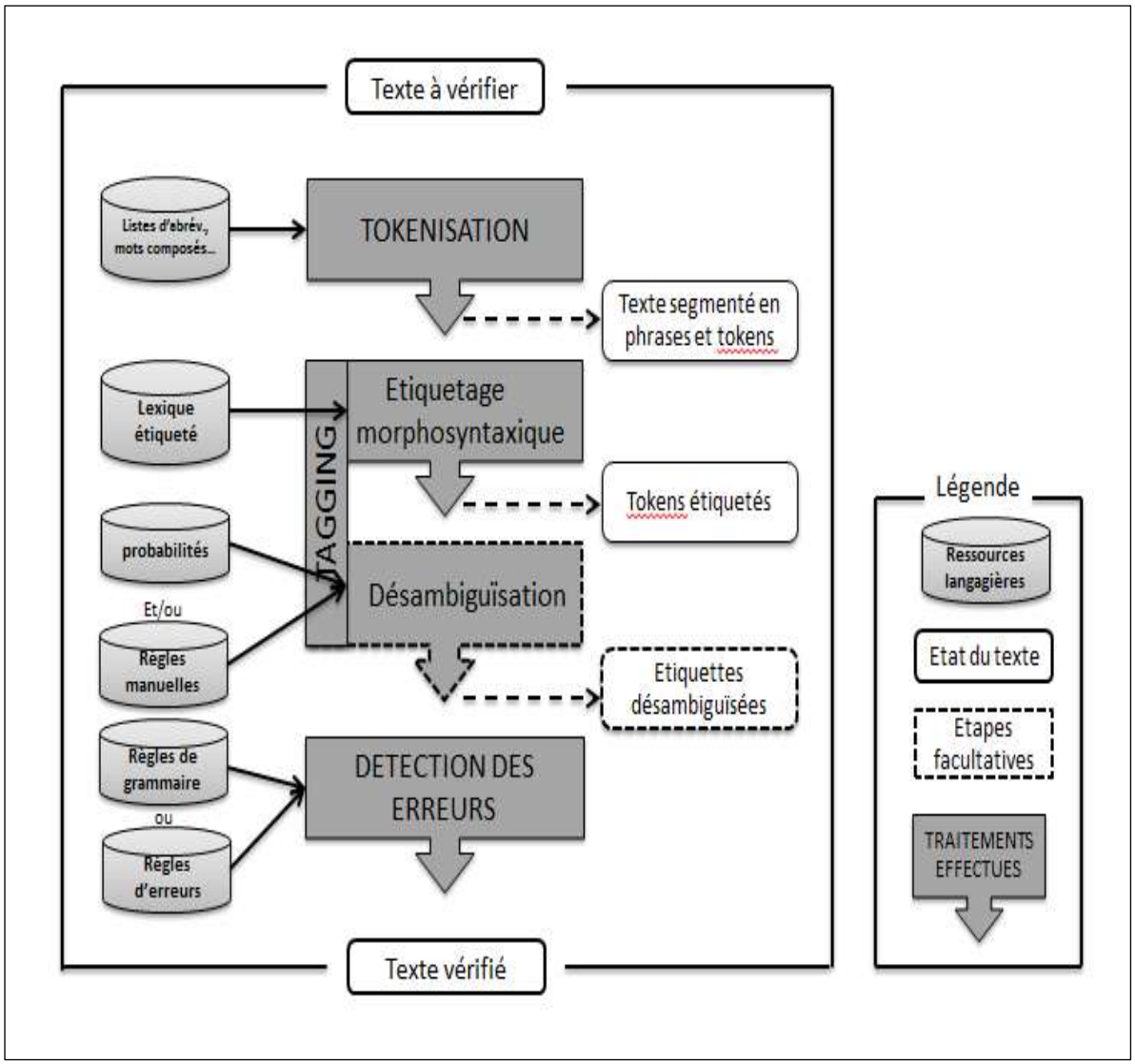


Figure I.1: structure générale en couche d'un correcteur grammatical [A. Souque, 14]

a) Segmentation (Tokenisation) :

La première étape de traitement consiste à découper la chaine de caractères en phrase et en mots (tokens). Nous appelons tokens les séquences de caractères du texte délimitées par des blancs ou des ponctuations. Token englobe en effet d'autres éléments graphique que les mots, comme les ponctuations, les nombres, les sigles.

La segmentation en tokens, n'est pas aussi triviale qu'elle peut sembler l'être à première vue, mais c'est sur elle que reposent par la suite tous les traitements et analyses du texte. Elle doit donc être réalisée le mieux possible, malgré les diverses difficultés auxquelles elle se heurte. [A. Souque, 14]

b) Etiquetage morphosyntaxique (Tagging) :

Au niveau de cette étape, l'étiqueteur attribue à chaque mot une ou plusieurs étiquettes (tag), contenant des informations sur sa catégorie grammaticale (verbe, nom, pronom …) , ainsi que sur ses traits de sous-catégorisation (genre, nombre, temps, personne…), appelés aussi traits morphosyntaxiques. Ces tags proviennent d'un lexique de formes fléchies étiquetées. Il s'agit de lexiques qui contiennent, comme pour les correcteurs orthographiques, toutes les formes fléchies de la langue, mais complétées par leurs caractéristiques morphosyntaxiques. Beaucoup de mots, dits « ambigus », reçoivent plusieurs tags. Ce sont les mots ayant des homographes avec des informations morphosyntaxiques différentes, on prend par exemple le segment « ذهب » qui on peut le considérer comme un verbe (الذهاب) et aussi comme un nom désignant l'or.

Cette multitude de tags pour un même mot peut facilement conduire à une mauvaise analyse morphosyntaxique du texte. Une désambiguïsation est donc une solution effectuée pour limiter le nombre d'étiquettes de ces mots et améliorer par la suite la détection des erreurs de grammaire. L'approche probabiliste est une approche pour désambiguïser les mots, elle nécessite un corpus d'apprentissage sans erreur, étiqueté avec les informations morphosyntaxique. Des calculs sont alors effectués. Il s'agit de la probabilité pour chaque mot d'avoir tel ou tel tag. Lors de l'étiquetage, ces probabilités sont appliquées à chaque mot du texte analysé, et chacun reçoit alors l'étiquette qui correspond à la plus forte probabilité. L'algorithme de Brill 1997 permet de générer des règles de désambiguïsation établies statistiquement.

Une autre approche consiste à utiliser des règles manuelles de désambiguïsation, sous forme d'expression régulières et fondées sur le contexte immédiat. Chaque règle consiste en un modèle d'un contexte en présence duquel tel mot prend tel tag. [A. Souque, 14]

c) Détection d'erreurs :

Cette dernière concerne la vérification de la grammaire. Il existe des règles de deux types :

- ✓ **Règles de grammaire :**

Certains correcteurs comme GRAC utilisent des règles de grammaire. Ces règles décrivent des modèles, des combinaisons de mots grammaticalement correctes. Chaque segment du texte à vérifier est comparé aux modèles contenus dans les règles de la base. S'il

est agrammatical, aucune correspondance exacte ne sera trouvée entre lui et les règles de grammaire, puisqu'elles répertorient uniquement des constructions syntaxiques correctes. Une erreur sera alors signalée par l'outil.

Avec le système utilisant les règles de grammaire, des phrases grammaticales peuvent être à tort déclarées agrammaticales simplement parce qu'elles contiennent une combinaison de mots qui n'est pas décrite de manière exactement identique dans les règles, mais qui n'en est pas moins correcte. Ce phénomène de fausse alarmes est communément appelé « bruit » dans pas moins correcte. [A. Souque, 14]

- ✓ **Règles d'erreurs :**

LanguageTool est l'un des correcteurs qui utilisent des règles d'erreurs. Le principe est le même que pour les règles de grammaire, à la différence près que ce sont cette fois les combinaisons de mots agrammaticales qui sont décrites dans les règles. Une erreur de grammaire est alors signalée par le correcteur lorsqu'un segment du texte et le modèle d'une règle coïncident.

Avec le système utilisant ces règles d'erreurs, des phrases agrammaticales peuvent ne pas être repérées, et considérées comme correctes, si les structures erronées qu'elles contiennent ne sont définies dans aucune règle. Il s'agit dans ce cas de « silence ».

Ce système de détection d'erreurs à base de règles présente des limites importantes, dont la principale tient au fait qu'il s'agit d'un système rigide qui impose que les segments de texte analysés coïncident parfaitement avec les patterns des règles. Ceci a pour conséquence que la détection d'une erreur échoue à partir du moment où il y a une différence, même minime, entre le segment de texte traité et le modèle dans la règle correspondante. Il peut s'agir d'un mot mal orthographié, mal étiqueté, ou encore d'un mot en plus ou en moins.

Donc, pour que la détection d'erreurs soit optimale, que ce soit avec les règles de grammaire ou avec les règles d'erreurs, il est nécessaire, mais cependant impossible, de répertorier absolument toutes les constructions syntaxiques possibles, soit correctes (pour les règles de grammaire), soit erronées (pour les règles d'erreurs). [A. Souque, 14]

I.5 Problème général de la correction grammaticale

Un correcteur grammatical est un outil qui a deux fonctionnalités principales :

- ✓ Il signale les phrases ou fragments incorrects (probablement erronés) dans un document.
- ✓ Il propose des corrections, avec éventuellement une explication « linguistique » de l'erreur.

En pratique, on demande des propriétés additionnelles à un correcteur grammatical :

- ✓ qu'il soit rapide : la vérification doit s'effectuer « en temps réel » au cours de la frappe.
- ✓ qu'il minimise le bruit : s'il y a trop de fausses alertes, les utilisateurs le débrancheront.
- ✓ qu'il minimise le silence : il doit laisser passer peu d'erreurs. [Clément & Kim & Marlet,09]

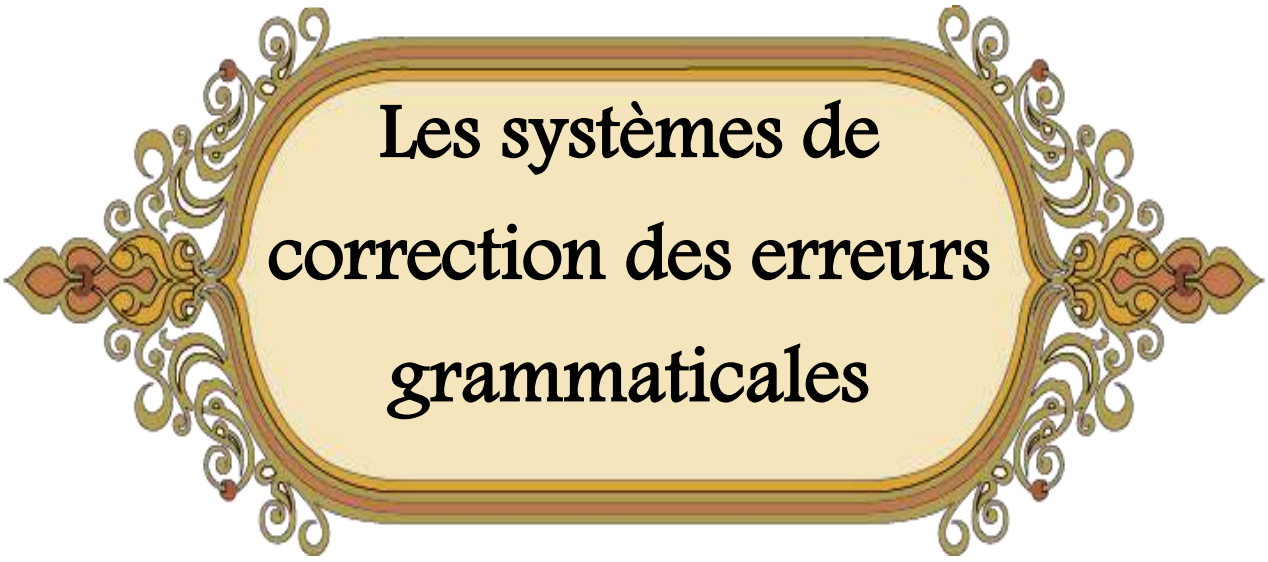
I.6 Conclusion :

La connaissance du fonctionnement des correcteurs doit conduire l'utilisateur à considérer cet outil comme un vérificateur, une aide à la correction, plus ou moins élaborée selon ses besoins.

Il est de plus en plus fréquent, de prévoir une nouvelle génération de correcteurs qui utiliseraient des techniques relevant de l'intelligence artificielle, qui permettraient de prendre en compte la dimension sémantique sans laquelle la vérification et la correction d'un texte ne peuvent être réellement performantes. Mais un correcteur conçu pour un type de corpus indifférencié ne pourra jamais remplacer un correcteur humain. En cela l'art du correcteur reste très proche de celui du traducteur.

Après avoir présenté les notions de TAL et de la grammaire dans leurs divers titres, nous avons retenu dans le cadre de ce travail, la définition de grammaire, les erreurs de grammaire, la correction grammaticale, le problème qu'elle oppose et enfin le détail du fonctionnement des correcteurs grammaticaux de façon générale.

CHAPITRE II:



II.1 Introduction :

L'utilisation croissante des ordinateurs à la fois pour la saisie de documents, manuelle, par scanner ou par saisie vocale, et pour la recherche dans des bases de données (encyclopédies, internet…) a rendu nécessaire l'intégration de modules de correction au minimum orthographique dans ce type de logiciel. Dans certains d'entre eux, notamment les traitements de texte, sont également intégrés des modules de correction grammaticale, minimale pour ne pas trop alourdir le système en termes de mémoire et de temps.

Ces jours-ci, les vérificateurs grammaticaux, bien que toujours loin d'être parfait, sont beaucoup mieux et plus faciles à utiliser. En fait, il est difficile de les ignorer.

Dans le cadre de cette recherche, notre objectif est de concevoir et de réaliser un système de détection et de correction des erreurs en langue arabe, capable d'une part de détecter le maximum de fautes réelles, et d'autre part de minimiser le maximum possible la liste des propositions générée pour la correction de chacune de ces fautes.

Au long de ce chapitre nous allons présenter un panorama sur les systèmes open/close source, suit par une bref présentation du correcteur grammatical Lightproof, et enfin nous détaillerons l'outil grammatical Language tool qui compose le titre principale de notre travail.

II.2 Vue générale sur les systèmes open source/close source :

II.2.1 *Différents types de logiciels :*

On classe les logiciels d'après la disponibilité du code source et de la licence qui régit la distribution du programme :

II.2.1.1 *code ouvert (open source) :*

Est un programme dont le code source est distribué et peut être utilisé, copié, étudié, modifié et redistribué sans restriction dont tout le monde peut lire le code source.[P. Bertrand, 11]

Caractéristiques de l'open source :

- ✓ Redistribution libre du logiciel.
- ✓ Accès libre au code source.
- ✓ Possibilité de modifier et redistribuer les travaux dérivés du programme à la condition que ce soit sous la même licence que l'original.
- ✓ Intégrité du code source de l'auteur permettant de différencier les différents contributeurs et contributions.
- ✓ Pas de discrimination envers des personnes ou groupes afin de permettre à tout le Monde de contribuer.
- ✓ Pas de discrimination envers des domaines d'application en ouvrant le projet à tout domaine, même commercial par exemple.
- ✓ Distribution de la licence permettant à tous les acteurs utilisant le programme de bénéficier des mêmes droits sans ajout de nouvelle licence.
- ✓ La licence ne doit pas être spécifique à un produit.
- ✓ La licence ne doit pas restreindre d'autres programmes.
- ✓ La licence doit être techniquement neutre. [K. Ronda, 14]

II.2.1.2 *code fermé (Source fermée) :*

Est un terme pour un logiciel dont la licence ne permet pas pour le libérer ou de la distribution du code source du logiciel ou ce dernier n'est disponible que pour une minorité de personne. Le code source de ces programmes est généralement considéré comme un secret commercial de la société. L'accès au code source par des tiers nécessite souvent la partie à signer un accord de non-divulgation**.**

II.2.1.3 *Logiciel libre :*

Tout le monde peut étudier, copier, modifier et distribuer des versions modifiées du logiciel (définition de la Free Software Foundation.) Les logiciels libres sont protégés pour la plupart par une licence d'utilisation ; Pour autant cela ne signifie pas que le logiciel est gratuit.

