Présentation de la plateforme d'analyse linguistique médiévale

1. Introduction

Tout au long de ce document, notre projet sera présenté à travers la méthodologie suivie pour développer la plateforme d'analyse linguistique médiévale. Nous allons aborder les différentes phases de réalisation de projet, par lesquelles nous parcourons les différentes tâches. D'un point de vue théorique, ce document montre une démarche pour réaliser un outil de TAL en se basant sur une méthode de génie logiciel.

2. Méthodologie du travail

Pour réaliser ce projet et l'accorder aux règles du génie logiciel, il faut bien organiser les tâches et suivre une méthodologie de travail précise. Il s'agit d'une méthode simple et générique.

Cette méthode a été présentée par Roques (2002) dans son livre *UML-Modéliser un site e-commerce*. Elle se situe entre deux méthodes agiles : UP (Unified Process) qui constitue un processus de développement itératif et incrémental par lequel le projet est découpé en phases courtes – à l'issue de chacune d'elle, une nouvelle version est incrémentée ; et XP (Extrême Programming) qui est une approche basée sur les besoins des utilisateurs où ces derniers valident et proposent des modifications à chaque incrémentation. [1]

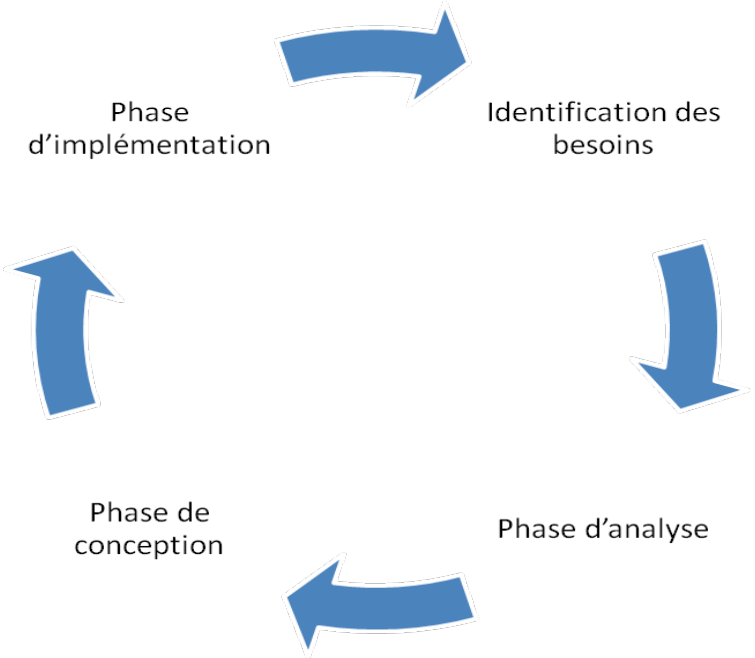

Fig. 1. *Cycle de notre méthodologie de travail*

Comme le montre la figure 1, cette méthode est composée de 4 phases :

- Identification des besoins et spécification fonctionnelle
- Phase d'analyse
- Phase de conception
- Phase d'implémentation

Parmi les caractéristiques de cette méthode, retenons :

· Une conduite par les cas d'utilisation : le but de système modélisé est de répondre à nos besoins. À chaque itération de la phase d'analyse, de conception et de test, on vérifie que les besoins ont bien été

[1] Pascal Roques, *UML-Modéliser un site e-commerce*, Paris, 2002.

satisfaits.

· Une méthode relativement légère et restreinte avec des activités de modélisation en analyse et conception : il n'est pas nécessaire de tout détailler et de préciser tous les éléments et les objets ; en revanche, il faut modéliser régulièrement les besoins des utilisateurs.

· Une méthode fondée sur l'utilisation d'un sous-ensemble nécessaire et suffisant du langage UML : on peut se limiter à un ensemble des diagrammes dont nous voyons qu'il satisfait les besoins – cas d'utilisation, diagramme de séquence et diagramme de classe.