II.2.1.4 *Logiciel propriétaire :*

Logiciel propriétaire est une expression qui désigne les programmes d'ordinateur dont l'utilisation est soumise à des restrictions concernant leur modification privée ou à des restrictions concernant la copie ou la publication de versions modifiées ou non modifiées, ces restrictions sont imposées au logiciel par un de ses propriétaires.

II.2.1.5 *Logiciel commercial:*

Logiciel destiné à la vente, il peut être libre ou propriétaire. [SOUSSI HAYMEN].

II.2.2 *Avantages du Logiciel Libre et Open Source :*

- **Avantages techniques**

Le code source libre d'accès permet à plus de gens de voir le code et de le corriger. Il peut ainsi se développer plus vite et mieux. Ce système peut être comparé à la façon dont fonctionne la recherche scientifique.

- **Avantages économiques**

Partager les coûts de développement, Le logiciel libre et open source crée un marché concurrentiel pour le support, et augmente potentiellement la qualité de celui-ci. Avec les logiciels propriétaires, seul le fournisseur qui a accès au code source peut offrir réellement un support décent, et a donc une sorte de monopole. [1]

II.3 Correcteur open source :

II.3.1 *LightProof:*

II.3.1.1 *Définition de light proof :*

LightProof (Nemeth, 2011) est un vérificateur de grammaire Open Source en cours de développement écrit en Python pour suite bureautique OpenOffice.org. Il est fondé sur des règles de système pris en charge par eux ou analyseur morphologique Hunspell (Nemeth, 2011). [D.William,13]

II.3.1.2 *Caractéristiques:*

- Extension pour OpenOffice.org 3.0.1 ou version ultérieure.
- conditions Python facultatifs dans les règles.
- intégration avec l'analyseur morphologique Hunspell.
- écrit en Python en utilisant le module regex (expression régulière).
- Le développement rapide de nouveaux modules de langues.[2]

II.3.1.3 *Licence:*

GNU LGPL v2.1 Lesser General Public License. [3]

II.3.1.4 *LightProof pour la langue arabe:*

Ce programme propose de corriger la ponctuation, et quelques erreurs courantes.

Les erreurs communes telles que : **علاوة على ذلك**

كثب–>من كثب#خطأ شائع:لأن الكثب هو القرب

ثبت بأنه–> ثبت أنه# خطأ شائع: لأن ثبت فعل لازم يتعدى بالهمزة والتضعيف

غسيل الأموال–>تبييض الأموال#لأنها ترجمة مبتذلة لـ (Money laundering).

Ceci est une version de démonstration dans la phase de développement et non dirigée pour l'utilisation étendue. [4]

L'erreur détectée	La version correcte
لم يخشى	لم يخش
لم ينهى	لم ينه
لم يرى	لم ير
من عشرون	من عشرين
الآنف الذكر	المذكور آنفا
بشكل عام	بوجه عام
أمر هام	أمر مهم
تفرقت الآراء	افترقت الآراء
لصالح	لمصلحة
إنشاء الله	إن شاء الله
هذه الرأس	هذا الرأس
هذا الأرنب	هذه الأرنب

Tableau II.1 : Exemple des erreurs testées en LightProof [5]

II.3.2 *Language Tools :*

II.3.2.1 *Définition de language Tools :*

LanguageTool (développé par Daniel Naber 2003) est un correcteur grammatical libre plurilingue open source utilisable en ligne ou installable directement sur ordinateur pour le français, l'anglais, l'allemand, le polonais, le breton, l'espéranto et plus de 20 autres langues. Il est capable de trouver de nombreuses erreurs qui ne peuvent pas être signalées par un simple correcteur orthographique, les erreurs de grammaire telles que les accords en genre ou en nombre, les conjugaisons incorrectes, etc. [6]

II.3.2.2 *Rôle :*

- Un bon outil, simple et gratuit, à utiliser soi-même ou à conseiller pour vérifier grammaire, orthographe, et règles typographiques d'un texte avant de le publier ou de le rendre. Les logiciels professionnels sont souvent très onéreux, mais heureusement il existe des solutions en ligne.
- LanguageTool trouve des erreurs en recherchant des motifs définis par des règles XML. Il est aussi possible de définir des règles en Java. On peut aussi l'installer sous forme d'applications pour OpenOffice et LibreOffice, directement sur PC (LanguageTool nécessite Java 7) ou via une extension Firefox.[6]

LanguageTool peut être utilisé :

- ✓ En tant que LibreOffice / OpenOffice.org Extension
- ✓ Comme une application autonome
- ✓ Comme une application autonome sur la ligne de commande
- ✓ En tant que Firefox Add-On [7]

II.3.2.3 *Licence et code source :*

LanguageTool est disponible sous la licence LGPL. Le code source est disponible dans le dépôt git de github. [6]

II.3.2.4 *Historique :*

- **Structure de LanguageTool à l'époque :**

La figure ci-dessous représente la structure de la première version de LanguageTool. Elle utilisait à la fois des règles, pour la correction, et des probabilités pour la

désambiguïsation. Elle était composée de plusieurs modules programmés dans le langage python. [A. Souque, 07]

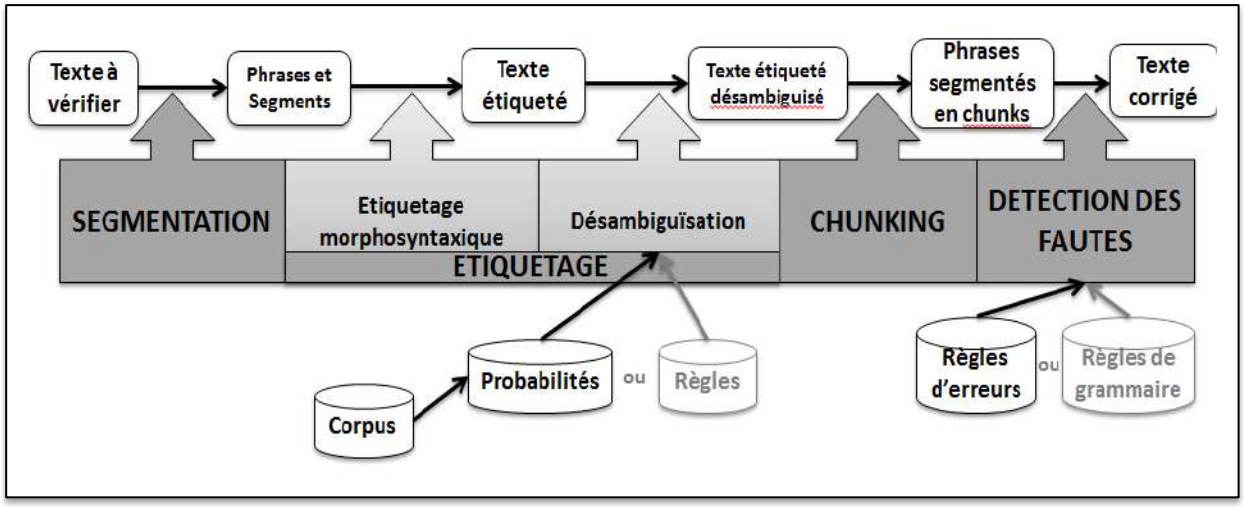


Figure II.1 : Structure de LanguageTool à l'époque [A. Souque, 07]

Le texte était d'abord découpé en phrases, puis en mots, ce qu'on appelle la segmentation (tokenisation). Les mots étaient ensuite étiquetés, et désambiguïsés par une méthode statistique, qui nécessitait au préalable un apprentissage sur corpus. Pour chaque mot, LanguageTool déterminait la probabilité que ce mot ait tel tag, puis la probabilité qu'il se trouve dans un certain contexte (entre un tag A et un tag B). Le tag pour lequel la somme de ces deux probabilités était la plus forte était appliquée au mot. Des règles écrites manuellement venaient en complément pour désambiguïser les mots qui n'avaient pas pu l'être avec le système de probabilités.

Suite au tagging, LanguageTool effectuait une étape de chunking, c'est-à-dire de segmentation des phrases en syntagmes minimaux (chunks), principalement nominaux ou verbaux. Les chunks étaient déterminés en fonction de patrons prédéfinis, contenus dans un fichier. Les règles définissant les patrons de chunks étaient du type : début de chunk, fin de chunk, nom du chunk. Lors de l'analyse de la phrase, si une suite de mots correspondait à un modèle, alors cette suite était définie comme chunk.

Arrivait enfin l'étape de détection des fautes de grammaire. Cette étape utilisait une base de règles d'erreurs formalisées en XML. Chaque règle contenait un identifiant (id), un nom (name), le modèle de l'erreur (pattern), une explication de la règle (message) à destination de l'utilisateur, et un exemple de l'erreur commise (example).

Les règles pouvaient aussi bien porter sur les mots que sur les tags ou sur les chunks. [A. Souque, 07]

- **Structure actuelle de LanguageTool :**

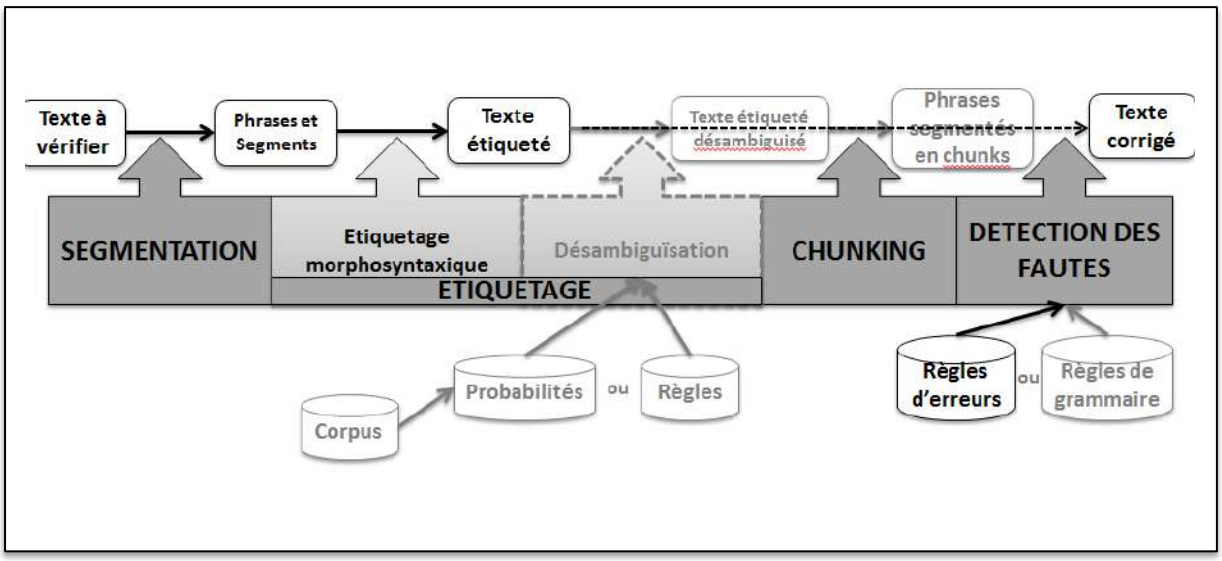


Figure II.2: Structure actuelle de LanguageTool [A. Souque, 07]

La nouvelle structure de LanguageTool, telle qu'illustrée dans la figure II.2, est actuellement composée de plusieurs classes Java qui effectuent :

➔ La segmentation du texte en phrases, puis en mots.

➔ L'étiquetage morphosyntaxique des mots : à partir d'un lexique de formes fléchies, chacune ayant une étiquette avec des informations morphosyntaxiques (catégorie grammaticale et traits de sous-catégorisation), chaque mot du texte reçoit une étiquette, ou plusieurs s'il est ambigu.

Aucune désambiguïsation n'étant effectuée, les mots qui ont plusieurs étiquettes les conservent toutes.

➔ La détection des erreurs de grammaire : elle s'effectue par comparaison du texte avec une base de règles décrivant des erreurs. Si une partie du texte correspond à une règle, alors une faute est signalée.

Les règles d'erreurs sont formalisées en XML, et composées de plusieurs éléments :

- **id et name** : l'identifiant et le nom de la règle.
- **Pattern** : le modèle de l'erreur. Le texte lui est comparé lors de la vérification. Il se présente généralement sous forme d'expressions régulières.

- **Message** : une description de la règle, à l'usage de l'utilisateur.
- **Example** : un exemple d'énoncé correspondant à la faute commise. Il y a en général au moins deux exemples : un exemple d'énoncé correct et un exemple d'énoncé incorrect. [A. Souque, 07]

II.3.2.5 *Avantages (points forts) de LanguageTool :*

✓ Le nombre de langues prises en charge, plus d'une vingtaine. Les langues principales Anglais, Français, Espagnol mais aussi d'autres, moins courantes comme le khmer ou encore l'ukrainien, le service prend également en charge plusieurs langues régionales comme le catalan ou le breton. Une diversité de langues qu'il est difficile de trouver y compris dans des outils payants.

✓ La possibilité de l'utiliser soit en ligne soit depuis un ordinateur sans avoir besoin de connexion à Internet. Outre le site, LanguageTool peut être téléchargé sur un ordinateur, il peut également être intégré aux suites Libre Office et Open Office. Il peut enfin être intégré grâce à une extension du navigateur Firefox.[8]

II.3.2.6 *Evaluation de LanguageTool :*

Pour évaluer LanguageTool, Naber a utilisé le British National Corpus (BNC, 2003) comme un échantillon de texte révisé pour réduire au minimum le nombre de fausses erreurs.

Il a également créé deux autres corpora La première a été extrait de listes de diffusion et les erreurs ont été annotés et classés manuellement. Le second a été créé en effectuant des recherches sur Internet des modèles d'erreur. Les résultats ont été compilés et utilisés où le motif d'erreur ne serait pas applicable, ce qui provoque une fausse erreur. [D.William,13]

II.3.2.7 *LanguageTool VS LightProof :*

LanguageTool	LightProof
vérificateur de grammaire Open Source en cours de développement écrit actuellement en java.	vérificateur de grammaire Open Source en cours de développement écrit en Python.
La langue arabe est non intégrée	La langue arabe existe en LightProof
développé en 2003	développé en 2011
Utilisé dans open office et comme une application autonome	Utilisé dans open office

Tableau II.2 : Comparaison entre languagetool et lightproof

II.4 Conclusion :

Les correcteurs grammaticaux ont vocation à détecter les erreurs pour lesquelles les correcteurs orthographiques ne sont pas compétents. Ils sont ainsi chargés d'une part des erreurs concernant l'organisation de la phrase (ordre des mots, …) et les relations entre les mots (accords,…), et d'autre part des erreurs de graphie qui conduisent à un mot existant mais incorrect dans le contexte de la phrase (confusion d'homophones, mauvais accord, ...etc).

Dans ce chapitre, nous avons vu ainsi les différents types de logiciels d'après la disponibilité du code source, citons quelques exemples de correcteurs grammaticaux pour chaque type. Ensuite, une définition du correcteur grammatical LightProof tant qu'il est chargé pour détecter les erreurs de grammaire de la langue arabe, et enfin une présentation globale sur LanguageTool notre outil pour traiter le problème de la correction grammaticale de l'Arabe.

CHAPITRE III



III.1 Introduction :

L'Arabe aujourd'hui gagne son importance dans le monde moderne globalisé, parlée dans de nombreux pays, et est de plus en plus populaire pour les communications sociales, les interactions d'affaires, et à des fins religieuses.

Dans la littérature, l'analyse d'erreur ne concerne que les erreurs grammaticales arabes les plus courantes, sans aucune indication de la fréquence d'apparition de ces erreurs. Cependant, il existe un besoin pour une étude approfondie qui répond à des questions telles que les suivantes :

Quelles sont les erreurs grammaticales les plus fréquentes? Quelles sont les erreurs pour un groupe linguistique particulier tel que les écrivains et les apprenants de la langue arabe sont les plus fréquentes? Quels sont les différents types de ces erreurs grammaticales?

III.2 La langue arabe, histoire et variantes:

L'arabe est une langue parlée par plus de 300 millions de personnes. Elle est langue officielle d'au moins 22 pays. C'est aussi la langue de référence pour plus d'un milliard de musulmans. Comme son nom l'indique, la langue arabe est la langue parlée à l'origine par le peuple arabe.

Le développement de la langue arabe a été associé à la naissance et la diffusion de l'islam. L'arabe s'est imposée, depuis l'époque arabo-musulmane, comme langue religieuse mais plus encore comme langue de l'administration, de la culture et de la pensée, des dictionnaires, des traités des sciences et des techniques. Ce développement s'est accompagné d'une rapide et profonde évolution (en particulier dans la syntaxe et l'enrichissement lexical). L'arabe peut être considéré comme un terme générique rassemblant plusieurs variétés :

- l'arabe classique : la langue du Coran, parlée au VIIe siècle.
- l'arabe standard moderne (l'ASM) : une forme un peu différenciée de l'arabe classique, et qui constitue la langue écrite de tous les pays arabophones. L'ASM reste le langage de la presse, de la littérature et de la correspondance formelle, alors que l'arabe classique appartient au domaine religieux et est pratiqué par les membres des humains.
- les dialectes arabes : malgré l'existence d'une langue commune, chaque pays a développé son propre dialecte. Issus de l'arabe classique, leurs systèmes

grammaticaux respectifs affichent de nettes divergences avec celui de l'ASM. [S.BOULAKNADEL, 08]

III.3 Caractéristique de la langue arabe:

L'Arab**e** est l'un des langages, souvent décrits comme morphologiquement complexe. Composé de 28 lettres (25 consonnes et 3 voyelles longues), les voyelles courtes n'étant pas représentées par des lettres mais par des diacritiques ,placées sur ou sous les consonnes. [S.Zaidi-Ayad,12]

Les textes arabes sont généralement non voyellés, ce qui constitue une grande source d'ambiguïté lexicale.

L'Arabe s'écrit de droite à gauche avec la particularité que les lettres épousent des formes différentes selon qu'elles soient au début, au milieu ou à la fin du mot.[S.Zaidi-Ayad,12]

Comme l'arabe est fortement structuré et très dérivationnel, la compréhension de l'arabe exige le traitement des constituants de cette langue à tous les niveaux: la morphologie, la syntaxe et la sémantique.

Chaque composant nécessite une étude approfondie et une exploitation des caractéristiques linguistiques associées.

Le lexique arabe comprend trois catégories de mots : noms, verbes et particules.

III.3.1 *Structure du mot :*

En arabe un mot peut signifier toute une phrase grâce à sa structure composée qui est une agglutination d'éléments de la grammaire, la représentation suivante schématise une structure possible d'un mot.[F.S. DOUZIDIA, 04]

Post fixe	**Suffixe**	**Corps schématique**	**Préfixe**	**Antéfixe**

- Antéfixes sont des prépositions ou des conjonctions.
- Préfixes et suffixes expriment les traits grammaticaux et indiquent les fonctions : cas du nom, mode du verbe et les modalités (nombre, genre, personne,…)
- Postfixes sont des pronoms personnels.

Exemple : أتتقبلوننا

La segmentation de ce mot donne les constituants suivants :

أ	تـ	تقبل	ونـ	نا

Antéfixe : أ conjonction d'interrogation.

Préfixe : تـ préfixe verbal du temps de l'inaccompli.

Corps schématique: تقبلdérivé de la racine: قبل.

Suffixe : ونـ suffixe verbal exprimant le pluriel.

Post fixe : نَا pronom suffixe complément du nom.

III.3.2 *Catégories de mot:*

L'Arabe considère 3 catégories de mots :

III.3.2.1 *Nom :*

L'élément désignant un être ou un objet qui exprime un sens indépendant du temps.

La déclinaison des noms se fait selon les règles suivantes:

- Le féminin singulier: On ajoute le ,ة exemple جميل devient جميلة.
- Le féminin pluriel : De la même manière, on rajoute pour le pluriel les deux lettres, ات exemple جميلة devient جميلات.
- Le masculin pluriel : Pour le pluriel masculin on rajoute les deux lettres ين ou ون dépendamment de la position du mot dans la phrase (sujet ou complément d'objet), exemple : القائل devient القائلين ou القائلون.
- Le Pluriel irrégulier: Il suit une diversité de règles complexes et dépend du nom. exemple : طبق devient أطباق. [F.S. DOUZIDIA, 04]

III.3.2.2 *Verbe :*

Entité exprimant un sens dépendant du temps, c'est un élément fondamental auquel se rattachent directement ou indirectement les divers mots qui constituent l'ensemble.

La plupart des mots en arabe, dérivent d'un verbe de trois lettres. Chaque verbe est donc la racine d'une famille de mots. Comme en français, le mot en arabe se déduit de la racine en rajoutant des suffixes ou des préfixes.

La conjugaison des verbes dépend de plusieurs facteurs :

- Le temps (accompli, inaccompli).
- Le nombre du sujet (singulier, duel, pluriel).
- Le genre du sujet (masculin, féminin).
- La personne (première, deuxième et troisième).
- Le mode (actif, passif).

La langue arabe dispose de trois temps :

- L'accompli : correspond au passé et se distingue par des suffixes.
- L'inaccompli présent: présente l'action en cours d'accomplissement, ses éléments sont préfixés.
- L'inaccompli futur : correspond à une action qui se déroulera au futur et est marqué par l'antéposition de س ou سوف au verbe. [F.S. DOUZIDIA, 04]

III.3.2.3 *Particules :*

Entités qui servent à situer les événements et les objets par rapport au temps et l'espace, et permettent un enchaînement cohérent du texte. Elles sont principalement les mots outils comme les conjonctions de coordination et de subordination.

Les particules sont classées selon leur sémantique et leur fonction dans la phrase, on en distingue plusieurs types (introduction, explication, conséquence, …). Elles jouent un rôle important dans l'interprétation de la phrase.

Elles servent à situer des faits ou des objets par rapport au temps ou au lieu, elles jouent également un rôle clé dans la cohérence et l'enchaînement d'un texte.

Comme exemple de particules qui désignent un temps منذ, قبل, بعد, un lieu حيث ou de référence الذين … etc [F.S. DOUZIDIA, 04]

III.4 Grammaire de la langue arabe :

La grammaire arabe se divise en deux branches :

III.4.1 *Morphologie, الصرف* :

Elle comprend :

a) Morphologie dérivationnelle, qui étudie la construction des unités lexicales et leur transformation selon le sens voulu. Ainsi, la dérivation morphologique est décrite sur une base morphosémantique : d'une même racine, se dérivent différentes unités lexicales selon des schèmes qui sont des adjonctions et des manipulations de la racine.

b) Morphologie flexionnelle concerne le marquage casuel pour le nom et l'adjectif ou la conjugaison du verbe, appelé **الإعراب.** [S.BOULAKNADEL, 08]

III.4.2 *Syntaxe, النحو* :

Qui étudie la formation correcte des phrases garantit la grammaticalité de la phrase en analysant :

a) La position des unités lexicales les unes par rapport aux autres, déterminant ainsi l'ordre des unités lexicales.

b) Le marquage casuel des unités lexicales de la phrase. Ainsi, la fonction syntaxique de l'unité lexicale est déterminée en s'appuyant sur la morphophonologie. [S.BOULAKNADEL,08]

Une phrase simple de l'Arabe peut être soit nominale ou verbale. Cette classification étant faite en fonction de l'absence ou la présence du verbe dans la phrase.

III.4.2.1 *Phrase nominale :*

La phrase nominale est formée par le rapprochement de deux termes : le sujet (dit **مبتدأ** : inchoactif) et l'attribut (dit **الخبر** : énonciatif). Ce type de phrases sert à exprimer une définition ou énoncer un jugement. La structure de la phrase nominale peut être normale (le sujet précède l'attribut) ou inversée (l'attribut précède le sujet). Le sujet est dans la plupart des

cas un nom déterminé, mais il peut être aussi un pronom personnel démonstratif ou autre.

L'attribut est simple (formé d'une seule expression) ou composé (formé d'une proposition ou d'une simili proposition composée d'une préposition avec son complément). L'attribut simple peut être un nom, un pronom personnel ou démonstratif. [M.DJOUDI, 91]

III.4.2.2 *Phrase verbale :*

On appelle phrase verbale, toute phrase contenant au moins deux éléments: le sujet (dit /فاعل/ : agent) et le verbe (dit /فعل : procès). Cette phrase exprime une action attribuée à un certain sujet, rapportée à un certain temps et dirigée s'il y a lieu, vers un certain objet. Les structures syntaxiques des phrases verbales les plus fréquentes sont Verbe + Sujet, ou bien Verbe + Sujet + Complément. Cependant, il se peut que le sujet précède le verbe. [M.DJOUDI, 91]

III.4.2.3 *Règles d'accord :*

a) Accord du verbe :

L'accord du verbe avec son sujet obéit à un certain nombre de règles:

- ✓ Si le sujet est un nom désignant des êtres humains, deux cas:
  - ➢ Le verbe précède son sujet (cas général); l'accord se fait seulement en genre, jamais en nombre, le verbe restant au singulier.
  - ➢ Le verbe suit son sujet il s'accorde en genre et en nombre.
- ✓ Si le sujet est un nom désignant des animés non humains ou des inanimés, deux cas:
  - ➢ Le sujet est un singulier: le verbe s'accorde en genre.
  - ➢ Le sujet est pluriel, le verbe s'accorde au féminin singulier, qu'il précède ou non son sujet. [M.DJOUDI, 91]

b) Accord de l'adjectif :

Les règles d'accord de l'adjectif sont fonction du nom auquel il se rapporte:

- ✓ Si le nom désigne des êtres humains, l'adjectif s'accorde en genre et en nombre.
- ✓ Si le nom désigne des animés non humains ou des inanimés, l'adjectif s'accorde:
  - ➢ Avec un nom singulier, en genre.
  - ➢ Avec un nom pluriel au féminin singulier [M.DJOUDI, 91]

III.5 Analyse des erreurs grammaticales communes en langue arabe :

Afin d'étudier la possibilité de développer un correcteur grammatical de la langue arabe, nous avons analysé et classé les erreurs grammaticales courantes qui se produisent lors de la formulation d'une phrase MSA dans un style formel.

Ces erreurs ont été vérifiées par des spécialistes Arabes pour qu'elles soient les erreurs grammaticales arabes les plus communes.

Ces erreurs sont représentatives de celles rencontrées par l'utilisateur moyen de traitement de texte lors de la frappe du texte arabe.

Par souci de clarification, des exemples d'erreurs arabes pertinents suivis par leur correction grammaticale sont donnés ainsi que leur classification par type. Pour chaque type d'erreur, un exemple erroné est expliqué dans une phrase arabe agrammaticale.

Type d'erreur	Exemple	Version correcte
Masculine au lieu de féminin	هذا المراجع	هذه المراجع
	حلفت يمينا صادقا	حلفت يمينا صادقة
	أدى اليمين القانوني	أدى اليمين القانونية
	هذا البئر	هذه البئر
	خنصري مجروح	خنصري مجروحة
	هذا اليمين (للقسم)	هذه اليمين
	هذا الريح	هذه الريح
	هذا الفخذ	هذه الفخذ
	هذا السن (للجارحة و العمر)	هذه السن
	هذا الساق	هذه الساق
	هذا الكتف	هذه الكتف
Féminin au lieu de masculin	انتفخت بطني	انتفخ بطني
	قميص جديدة / قديمة	قميص جديد / قديم
	لا بد من حرق الوقود الكافية	لا بد من حرق الوقود الكافي
	ميناء أمينة	ميناء أمين
	هذا الأمر في غاية الوضاحة	هذا الأمر في غاية الوضوح
	لم يثبت لهذا الدواء نجاعة	لم يثبت لهذا الدواء نجوع
	هذه نجمة (بمعنى كوكب)	هذا نجم
	هذه ذقنه	هذا ذقنه
	هذه الرفات	هذا الرفات
Prépositions	يدل إلى	يدل على
	برهنت عن	برهنت على
	التمادي على الباطل	التمادي في الباطل
	تمرست على الأمور	تمرست بالأمور
	أوشك على الموت	أوشك أن يموت
	أوشك على الانتهاء	أوشك أن ينتهي
Accord de genre et nombre entre l'inchoactif et l'énonciatif	الجنود يدافعان عن الوطن	الجنود يدافعون عن الوطن
Accord de genre et nombre entre l'accusatif circonstanciel et le sujet qu'elle modifie	جاءت بعض السيدات تحملن أطفالهن	جاءت بعض السيدات يحملن أطفالهن

Accord de genre, nombre et définition entre l'adjectif et le nom qu'il modifie	الرجال الكريم يساعدون الناس	الرجال الكرماء يساعدون الناس
Accord de genre entre le verbe et le sujet	شرب البنت عصير البرتقال	شربت البنت عصير البرتقال
Pluriel	الموانئ	الموانى
	إطارات	أطر
	بدائل (للأشياء)	أبدال
	بدلات (جمع بَدَل)	أبدال
	جديان / جدايا (جمع جدي)	جداء
	المحلات (جمع المحل)	المَحال
	أمساء (جمع مساء)	أمسية
	خِيطان	أخياط / خيوط
	أدخنة (جمع دخان)	دواخن
	أفمام (جمع فم)	أفواه
	سندات (جمع سند)	أسناد
	صباحات (جمع صباح)	أصْبحة
	صُحْر (جمع صحراء)	صحراوات / صحاري /صحارى
Autres erreurs communes	ثلاث/ أربع / خمس/ ست/ سبع/ ثمان/ تسع/ عشر شهور	ثلاثة/ أربعة / خمسة/ ستة/ سبعة/ ثمانية/ تسعة/ عشرة شهور
	ثلاثة/ أربعة / خمسة/ ستة/ سبعة/ ثمانية/ تسعة/ عشرة نسوة	ثلاث/ أربع / خمس/ ست/ سبع/ ثمان/ تسع/ عشر نسوة
	ثلاثة/ أربعة / خمسة/ ستة/ سبعة/ ثمانية/ تسعة/ عشرة سنين	ثلاث/ أربع / خمس/ ست/ سبع/ ثمان/ تسع/ عشر سنين

Tableau III.1 Les erreurs grammaticales communes de l'Arabe[S.Zaabalaoui,06] [K.Shaalan,05]

III.6 Conclusion :

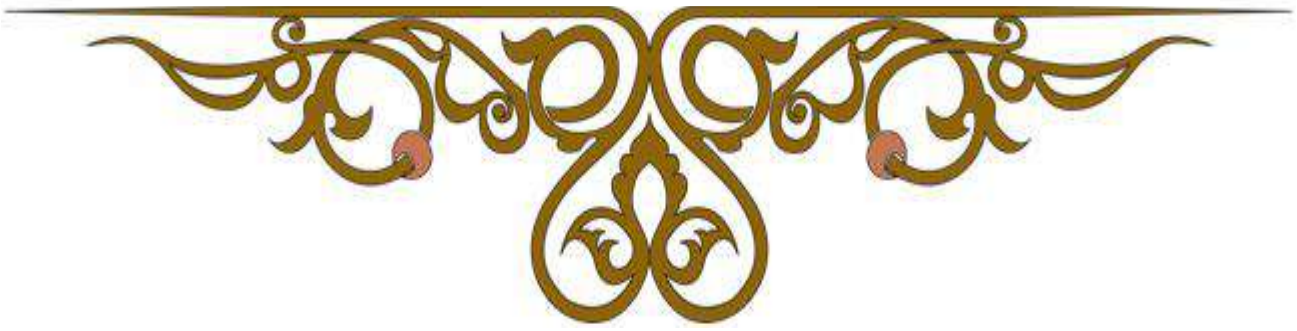
Les erreurs grammaticales présentent un problème délicat, vu que nous ne pouvons les détecter qu'à un niveau très évolué de l'analyse. Pour la langue arabe la collecte d'erreurs augmente encore en raison des caractéristiques de cette langue qui rendent ce genre d'erreurs fréquemment présent. De ce fait, nous avons à traiter des erreurs présentant une difficulté pour les détecter comme pour les corriger.