3. Identification des besoins et spécifications fonctionnelles

L'objectif principal de la phase d'identification des besoins et des spécifications fonctionnelles est d'étudier la faisabilité du projet, de délimiter la portée du système envisagé, d'identifier les risques critiques et de tracer l'architecture candidate du système en question.

Dans un premier temps, les cas d'utilisation sont créés afin d'identifier et de modéliser les besoins. Ces derniers sont déterminés à partir des informations recueillies lors des rencontres entre le système et les utilisateurs. Après l'identification et la modélisation grâce au diagramme des cas d'utilisation, la description textuelle des cas d'utilisation est détaillée, ainsi que la production de diagrammes de séquence illustrant cette description textuelle.

Afin de modéliser notre application, nous utilisons le langage de modélisation UML (*Unified modeling language*) qui est une notation permettant de modéliser un problème de façon standard.

Ce langage est né de la fusion de plusieurs méthodes existant auparavant. Il est devenu désormais la référence en terme de modélisation objet à tel point que sa connaissance est souvent nécessaire pour un développeur objet.

La figure 2 présente le diagramme des cas d'utilisation qui est un diagramme UML permettant de structurer les besoins des utilisateurs et les objectifs correspondants d'un système ; il permet d'identifier les utilisateurs du système (acteurs) et leurs interactions avec le système.[2]