Dans le but de proposer une méthode valide et performante visant la détection puis la correction des erreurs en langue arabe, il s'avère important et préliminaire de limiter les erreurs communes. Plus encore, il est nécessaire de décrire les différents types de ces erreurs pouvant survenir sous une forme et les classer afin de mieux percevoir le problème.

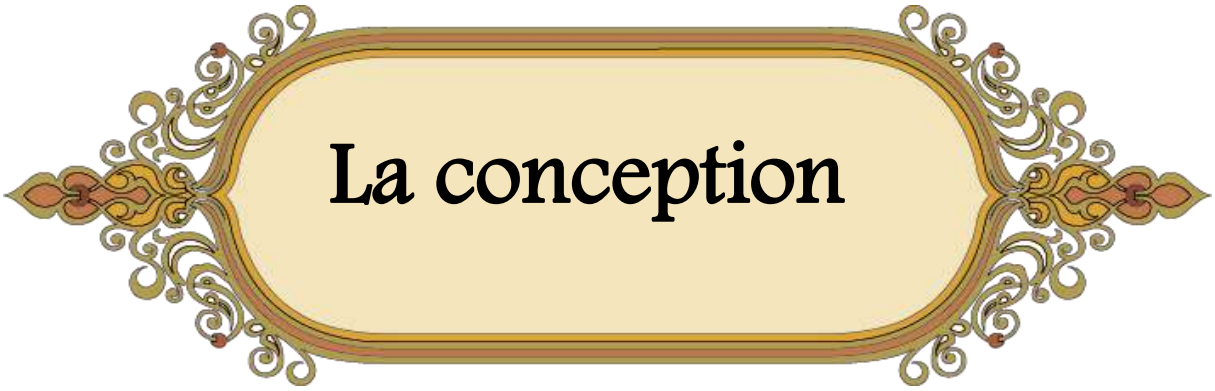
Partie II :



Conception et réalisation



CHAPITRE IV:



IV.1 Introduction :

Comme nous l'avons déjà indiqué, LanguageTool pourrait devenir, après quelques modifications, un correcteur grammatical bien adapté à l'Arabe. C'est pourquoi nous avons décidé de commencer à travailler sur cet outil.

Ce chapitre décrit comment nous avons doté le correcteur des règles nécessaires à la détection des fautes de grammaire dans un texte écrit en arabe.

Tout d'abord, nous présentons la morphologie et L'ambiguïté morphologique dans le traitement automatique de la langue arabe, la segmentation et l'étiquetage morphosyntaxique. Ensuite nous décrivons la conception de notre approche avec certaines catégories d'erreurs dont l'objectif est d'enrichir la base des règles d'erreur, pour ce faire nous expliquons et détaillons les différentes étapes du notre prototype.

IV.2 Morphologie : Principe et Objectif

IV.2.1 *Principe*

La morphologie est un domaine de la langue qui permet la description des règles régissant la structure interne des mots (unités lexicales), chez les grammairiens la morphologie est l'étude des formes des mots (flexion et dérivation), en d'autres termes, la morphologie est l'étude des mots considéré isolément (hors contexte) sous le double aspect de la nature et des variations qu'ils peuvent subir [Hoceini Y,02]. En langue arabe, l'analyse morphologique est d'autant plus importante que les mots sont fortement agglutinés, c'est-à-dire qu'ils sont formés dans leur majorité par assemblage d'unités lexicales et grammaticales élémentaires.

Ainsi Le traitement morphologique est considéré comme une introduction principale à la compréhension globale d'une langue naturelle ; il joue un rôle très important aussi bien du côté linguistique que du côté technique. [Baloul S,03].

IV.2.2 *Objectif*

La plupart des études faites sur la morphologie arabe dans le passé ou ou même de nos jours visent généralement à satisfaire les points suivants:

- L'analyse des mots réellement existant.
- Fournir les données nécessaires aux travaux des différents niveaux de traitement (syntaxe, sémantique et pragmatique). [Hoceini Y,02]

IV.3 Segmentation

C'est la première étape dans le processus de traitement au moyen d'un outil statistique est de la subdiviser en plusieurs unités d'information appelées segments (tokens) qui sont traditionnellement, des mots simples.

La segmentation est étroitement liée à l'analyse morphologique. C'est beaucoup plus avec le cas des langues riches et complexes morphologiquement telle que l'arabe, qui présente un système d'écriture qui combine une écriture segmentée, et une écriture non segmentée. [DILEKH T, 11]

IV.3.1 *Types de segmentation*

Il existe plusieurs niveaux d'analyse signes d'intérêt permettent de repérer les différents éléments constituant le texte et en définir les frontières. On peut s'arrêter au niveau mot graphique, au niveau des unités lexicales ou aller au-delà de celles-ci pour arriver aux unités de base (*les morphèmes*). [DILEKH T, 11]

Selon la visée de l'analyse à entreprendre : lexicale, morphologique ou syntaxique, on peut généralement trouver trois types de segmentation :

- ✓ **Segmentation lexicale** (tokenization) **:** produit des segments lexicaux (tokens).
- ✓ **Segmentation morphologique :** produit des petites unités distinctes appelées morphèmes.
- ✓ **Segmentation syntaxique (chunking) :** produit des unités indépendantes supérieures aux mots, comme les propositions, les syntagmes …etc.[DILEKH T, 11]

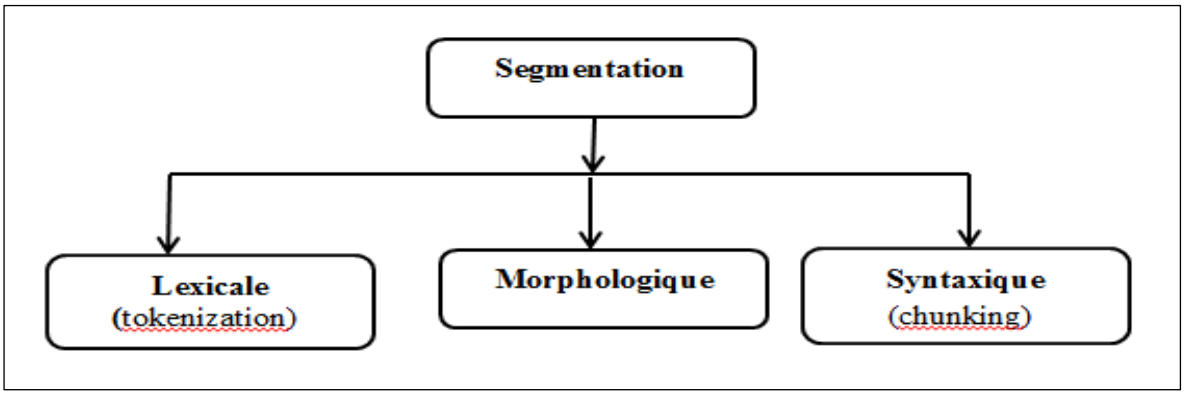


Figure IV.1 : Les types de segmentation

IV.3.2 *Module de segmentation*

Lorsqu'on évoque la segmentation dans le traitement automatique des langues dites naturelles, on parle le plus souvent de trois (03) niveaux de segmentations :

- Segmentation au niveau du texte.
- Segmentation au niveau de la phrase.
- Segmentation au niveau du mot.

Dans notre travail, nous appliquons la segmentation lexicale comme une étape préliminaire requise pour notre traitement ultérieur. C'est une opération consistant à structurer le texte en passant d'un ensemble continu de caractère a une suite discrète d'items lexicaux (segment principaux), en se basant sur les espaces blancs et les signes de ponctuation comme les séparateurs entre les mots , tel que la virgule ',' , le point d'interrogation ' ?' et le point virgule ' ;' . Les verbes et les noms arabes peuvent avoir des segments secondaires. La figure suivante présente un exemple de segmentation :

فعُدْ إلى لغـةِ القـرآنِ صـافيَة ، تَجْلو لكَ الدَّربَ سهْلاً كانَ أو جبـلا.											
فعد	الى	لغة	القرآنِ	صافية	تَجْلو	لكَ	الدَّربَ	سهْلاً	كانَ	أو	جبـلا

Figure IV.2 : exemple d'extraction des segments

IV.4 Etiquetage Morphosyntaxique

- **Qu'est ce que le Tagging?**

Le Tagging (étiquetage ou marquage) est le fait d'assigner une étiquette ou un tag à un mot. Le tag contient des informations morphosyntaxiques sur le mot, c'est-à-dire des informations concernant la forme et la fonction du mot. Elles comprennent notamment la catégorie grammaticale, le genre, le nombre, le temps et le mode.[Vergne J, Giguet E, 98]

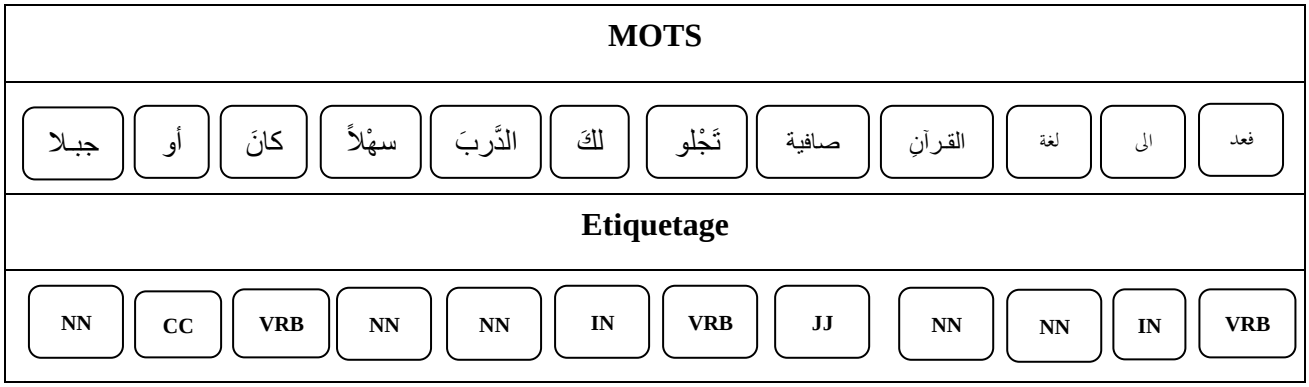


Figure IV.3 : exemple d'étiquetage morphosyntaxique

IV.5 Les différentes étapes/phases du prototype

Pour certaines catégories d'erreurs, on a trouvé qu'on peut extraire deux classes, celles qu'on peut les rassembler dans une même règles (en utilisant les expressions régulières), et celles qu'on ne peut pas. Les deux types de règles seront bien détaillés dans le chapitre suivant.

Voici maintenant un exemple général de comment se fait le démarche de correction pour quelques catégories.

IV.5.1 *Masculin au lieu de féminin*

L'erreur : حلفت يمينا صادقا

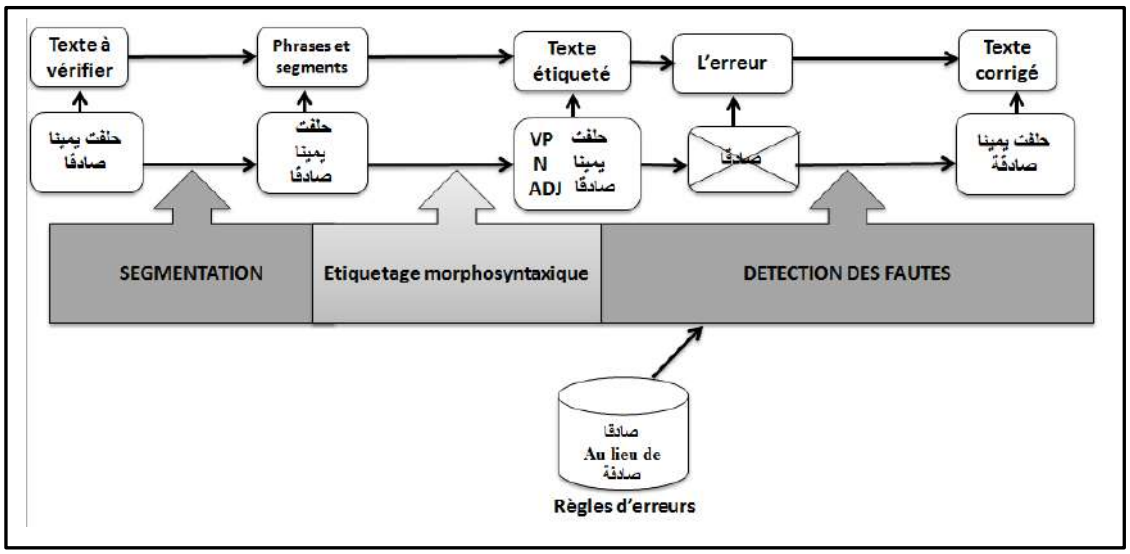


Figure IV.4 : Démarche de correction « catégorie Masculin au lieu de féminin »

Cet exemple se présente dans le but de corriger les erreurs concernant les mots considérées comme des mots masculins et pas féminins.

Le mot «اليمين » qui signifie « الَقسَم» est un nom féminin, mais les écrivains le rendent souvent masculin et disent « حلفت يمينا صادقا» ou « اليمين القانوني». Selon l'histoire de ce mot dans ce sens, et selon les professeurs « DAGHER » et « EL ADNANI », le plus correct est de dire « حلفت يمينا صادقة» et également «اليمين القانونية ». [S.Zaabalaoui, 06]

Le premier traitement consiste à découper la phrase « حلفت يمينا صادقا » en mots, ou plus exactement en tokens حلفت | يمينا | صادقا . C'est la segmentation, ou tokenization, réalisée par le tokenizer.

Un étiquetage morphosyntaxique s'effectue ensuite, chaque mot se voit attribuer l'étiquette (*tag*) correspondante.

حلفت ----- > Verbe au passé (VP)

يمينا------ > Nom singulier (NN)

صادقا ------> Adjectif (JJ)

Une fois l'étiquetage terminé, commence alors l'étape de vérification de la grammaire, le correcteur compare le texte entré au modèle de règle d'erreur de sa base. Donc il propose de remplacer le token صادقا par صادقة.

IV.5.2 *Féminin au lieu de masculin*

L'erreur : هذا الأمر في غاية الوضاحة

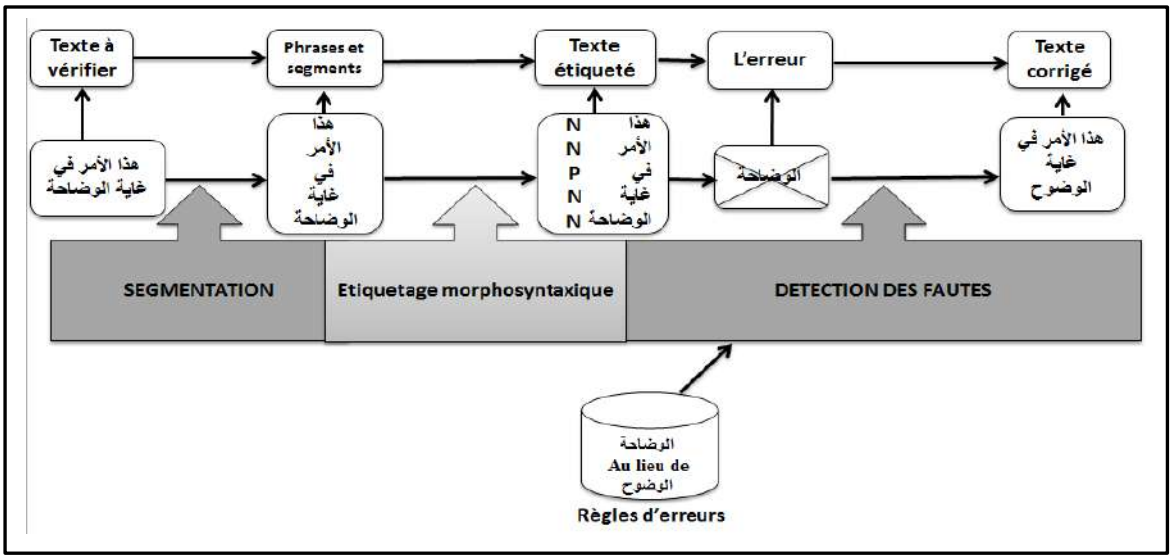


Figure IV.5 : Démarche de correction « catégorie Féminin au lieu de masculin »

Au contraire de l'exemple précédent, cette fois ci, nous montrons comment le correcteur fait le traitement des erreurs concernant les mots considérés comme féminin et non pas masculin.

Le principe reste le même, donc selon la règle, le mot « الوضاحة » sera marqué comme un mot erroné et doit se remplacé avec son équivalent masculin «الوضوح».

IV.5.3 *Prépositions*

L'erreur : برهنت عن

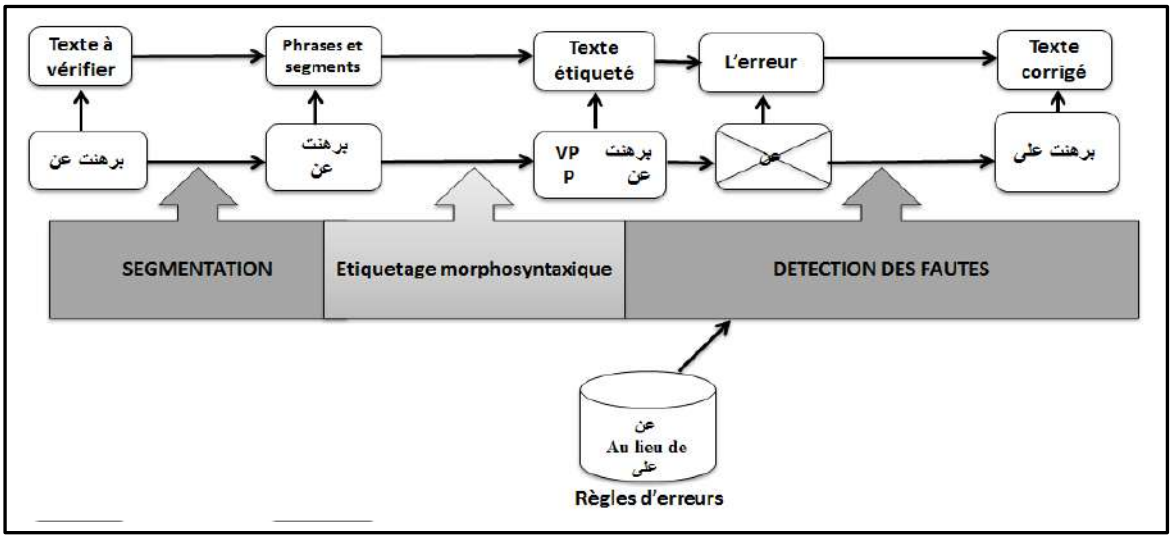


Figure IV.6 : Démarche de correction « catégorie Prépositions »

Le mot « برهان» signifie l'argument et la preuve, et d'après les grammairiens arabes, le verbe « برهن » ne peut pas être suivi par la préposition « عن », donc on ne dit pas « برهنت عن صحة قولي» mais on dit « برهنت على صحة قولي».

L'architecture ci-dessus s'explique comment le correcteur signale une alerte au niveau du token erroné, et par la suite propose la préposition « على» comme une solution d'après ce que la règle indique.

IV.5.4 *Accord de genre entre le verbe et le sujet*

L'erreur : شرب البنت عصير البرتقال

L'architecture ci-dessous concerne le cas d'un verbe au singulier masculin « شرب » avec un sujet singulier féminin.

Donc après les deux premiers étapes, la dernière comme les autres cas, consiste à comparer la syntaxe entrée par celle du corpus, alors le token « شربت » est proposé comme une solution pour que la phrase (le sens) soit correcte.

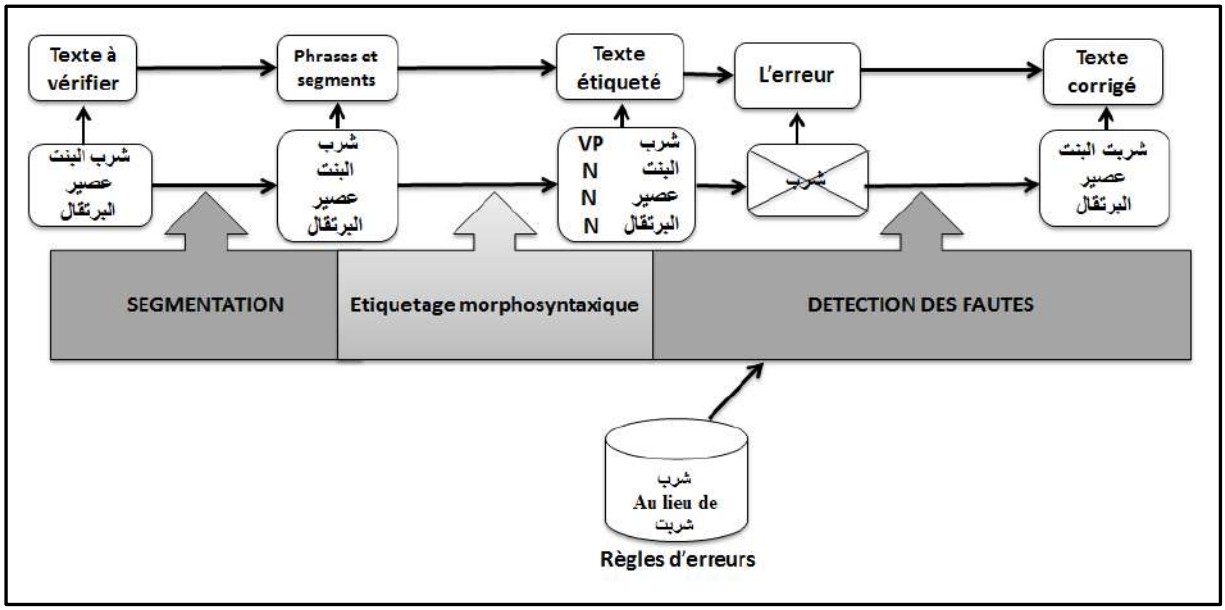


Figure IV.7 : Démarche de correction « catégorie Accord de genre entre V et S »

IV.5.5 *Pluriels*

L'erreur : على جميع الأصعدة

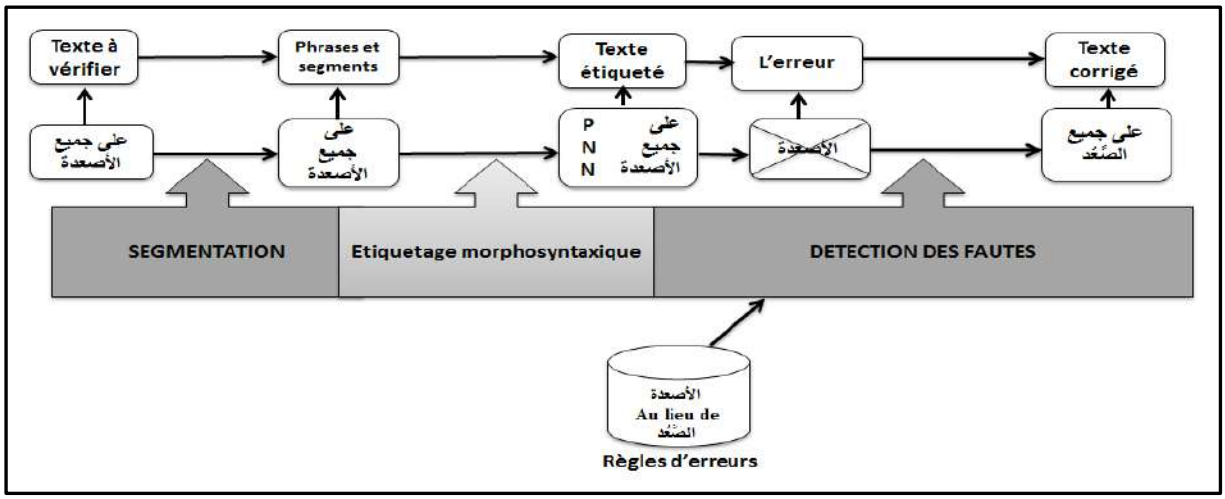


Figure IV.8 : Démarche de correction « catégorie Pluriels »

Cette catégorie contient des exemples hétérogènes dont nous ne pouvons pas les traiter tous ensemble.

Et de même manière, le démarche reste ressemble aux autres cas précédents, mais au niveau de réalisation, chaque règle aura une syntaxe différente.

IV.5.6 *L'accord entre les nombres et les noms qu'ils suivent*

L'erreur : عشرة سنين

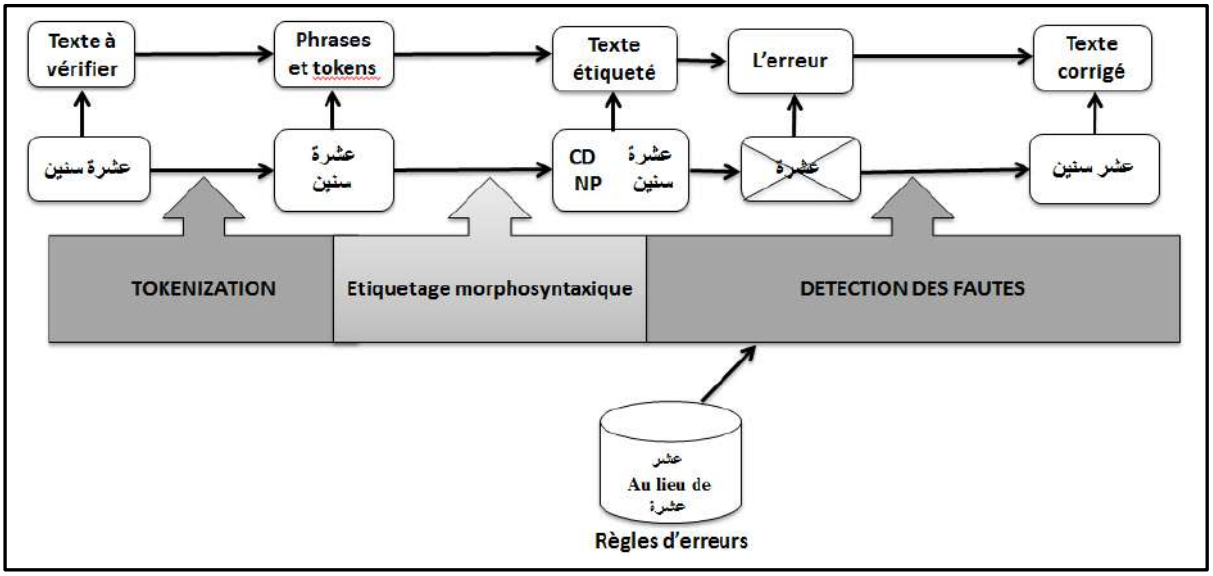


Figure IV.9 : Démarche de correction « catégorie Autres erreurs communes »

Dans ce type d'erreurs, la correction est basée sur la vérification des noms qui suivent les nombres, donc, dans certains cas, le nombre doit être en sa formule féminine, par contre, il doit être en état masculin dans autres cas.

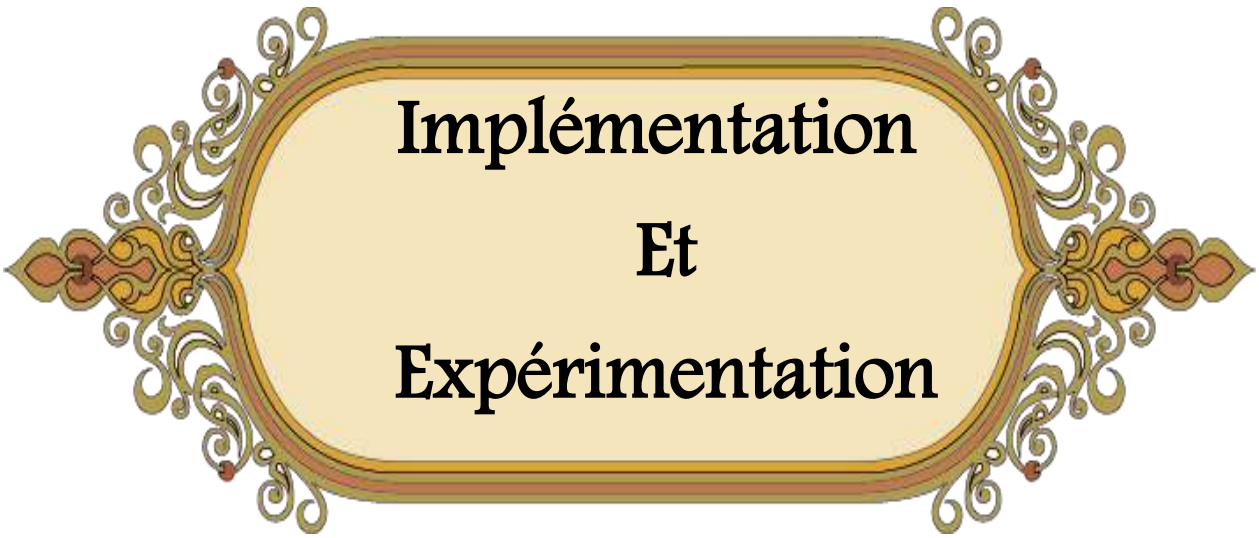
L'exemple ci-dessus explique que si le nombre précède le token «سنين», le correcteur doit proposer la forme correcte du « عشرة» en lui remplaçant par « عشر».

IV.6 Conclusion

La grammaire arabe est une branche de la linguistique contemporaine qui étudie la morphologie et formation des mots, la syntaxe, et la composition des mots en phrases. Dans ce chapitre, après avoir présenté la morphologie et l'ambigüité morphologique, nous avons donné une conception globale de notre approche par l'étude des différentes catégories d'erreurs commune en langue arabe en commençant par la segmentation des phrases en mots (tokens) avant de passer à l'étiquetage de ces tokens et en fin nous passons à la phase de détection des erreurs de grammaire.

Nous consacrons maintenant le chapitre suivant à la réalisation et l'implémentation des différentes procédures de notre projet, ainsi à l'évaluation des résultats obtenus.

CHAPITRE V :



V.1 Introduction

Dans ce chapitre, nous allons décrire les différentes fonctionnalités de notre approche de l'ajout de support d'un outil de correction grammaticale libre pour l'Arabe.