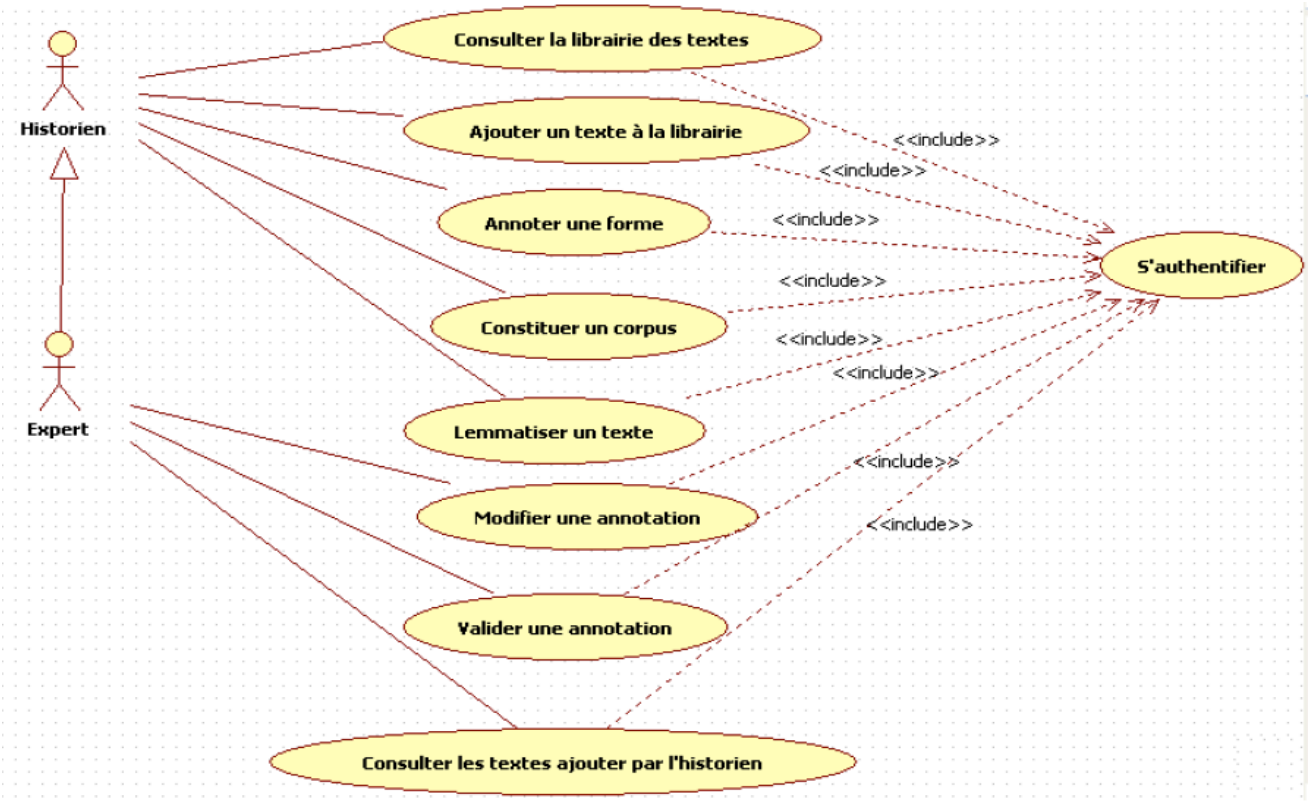

Fig. 2. *Diagramme des cas d'utilisation*

[2] Site UML en français : http://uml.free.fr/

Le diagramme des cas d'utilisation montre l'existence de deux acteurs, l'Historien et l'Expert. Nous remarquons qu'il y a une relation d'héritage entre eux. Par conséquent, l'expert peut effectuer les mêmes tâches que l'historien ; de plus, il peut contrôler les opérations réalisées par l'historien.

À partir de ce diagramme, chaque cas d'utilisation peut être décrit et analysé. À titre d'exemple, voici la description des cas d'utilisation « consulter la librairie » et « Annoter une forme » avec la présentation du diagramme des séquences et d'un scénario.

Cas d'utilisation « Consulter la librairie »

Titre : Consultation de la librairie
Descriptions : Il s'agit de visualiser les différents textes de la librairie. L'historien a le droit de visualiser ces textes et seulement les textes qu'il a ajoutés. Cette visualisation lui permet de construire son corpus pour l'annotation et la lemmatisation.
Acteurs : historien, expert.

Diagramme de séquence

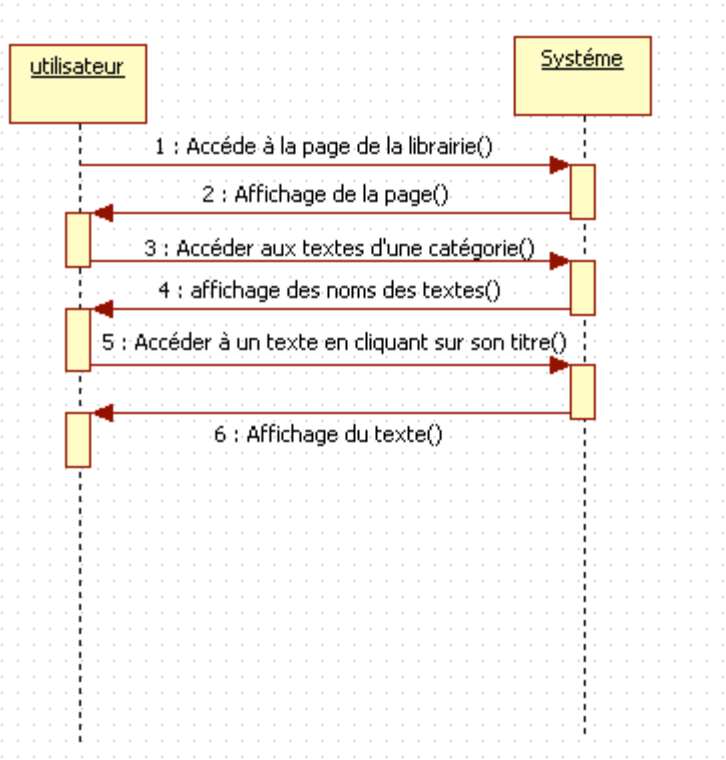

Fig. 3. *Diagramme de séquence de cas d'utilisation « Consulter la librairie »*

Description des scénarios

Action acteur	Action système
1. L'acteur accède à la page de la librairie qui contient les textes.	2. Le système affiche la page de la librairie.
	3. Le système affiche tous les titres des textes stockés dans la base des données.
4. L'acteur accède au texte	
	5. Le système affiche le texte et les informations qui lui sont liées.