Nous présentons dans ce chapitre l'implémentation, le prototype informatique suivi et l'expérimentation de notre approche, ainsi les résultats d'évaluation de notre application que nous avons développée, en utilisant des scripts java et d'autres technologies.

V.2 Implémentation de l'approche

Dans cette partie, nous décrivons d'abord l'environnement de développement et les outils utilisés. Ensuite, nous citons les principales scripts développées et décrire les résultats de l'application tout en détaillant leurs fonctionnalités.

V.2.1 *Environnement de développement*

Pour implémenter notre approche de l'ajout de support d'un outil de correction grammaticale libre pour l'Arabe, nous avons utilisé le langage de programmation JAVA car la nouvelle version de languagetool est composée de plusieurs classes Java, et aussi nous avons utilisé langage XML.

V.2.1.1 *Langage de programmation JAVA*

Java est une nouvelle technologie conçue par Sun Microsystems l'un des premiers fabricants des stations de travail Sun a été mis au point en 1991. C'est un langage orienté objet crée à base du langage C++. Il est multi plateforme grâce à sa machine virtuelle appelée Java Virtuelle Machine (JVM). Depuis son apparition, java est devenu l'un des piliers de la programmation dans le monde; grâce à l'animation qu'autorisait ce langage et surtout grâce à sa relation directe avec le web. [Sun Microsystems, 95]

Nous avons choisi le langage JAVA parce que la structure actuelle de LanguageTool est composée de plusieurs classes Java.

V.2.1.2 *Langage XML*

Le langage XML (eXtended Markup Language) est un format général de documents orienté texte. Il s'est imposé comme un standard incontournable de l'informatique. Il est aussi

bien utilisé pour le stockage de documents que pour la transmission de données entre applications. Sa simplicité, sa flexibilité et ses possibilités d'extension ont permis de l'adapter à de multiples domaines. De nombreuses technologies se sont développées autour de XML et enrichissent ainsi son environnement.

Le langage XML dérive de SGML (Standard Generalized Markup Language) et de HTML (HyperText Markup Language). Comme ces derniers, il s'agit d'un langage orienté texte et formé de balises qui permettent d'organiser les données de manière structurée. [Olivier Carton, 15]

V.2.1.3 *Environnement de développement ECLIPSE*

Eclipse est un IDE, Integrated Development Environment (EDI environnement de développement intégré en français), c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par IBM, est gratuit et disponible pour la plupart des systèmes d'exploitation. [9]

Nous avons utilisé l'environnement JAVA ECLIPSE version 4.5 (MARS). Il a été développé en juin 2015. Est une new Java fonctionnalités IDE, y compris vue hiérarchique des projets imbriqués, possibilité de personnaliser les perspectives et les améliorations de la vitesse pour la recherche de texte, en plus la version mars a connue l'amélioration du support de Maven, y compris le soutien pour Maven 3.3.3, l'amélioration de Maven archétypes intégration et une meilleure auto-complétion dans l'éditeur de pom. Les rapports d'erreurs automatisée qui permet aux utilisateurs d'Eclipse de signaler directement les erreurs de projets Eclipse.[10]

V.3 Apache Maven

V.3.1 *Définition*

- ✓ Apache Maven est un logiciel de gestion de projet et un outil de construction, Basé sur :
  - le concept d'un modèle d'objet du projet (POM).
  - Le respect de convention et normes standards.
- ✓ Elément central dans la gestion d'une infrastructure de projet informatique :
  - Construction du livrable final.

- Gestion des bibliothèques logicielles de dépendance
- Génération de la documentation et rapport du projet
- Déploiement des versions de livraison sur les plateformes cibles

Maven peut mettre en place le moyen de travailler selon les normes dans un temps très court. Comme la plupart des configurations de projet sont simples et réutilisables, Maven rend la vie du développeur facile, tout en créant des rapports, des contrôles, construire et tester des configurations d'automatisation. [Maxime Gréau, 11]

Qu'est-ce que Archétype?

Archetype est un plugin Maven dont la tâche est de créer une structure de projet selon son modèle.[11]

Qu'est-ce que SNAPSHOTS ?

Est une version spéciale qui indique une copie de développement actuel. Contrairement aux versions régulières, Maven vérifie une nouvelle version de SNAPSHOTS dans un dépôt distant pour chaque construction. [11]

V.3.2 *POM*

POM (**P**roject **O**bject **M**odel) est l'unité fondamentale du travail dans Maven. Il est un fichier XML qui réside dans le répertoire de base du projet **pom.xml**. [11]

Le **POM** contient des informations sur le projet et divers détails de configuration utilisé par Maven pour construire le projet (s). [11]

POM contient également les objectifs et les plugins. Pendant l'exécution d'une tâche ou un objectif, Maven cherche le POM dans le répertoire courant, il lit le POM, obtient les informations de configuration nécessaires, puis exécute le but. [Maxime Gréau, 11]

Il convient de noter qu'il devrait y avoir un fichier POM unique pour chaque projet.

Le POM est considérée comme Descripteur du projet au format XML dont 4 éléments sont obligatoires :

- modelVersion : précise la version du modèle objet utilisé
- groupId : identifie le projet dans un espace de nommage
- artifactId : correspond à la valeur principale contenue dans le nom du fichier de sortie construit par le projet (fichier XML ou archive)

- version : donne la version en cours du projet. [Maxime Gréau, 11]

Le fichier POM.xml de notre projet est basé sur les éléments suivants :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.languagetool</groupId>
        <artifactId>languagetool-parent</artifactId>
        <version>3.5-SNAPSHOT</version>
        <relativePath>../../pom.xml</relativePath>
    </parent>

    <artifactId>language-ar</artifactId>
    <url>http://www.languagetool.org</url>
    <name>Arabic module for LanguageTool</name>
```

Figure V.1 : Fichier pom.xml du projet « language-ar »

V.3.3 *Plugins*

❖ Quels sont Maven Plugins?

Maven est en fait un cadre d'exécution du plugin où chaque tâche est en fait fait par des plugins. Plugins Maven sont généralement utilisés pour:

- Créer un fichier jar
- Créer un fichier War (est un fichier JAR utilisé pour contenir un ensemble de JavaServer Pages, servlets, classes Java, fichiers XML, et des pages web statiques (HTML, JavaScript…), le tout constituant une application web. Cette archive est utilisée pour déployer une application web sur un serveur d'applications).
- Compiler des fichiers de code.
- Les tests unitaires du code.
- Créer de la documentation du projet.
- Créer des rapports de projet. [11]

V.4 Encodage

La langue arabe est encodée suivant plusieurs formats d'encodage comme Unicode, ISO- 8859-6, CP1256, etc.

Le standard Unicode est un mécanisme universel de codage de caractères. Il définit une manière cohérente de coder des textes multilingue et facilite l'échange de données textuelles. Il est obligatoire pour la plupart des protocoles de l'internet, et actuellement supporté par plusieurs systèmes d'exploitation, ainsi que par plusieurs langage de programmation et par des logiciels.

Les caractéristiques de l'arabe vues auparavant constituent des problèmes majeurs face aux technologies informatiques. Pour remédier ces problèmes, le standard Unicode offre toute une panoplie de codes de formatage et des algorithmes permettant, par conséquent, un traitement informatique fiable de l'écriture arabes.

A l'heure actuelle, les données Unicode peuvent être codées sous trois formes principales :

- Une forme codée sur 32 bits (UTF-32).
- Une forme de 16 bits (UTF-16).
- Une forme de 08 bits (UTF-8).

UTF-8 est un codage constitué de suites d'octets ; les bits de poids le plus fort d'un octet indiquent la position de celui-ci dans la suite d'octets. **[**Patrick Andries, 08]

V.5 Présentation de l'application

L'application que nous avons développée, en se basant sur l'intégration de la langue arabe dans l'outil LanguageTool et les solutions proposées lors de la conception présentée dans le chapitre précédent, est composée de son propre fichier pom.xml, et grammar.xml qui est le fichier principal des règles proposées, et d'autre classes java présentant les différentes phases pour atteindre notre but. Nous détaillons par la suite chaque composante et sa fonctionnalité ainsi le résultat obtenu.

V.5.1 *Création du projet et l'ajout du support « ar »*

L'ajout d'une nouvelle langue dans LanguageTool nécessite les étapes suivantes :

1- Extraction du fichier langaugetool-master.zip obtenu d'après la plateforme github.

2- Déplacement vers le répertoire languagetool-master/languagetool-language-modules.

3- Création du nouveau projet maven « ar » en utilisant la commande suivante :

mvn archetype :generate –DgroupId=org.languagetool –DartifactId=ar –DarchetypeArtifactId=maven-archetype-quickstart –DinteractiveMode=false

Le terme –DartifactId=ar est choisi selon le code **ISO 639-1**.

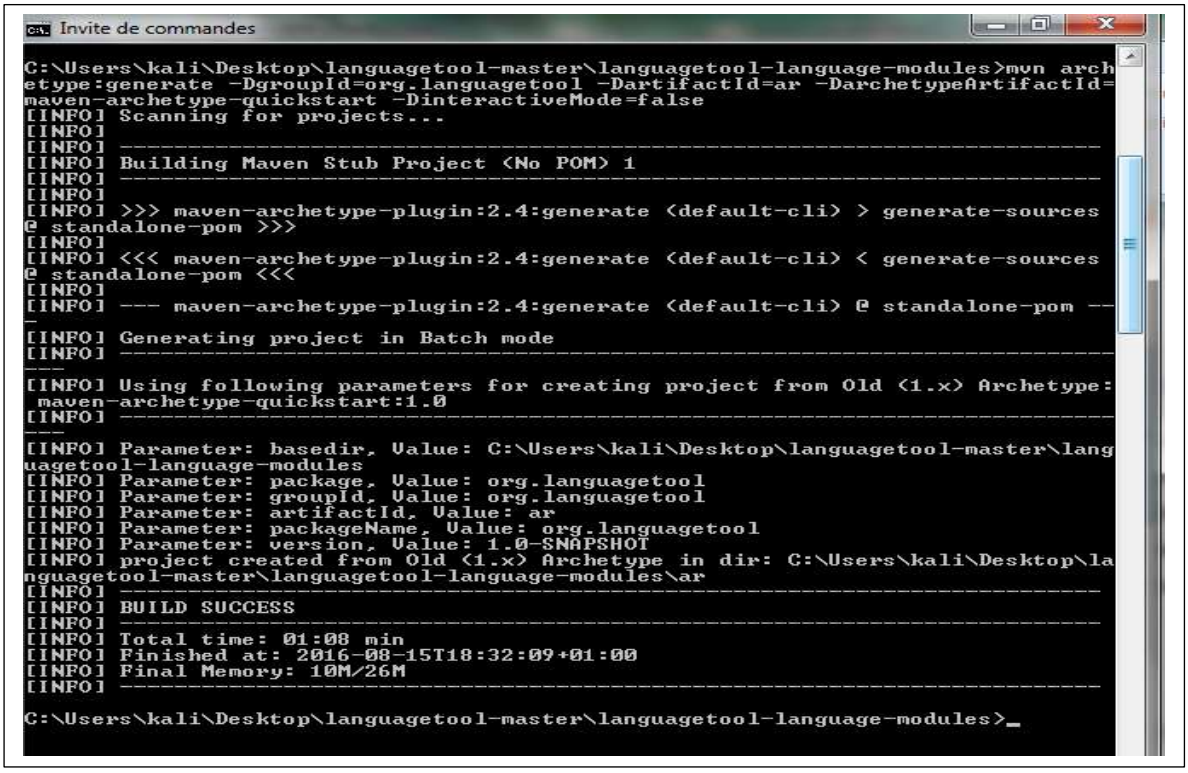


Figure V.2 : Création du projet « ar » BUILD SUCCESS

Donc un dossier « ar » est créé dans le répertoire précédent contenant un fichier pom.xml que nous allons adapter « Arabic module for LanguageTool » comme « name » de ce projet et « language-ar » comme son « Artifact Id ».

A ce niveau-là on doit créer quelques fichiers.

V.5.2 *Création des fichiers nécessaires java et xml*

V.5.2.1 *Arabic.java*

Il définit la classe « Arabic» contenant les principaux méthodes dans ce projet tel que :

```
public class Arabic extends Language implements AutoCloseable {
@Override
  public String getName() {
    return "Arabic";
  }

@Override
  public String getShortName() {
    return "ar";
  }
@Override
  public String[] getCountries() {
    return new String[]{"DZ","EG","TN","SA"};
  }
@Override
  public SentenceTokenizer getSentenceTokenizer() {
    if (sentenceTokenizer == null) {
      sentenceTokenizer = new SRXSentenceTokenizer(this);
    }
    return sentenceTokenizer;
  }
@Override
  public WordTokenizer getWordTokenizer() {
    if (wordTokenizer == null) {
      wordTokenizer = new ArabicWordTokenizer();
    }
    return wordTokenizer;
  }
@Override
  public Tagger getTagger() {
    if (tagger == null) {
      tagger = new ArabicTagger();
    }
}
    return tagger;
}
```

Figure V.3: Arabic.java

V.5.2.2 *language-module.properties*

Ce fichier doit pointer vers notre classe « Arabic », il sera chargé par LanguageTool lors de l'exécution pour détecter la langue (Arabe) prise en charge.

```
languageClasses=_org.languagetool.language.Arabic
```

V.5.2.3 *grammar.xml*

Le fichier principal des règles, un formalisme en XML, il décrit des motifs linguistiques en utilisant le vocabulaire suivant :

.5.2.3.1 *Element <rule>*

Cet élément constitue le niveau le plus élevé de la hiérarchie, et englobe tous les autres. Il permet de délimiter chaque règle, et donc de décrire chaque motif, le traitement à effectuer, et les messages à retourner à l'utilisateur. Il possède 2 attributs, ID et name, dont la fonction est de définir respectivement un identifiant unique et un nom pour la règle. [A. Souque, 07]

Dans le formalisme XML, l'élément <rule> se présente ainsi :

```
<rule id="ReplaceDeterminerF" name="هذا (هذه) ">
      ...
</rule>
```

Entre les balises <rule></rule>, nous avons ensuite les éléments du niveau inférieur : pattern, message, example.

V.5.2.3.2 *Element <example>*

Il s'agit d'un élément servant à donner des exemples d'énoncés corrects ou incorrects, en rapport avec l'erreur détectée par le motif de la règle. Il contient un attribut type indiquant le type d'exemple qu'il donne et qui a donc pour valeur soit "correct" soit "incorrect".

L'exemple peut également contenir un ou plusieurs autres éléments <marker>, qui permettent, lors de l'affichage, de mettre en évidence le ou les mots que les balises de cet élément encadrent. [A. Souque, 07]

```
<rule>
      ...
      <example correction="القانونية"><marker>القانوني</marker></example>
      <example>جميع الرؤساء يؤدون اليمين القانونية</example>
</rule>
```

V.5.2.3.3 *Element <message>*

Cet élément contient un message pour expliquer à l'utilisateur quelle faute a été détectée. Il n'a pas d'attribut. En revanche il peut contenir des éléments <suggestion>, dont les balises entourent des mots qui seront suggérés en correction. [A. Souque, 07]

```
<rule>
      ...
<message>صيغة الجمع غير صحيحة:هو دليل جمع ..<suggestion>أدلة</suggestion></message>
</rule>
```

V.5.2.3.4 *Element <pattern>*

C'est le dernier élément de <rule> que nous présentons. Il délimite la description du motif. Il est composé d'un ou de plusieurs éléments <token>.[A. Souque, 07]

```
<rule>
 <pattern>
                ...
   </pattern>
   <message>صيغة الجمع غير صحيحة:هو محل جمع: <suggestion>محال</suggestion>؟</message>
   <example correction="محال"><marker>محلات</marker></example>
</rule>
```

Dans les balises <pattern> </pattern>, nous trouvons l'élément du niveau inférieur : <token>.

V.5.2.3.5 *Element <token>*

Les tokens du motif sont décrits dans autant d'éléments <token>. Chacun contient plusieurs attributs facultatifs : [A. Souque, 07]

- **regexp** : signifie expression régulière (regular expression). Cet attribut, s'il a la valeur "yes", permet de mettre plusieurs mots dans un même <token>, en les séparant avec la barre verticale "|",[A. Souque, 07] , pour les traiter similairement comme :

```
<rule  >
   <pattern>
      <token regexp="yes">الكتف | الفخذ | الساق | الريح | البئر | المراجع</token>
   </pattern>
                  ...
</rule>
```

- **skip** : est un attribut qui spécifie un nombre de tokens pouvant être ignorés, avant de trouver le token suivant dans le motif. skip="2" permet de sauter 0, 1 ou 2 tokens, entre l'élément <token> qui possède cet attribut, et celui qui suit dans la description du motif.

[A.Souque, 07]

```
<rule id="G_N_C_A_S" name="تحملن (يحملن)">
  <pattern>
      <token skip="1">جاءت</token>
      <token>السيدات</token>
            ...
   </pattern>
             ...
</rule>
```

- **negate** : lorsque sa valeur est "yes", indique que le token ne doit pas faire partie du motif.[A. Souque, 07]

```
<rule>
      <pattern>
            .  .  .
            <marker><token negate ="yes"> على</token></marker>
      </pattern>
      .  .  .
</rule>
```

Ce formalisme fonctionne selon un système, dont la mesure où le principe est toujours de rechercher, dans le texte, des motifs identiques à ceux décrits dans les règles.

V.5.3 *Integration de l'Arabe et résultat*

Après avoir terminé les étapes précédentes, il nous reste que intégrer la langue arabe à des fichiers systèmes et finalement faire de l'exécution :

- Ajouter l'entrée « ar = Arabic » au fichier « MessagesBundle.properties » qui contient chaque abréviation avec son indication.
- Ajouter l'Arabe comme une dépendance dans le fichier « languagetool-language-modules/all/pom.xml ».

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.languagetool</groupId>
        <artifactId>languagetool-parent</artifactId>
        <version>3.5-SNAPSHOT</version>
        <relativePath>../../pom.xml</relativePath>
    </parent>
    <artifactId>language-all</artifactId>
    <url>http://www.languagetool.org</url>
    <name>All language modules for LanguageTool</name>
    <description>Meta project that has dependencies on all language modules for
LanguageTool</description>
<dependencies>
         ...
        <dependency>
            <groupId>org.languagetool</groupId>
            <artifactId>language-ar</artifactId>
            <version>${languagetool.version}</version>
        </dependency>
       ...
</dependencies>
```

- Ajouter <module>languagetool-language-modules/ar</module> au fichier «pom.xml» du grand projet « LanguageTool-master ».
- Finalement on exécute la commande « mvn clean package » sur le répertoire du projet « ar ». Le résultat se présente comme un fichier jar et doit maintenant supporter notre nouvelle langue « Arabe Algerie».



Figure V.4 : Interface de LanguageTool « Arabic Algérie »

V.6 Développement des règles d'erreurs

Comme nous avons dit dans le chapitre précèdent certains catégories d'erreurs peuvent comporter 2 types de règles selon la relation entre les fautes.

Nous avons rédigé quelques règles en fonction des erreurs que nous avons observées dans le corpus (Tableau des erreurs) et que nous pensions pouvoir traiter. Pour quelques-unes, nous donnons en exemple le patron de la faute, donc la partie pattern de la règle, telle que nous l'avons définie pour que celle-ci fonctionne comme nous le souhaitions.

V.6.1 *Masculin au lieu de féminin*

L'exemple suivant montre comment l'utilisation des expressions régulières nous permet de regrouper plusieurs règles en une seule.

```
<rule id="ReplaceDeterminerF" name="هذا (هذه)">

      <pattern>
            <marker>
                  <token>هذا</token>
            </marker>
            <token regexp="yes">الكتف|الفخذ|الساق|الريح|البئر|المراجع</token>
      </pattern>
      <message>أتقصد هذه <suggestion></suggestion>؟ الإسم مؤنث..</message>
      <example correction="هذه"> <marker>هذا</marker></example>

</rule>
```

Le 2ème token contient plusieurs mots qu'on doit changer leur déterminant « هذا » par « هذه ».Donc l'expression régulière sert à réduire le travail en rassemblant les mots dans une seule balise <token>.

La balise <marker> fait un trait sous les tokens qu'elle contient lors de l'exécution, généralement sont les tokens erronés qu'on doit les remplacer avec ses corrections.

Cette règle fait pouvoir le correcteur d'afficher un message descriptif du motif d'erreur, plus une suggestion et enfin un exemple de type « correct » pour remplacer les fautes lorsque le clic droit sur le token marqué.

- **Résultat :**



Figure V.5 : Résultat de la règle «هذا (هذه) المراجع »

V.6.2 *Féminin au lieu de masculin*

```
<rulegroup id="NounMasculin" name="الإسم مذكر ">
      <rule>
            <pattern>
                  <token>وقود</token>
                  <marker><token>كافية</token></marker>
            </pattern>
            <message>أتقصد <suggestion>كافي</suggestion>؟</message>
            <example correction="كافي"><marker>كافية</marker></example>
            <example>لا بد من حرق وقود كافي</example>
      </rule>
. . .
```

Plusieurs écrivains et locuteurs de la langue arabe pensent que le mot «وقود » est un nom féminin, par contre, les grammairiens arabes le classifient un nom masculin, et comme l'adjectif doit compatible avec le nom qui la précède nous consacrons cet exemple pour faire corriger l'erreur connue « وقود كافية ».

Le principe est basé sur le marquage du token qui rend la phrase agrammaticale et lui remplace par le mot correct, plus un exemple descriptif qui fait éclaircir la règle.

- **Résultat :**

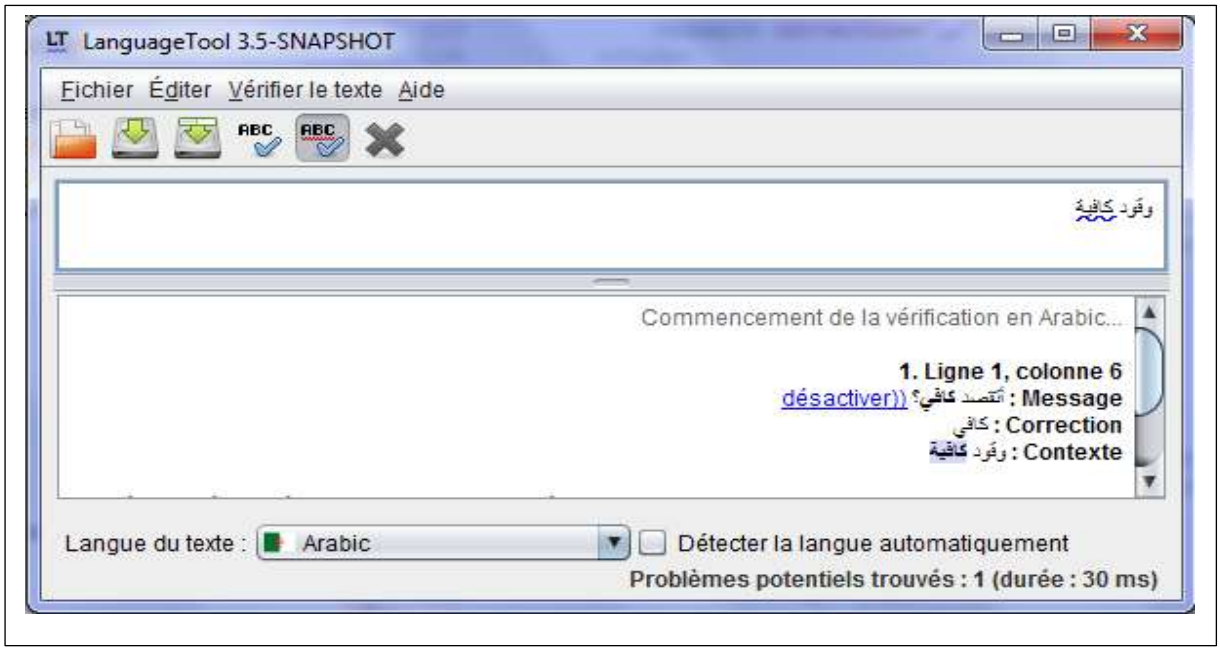


Figure V.6 : Résultat de la règle «وقود كافية (كافي)» -1



Figure V.7 : Résultat de la règle «وقود كافية (كافي)» -2

V.6.3 *Prépositions*

```
<rulegroup id="RepPrep" name="حروف الجر ">
<rule>
      <pattern>
            <token >برهنت</token>
            <marker><token negate ="yes"> على</token></marker>
      </pattern>
      <message>أتقصد<suggestion>على</suggestion></message>
      <example correction="على"><marker>/2</marker></example>
</rule>
```

Nous avons déjà cité dans les chapitres précédents que le verbe « برهنت» compose une fausse syntaxe lors de la combinaison avec la préposition « عن». Donc au cas normal, nous pouvons faire ça par une règle simple, mais nous avons choisi d'intégrer l'attribut « negate » dans ce cas pour expliquer un peu son rôle.

Alors, la balise <marker> sert à faire un trait sous le token qu'elle contient sauf le token «على» puisqu'il ne fait pas partie du motif, c'est comme un token négligeable.

Un problème reste maintenant, c'est comment marquer le mot erroné dans le contexte ?, « /* » permet d'insérer le token n° * dont « * » signifit le nombre du token {1,2,3..}, donc si on met « /1/3 » le resultat s'affiche « token1 token3 ».

- **Résultat**

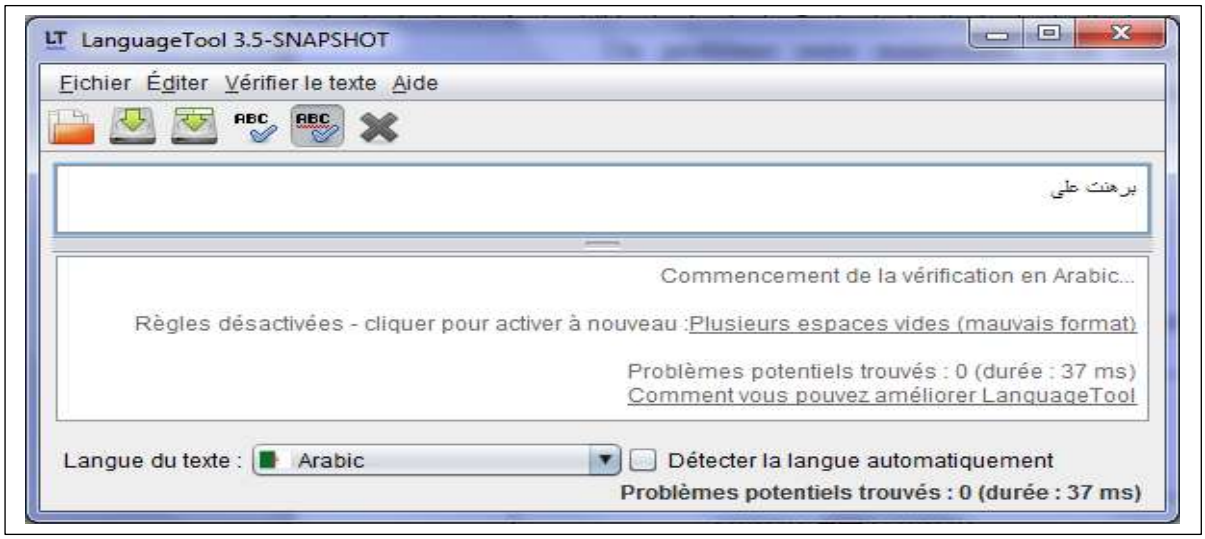


Figure V.8 : Résultat de la règle « برهنت عن (على)» -1

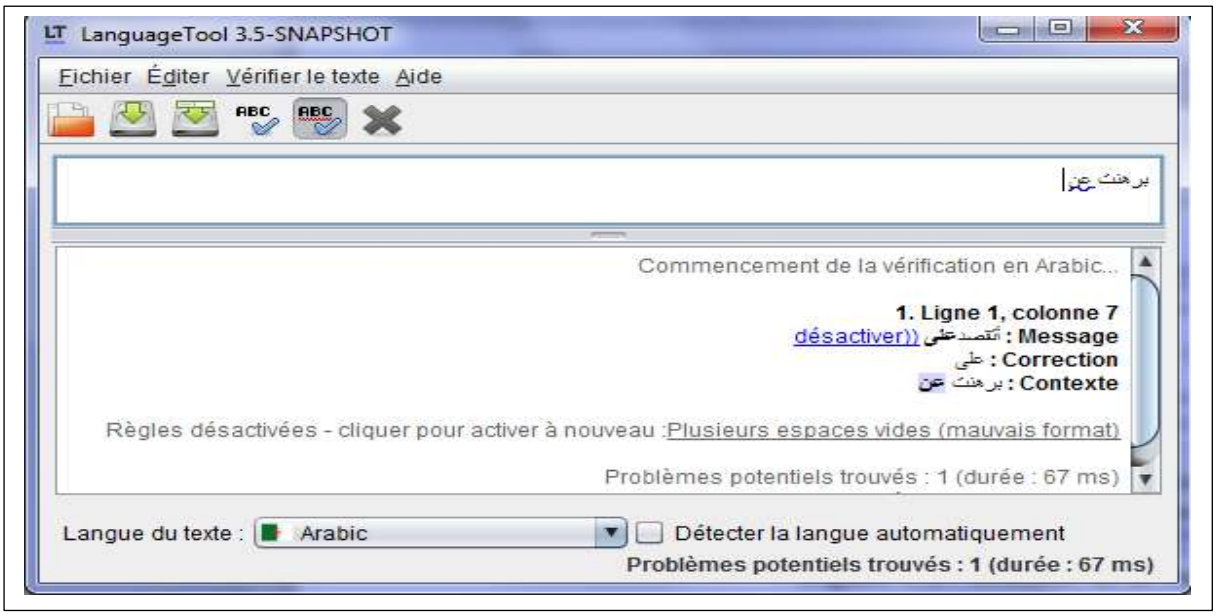


Figure V.9 : Résultat de la règle « برهنت عن (على)» -2

V.6.4 *Accord de genre et nombre entre l'accusatif circonstanciel et le sujet qu'elle modifie*

```
<rule id="G_N_C_A_S" name="(تحملن) (يحملن)">
      <pattern>
            <token skip="1">جاءت</token>
            <token>السيدات</token>
            <marker><token>تحملن</token></marker>
            <token>أطفالهن</token>
      </pattern>
      <message>أتقصد <suggestion>يحملن</suggestion>؟</message>
      <example correction="يحملن"> <marker>تحملن</marker></example>

</rule>
```

Le but dans cet exemple est de faire l'accord entre l'accusatif circonstanciel et le sujet qu'elle modifie, le principe reste le même c'est faire rechercher, dans le texte, des motifs identiques à ceux décrits dans les règles.

Mais qu'est qu'il va se passe si on fait un petit changement dans le texte entré, «skip="1"» nous permet d'ajouter un seul token entre «جاءت » et « السيدات» sans signaler une erreur de syntaxe.

- **Résultat**



Figure V.10 : Résultat de la règle «جاءت السيدات تحملن (يحملن) أطفالهن»

V.7 Conclusion et perspective

La correction grammaticale est une fonctionnalité très demandée par les utilisateurs de traitement de texte.