Cas d'utilisation « Annoter une forme »

Titre : Annotation d'une forme
Descriptions : Ce cas permet à l'utilisateur d'annoter les formes d'un texte. Cet utilisateur peut utiliser les annotations proposées par notre système et les modifier.
Acteurs : historien, expert.

Diagramme de séquence

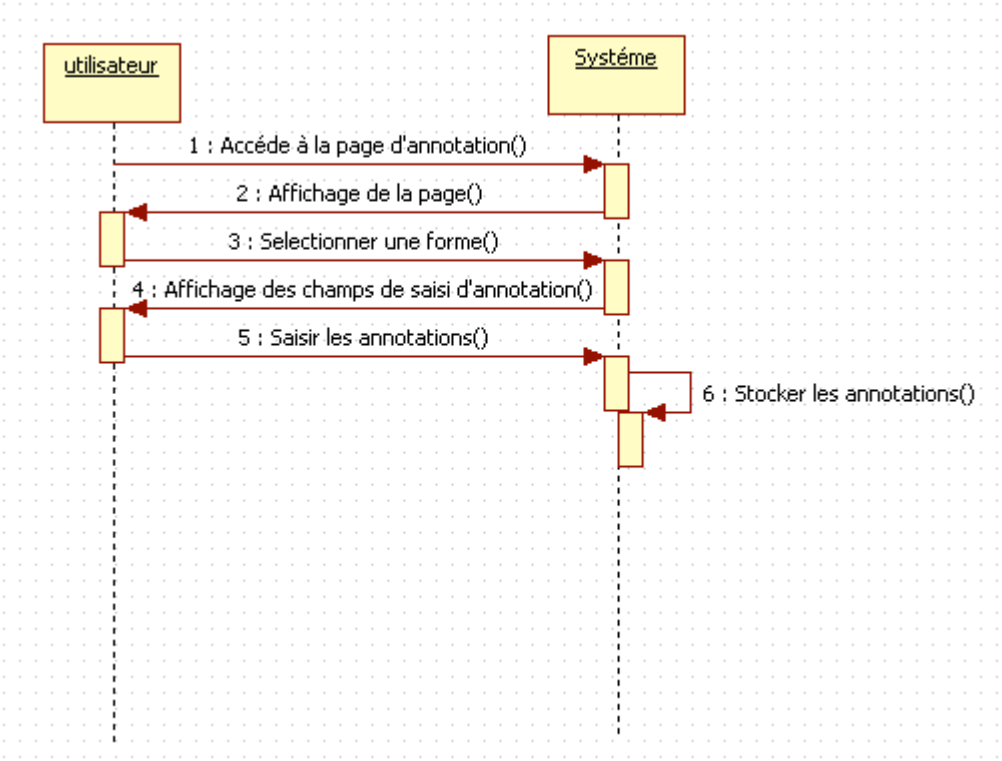

Fig. 4. *Diagramme de séquence de cas d'utilisation « Annoter une forme »*

Description des scénarios

Action acteur	Action système
1. L'acteur choisi un texte.	2. Le système affiche le texte choisi par l'utilisateur.
3. L'acteur clique sur une forme de texte.	4. Le système affiche les champs à remplir par les valeurs d'annotation.
5. L'acteur saisit ces annotations et les envoie en cliquant sur un bouton.	6. Le système stocke ces annotations dans sa base de croyance.

Une fois chaque cas d'utilisation analysé, il est possible de distinguer les différents modules qui doivent être implémentés dans la plateforme.