Nous avons effectué un travail de recherche très intéressant sur la correction grammaticale et ses outils. Nous avons adapté la langue arabe à l'utile language tool et avons proposé des solutions permettant d'améliorer la détection des fautes par l'étude de certaines erreurs communes.

Ce travail est poursuivre dans ce domaine, afin de développer un nouveau moteur de correction adapté à l'interprétation de règles.

Bien sûr, la détection des erreurs est pour le moment loin d'être parfaite, si tant est qu'elle puisse l'être. Il ne s'agit là que d'un premier correcteur, qui requiert encore beaucoup de travail pour être réellement efficace.

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.

L'évaluation d'un système est une étape cruciale de son développement, car elle permet de mettre en évidence les points forts et les limites de ce système, afin de dégager des pistes pour l'améliorer. Elle permet également de comparer ce système avec des approches concurrentes.

Proposition de formalisme

Pour améliorer la correction grammaticale, et en particulier pour prendre en compte la fonction d'unification, un interpréteur compatible avec ce formalisme est nécessaire. Le développement d'un tel outil consistera alors également à compléter la base de règles. Dans ce but, il faut réaliser, de manière approfondie, l'analyse d'un corpus d'erreurs constitué. Nous pourrons alors en extraire des fautes, dont nous définirons la typologie, et à partir desquelles nous créerons de nouvelles règles.

Nouvelle structure

Il nous faut donc à présent penser au développement d'un moteur d'inférence adapté, capable de fonctionner. Nous avons alors deux possibilités. La première est de partir de la version actuelle de LanguageTool comme base, et de la transformer selon nos besoins. La seconde est de concevoir un outil entièrement nouveau. [A. Souque, 07]

Quelle que soit la solution adoptée, l'outil aura la structure suivante, schématisée comme suite :

➔ Tokenization: segmentation du texte en phrases, puis en tokens.

➔ Tagging: étiquetage, correction orthographique, puis désambiguïsation

- Les tokens seront étiquetés en XML.
- L'utilisateur sera invité à corriger, au fur et à mesure, les éventuelles fautes d'orthographe dans les mots inconnus, qui pourront alors recevoir un tag.
- Les tokens seront ensuite désambiguïsés, avec une méthode à base de règles.

➔ Chunking: délimitation des chunks au sein des phrases.

➔ Détection des fautes: succession de plusieurs types de détection.

- La substitution d'un mot par un autre, comme une confusion d'homophones ou une faute de frappe, aboutit souvent à la présence d'un mot d'une catégorie grammaticale inattendue. Certaines de ces erreurs pourront être détectées, puis corrigées par l'utilisateur. Un retour à l'étape du tagging permettra ensuite de rectifier l'étiquetage, ainsi que la segmentation en chunk.
- Les erreurs d'accords au sein des chunks seront signalées à l'utilisateur qui pourra les corriger au fur et à mesure. Les traits de ces chunks seront alors rectifiés.
- Les accords erronés entre chunks pourront enfin être détectés, signalés, et corrigés. [A. Souque, 07]

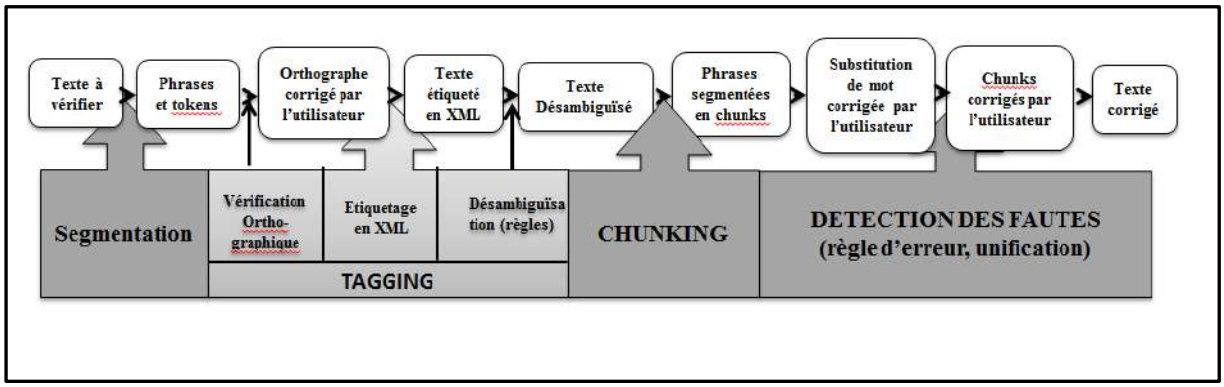


Figure V.11 : Proposition d'une structure pour un nouvel outil [A. Souque, 07]

La principale amélioration sera bien sûr de pouvoir corriger les très fréquentes erreurs d'accords, grâce à l'utilisation combinée d'étiquettes en XML, d'une segmentation en chunks, et du principe de l'unification. Cette méthode de détection permettra de corriger toute faute d'accord, quel que soit son contexte, sans nécessite d'interminables listes énumératives de règles. [A. Souque, 07]



Conclusion générale



Conclusion générale

Le correcteur grammatical est sans doute le logiciel avec lequel l'utilisateur a le plus de rapports "passionnels" donc constitue un outil de base pour tout utilisateur de traitement de texte. Utilisant généralement les techniques de développement les plus innovantes, il peut être considéré comme le logiciel grand public appartenant le plus au vaste domaine de l'intelligence artificielle.

La correction des fautes suppose que la fonction grammaticale de chacun des mots de la phrase ait pu être déterminée afin qu'en appliquant l'ensemble des règles d'accord on puisse mettre en valeur et corriger ces erreurs.

Notre projet s'insère dans le cadre de travail sur les logiciels open source dont le code source est accessible au public et que les utilisateurs peuvent utiliser tel quel ou modifier. Notre objectif est donc l'intégration de la langue arabe dans ce type de logiciel qui est LanguageTool pour la correction grammaticale de cette langue.

Dans notre projet nous avons proposé une démarche qui permette la vérification grammaticale automatique des erreurs communes de la langue arabe. Cette démarche a été basée sur les ressources linguistiques.

Tout d'abord, nous avons à travers une recherche étudiée en détail les différentes phases de cette démarche comme la segmentation, l'étiquetage morphosyntaxique et en fin la détection des erreurs qui basée sur des règles d'erreurs. Le résultat de cette recherche, nous a permis de dessiner un schéma pour la conception de notre approche de vérification. Finalement nous avons implémentée notre stratégie qui combine tous ces phases de vérification grammaticale.

Nous avons conçu et réalisé un prototype d'un outil qui implémente les méthodes citées. Cependant, ce prototype est en phase de développement et en test.


لغة أُودِعَ في أصدافها
من قوانين الهدى أنهى دُرَر
أقلمٍ نُسِجَ على منوالها
كلم التنزيل في أسمى سور

Glossaire

Chunk (Abney, 1991), aussi appelé syntagme minimal (Vergne, 1999) : Syntagme constitué d'un groupe de mots fonctionnels (déterminants, pronoms, adjectifs,etc.) réunis autour d'un mot lexical (nom ou verbe le plus souvent).

Unité de calcul au sein d'une phrase, délimitée par des mots grammaticaux, ponctuations ou marques morphologiques.

Chunker (segmenter en chunks) Outil permettant de calculer automatiquement les chunks dans un texte.

Chunking (segmentation en chunks) Découpage des phrases en chunks.

Désambiguïsation Réduction du nombre d'étiquettes (ou tags) possédées par un mot (ou token) ambigu, c'est-à-dire pouvant avoir plusieurs catégories grammaticales ou sous-catégories possibles.

Cette réduction se fait à l'aide de règles statistiques ou manuelles. Les règles statistiques consistent à appliquer aux mots ambigus les probabilités qu'ils aient tel ou tel tag, et à ne conserver que le ou les tags les plus probables.

Les probabilités sont calculées préalablement à partir d'un corpus d'apprentissage.

Les règles manuelles se fondent sur le contexte immédiat des mots ambigus et décrivent des modèles de contextes en présence desquels tels mots prennent tel (s) tag(s).

LanguageTool utilise des règles manuelles.

Étiquetage morphosyntaxique Fait d'attribuer à chaque forme graphique (token) du texte la ou les étiquette(s) morphosyntaxique(s) (tag) qui lui correspond(ent).

L'étiquetage se fait à partir d'un lexique contenant tous les mots de la langue et les informations sur les catégories et sous-catégories de chacun.

Expression régulière Suite de caractères typographiques permettant de définir un modèle de chaînes de caractères et de trouver les chaînes correspondantes dans un bloc de texte.

GRAC (grammar checker) un correcteur indépendant de toute langue, développé par Maxime.Biais, il utilise la méthode probabiliste pour la désambiguïsation et des règles de grammaire pour le détection des fautes.

Pattern (motif, patron) Dans ce mémoire, le pattern est un élément contenu dans les règles qui décrit des combinaisons possibles de mots. Dans les règles de correction, les patterns décrivent des modèles d'erreurs.

La description dans les patterns se fait généralement à l'aide d'expressions régulières.

Tag (étiquette) Étiquette attribuée à chaque token du texte et contenant les informations morphosyntaxiques de ce token, c'est-à-dire sa catégorie grammaticale (nom, verbe, particule, etc.) et ses sous-catégories (genre, nombre, temps, personne, etc.).

Tagset (jeu d'étiquettes) Liste de toutes les étiquettes utilisables par l'étiqueteur morphosyntaxique (tagger).

Token (jeton) Unité issue de la tokenization. Il s'agit généralement d'un mot, ou plutôt d'une forme graphique. En effet, le mot en linguistique peut désigner une unité sémantique constituée de plusieurs formes graphiques.

Tokenization (segmentation en jetons) Segmentation du texte en phrases puis en tokens. Le découpage s'effectue au niveau des séparateurs, constitués des espaces et des ponctuations.

Tokenizer (segmenteur en jetons) Outil permettant de segmenter automatiquement le texte en phrases puis en tokens.

Bibliographie

[A. Souque, 07] Agnès Souque, UNIVERSITÉ STENDHAL – GRENOBLE 3, LABORATOIRE LIDILEM, CONCEPTION ET DÉVELOPPEMENT D'UN FORMALISME DE CORRECTION GRAMMATICALE AUTOMATIQUE, Juin 2007

[A. Souque, 14] Agnès Souque. Modèle de vérification grammaticale automatique gauche-droite. Linguistique, Université Grenoble Alpes, 2014.

[A.SOUQUE, 16] Agnès Souque. Modèle de vérification grammaticale automatique gauche-droite. Linguistique. Université Grenoble Alpes, 4 Jan 2016, Français.

[B.SORENSEN, 07] Bjarne SORENSEN, ERP Open Source ou Commercial, Genève, 7 Dec 2007

[Baloul S,03] Baloul S ,Développement d'un système automatique de synthèse de la parole à partir du texte arabe standard voyellé, thèse de doctorat, université du Mans, (2003).

[CHARNET & PANCKHURST, 98] Chantal CHARNET, Rachel PANCKHURST, Le correcteur grammatical : un auxiliaire efficace pour l'enseignant ? Quelques éléments de réflexion, ALSIC (Apprentissage des Langues et Systèmes d'Information et de Communication), Université Paul Valéry, Montpellier III, France 1998

[C. Jacquet-Pfau, 01] [C. Jacquet-Pfau, 01] Jacquet-Pfau Christine, « Correcteurs orthographiques et grammaticaux. Quel(s) outil(s) pour quel rédacteur ?», Revue française de linguistique appliquée Fev/2001 (Vol. VI) , p. 81-94

[Clément & Kim & Marlet,09] Lionel Clément(Université Bordeaux 1, LaBR), Kim Gerdes(ILPGA, LPP, Sorbonne Nouvell), Renaud Marlet(INRIA, LaBRI), Grammaires d'erreur – correction grammaticale avec analyse profonde et proposition de corrections minimales, TALN 2009 – Session posters, Senlis, 24–26 juin 2009.

[DILEKH T, 11] DILEKH Tahar, Implémentation d'un outil d'indexation et de recherche des textes en arabe, 28 /09 /2011.

[F.S. DOUZIDIA, 04] Fouad Soufiane Douzidia, Résumé automatique de texte arabe, Université de Montréal, Septembre, 2004.

[G. Perrier, 14] GUY Perrier, Initiation au traitement automatique des langues, 26 DECEMBRE 2014.

[H. SOUSSI] SOUSSI HAYMEN, cours informatique, Le PROGRAMME INFORMATIQUE

[Hoceini Y,02] Hoceini Y ,Un système d'analyse morphologique de la langue arabe, mémoire magister, école nationale supérieure d'informatique, (2002).

[K.Shaalan, 05] Khaled F. Shaalan, Arabic GramCheck:a grammar checker for Arabic, Institute of Informatics, The British University in Dubai (BUID), P.O. Box 502216,Dubai, United Arab Emirates, 2005

[K. Ronda, 14] Kenza Ronda , Logiciels libres et ERPs Open Source : maturité des ERPs Open Source et adéquation dans une PME ou TPE, 21 Jan 2014.

[M.DJOUDI, 91] Contribution à l'étude et à la reconnaissance automatique de la parole en Arabe standard, Université de Nancy1, 12 Nov 1991.

[Maxime Gréau, 11] Maxime Gréau, Apache Maven 3.x, Décembre 2011.

[Nancy-Université] Nancy-Université UFR de langues et littératures étrangères, Département d'arabe.

[Olivier Carton, 15] Olivier Carton, L'essentiel de XML, Cours XML, Version du 13/10/2015

[Patrick Andries, 08] Patrick Andries, Unicode 5.0 en pratique, DUNOD 2008.

[P. Bertrand, 11] patrice Bertrand, Comprendre l'open source et les logiciels libres, 2011

[S.BOULAKNADEL, 08] Siham Boulaknadel Traitement Automatique des Langues et Recherche d'Information en langue arabe, Université de Nantes, 18 Oct 2008.

[S. STEER & M. FITZGIBBON] Sylvain STEER & Magali FITZGIBBON, Recueil de fiches explicatives de licences libres, INRIA.

[S.Zaabalaoui, 06] Salah eddine al Zaabalaoui, معجم أخطاء الكتاب، دار الثقافة و التراث دمشق-سورية ، 2006

[S. Zaidi–Ayad, 12] Soraya Zaidi–Ayad, Une plateforme pour la construction d'ontologie en arabe : Extraction des termes et des relations à partir de textes (Application sur le Saint Coran), Université Badji Mokhtar, Annaba, 2012/2013.

[Sun Microsystems, 95] The Java Language: An Overview Sun Microsystems 1995

[VIENNEY& BIOUD, 04] Séverine VIENNEY et Mounira BIOUD, correction automatique: bilan perspective, Université de Franche-Comté, 2004.

[Vergne J, Giguet E ,98] Vergne J, Giguet E, Regards théorique sur le tagging, TALN, pp 22-31, (1998)

Webographie :

[1] : https://fr.opensuse.org/Logiciel_Libre_et_Open_Source

[2] :http://extensions.services.openoffice.org/fr/project/lightproof-grammar-checker-development-framework

[3] : https://launchpad.net/lightproof

[4] : https://tahadz.wordpress.com/2011/06/20/alightproof/

[5] :http://svn.arabeyes.org/viewvc/projects/arabiclightproof/lightproof-1.4/data/ar_DZ.dat?view=markup&pathrev=11759

[6] : https://languagetool.org/fr/

[7] : https://www.languagetool.org/usage/

[8] : http://outilstice.com/2014/01/language-tool-correcteur-open-source-en-ligne/

[9] http://www.enseignement.polytechnique.fr/informatique/profs/Julien.Cervelle/eclipse/

[10] https://eclipse.org/mars/

[11] http://www.tutorialspoint.com/maven/maven_tutorial.pdf