Quatre modules principaux sont distingués :

- La librairie des textes : consulter des textes, rechercher un texte, constituer un corpus et ajouter un texte.
- Analyse linguistique : annotation manuelle, étiquetage morphosyntaxique, détection des entités nommées.
- Administration : gérer les comptes utilisateurs, gérer le corpus de travail et les annotations.
- Import et export : import et export vers le XML TEI et les logiciels de textométrie tels que Analyse, Hyperbase, Lexico 3, Textométrie, Le Trameur, TXM, etc.

4. Phase d'analyse

Après l'identification et l'analyse des différents besoins, la phase d'analyse s'appuie sur les spécifications et vise à déterminer les problèmes posés. On va se concentrer sur les différents métiers nécessaires au développement de notre application : traitement automatique des langues et développement Web.

4.1. Les différents processus

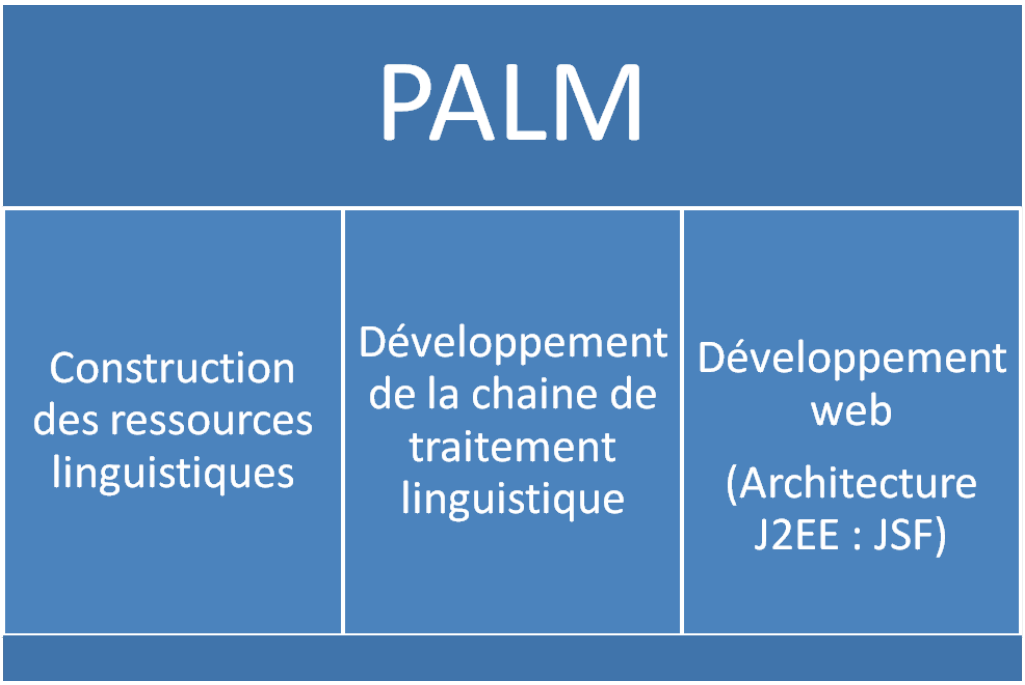

Fig. 5. *Les différents processus de la plateforme*

Comme le montre la figure 5, nous distinguons trois processus pour notre application : construction des ressources linguistiques, développement de la chaine de traitement linguistique et développement web en utilisant la technologie Java Server Faces de l'architecture J2EE.

Notre analyse est inspirée des méthodes d'intelligence artificielle dans lesquelles nous distinguons un processus de représentation des connaissances, qui correspond aux constructions des ressources linguistiques, et un processus de raisonnement, qui correspond à la chaîne de traitement linguistique.

Ces deux processus peuvent être détaillés de la manière suivante :

- **Construction des ressources linguistiques**

Nous distinguons deux types de ressources linguistiques :

1. Des ressources linguistiques externes : corpus, dictionnaires et toutes sortes de grammaires locales.
2. Des ressources linguistiques locales : il s'agit des ressources linguistiques constituées par notre équipe, tels que les corpus, les dictionnaires, les règles de morphologie, les règles syntaxiques, etc.

- **Développement de la chaîne de traitement linguistique**

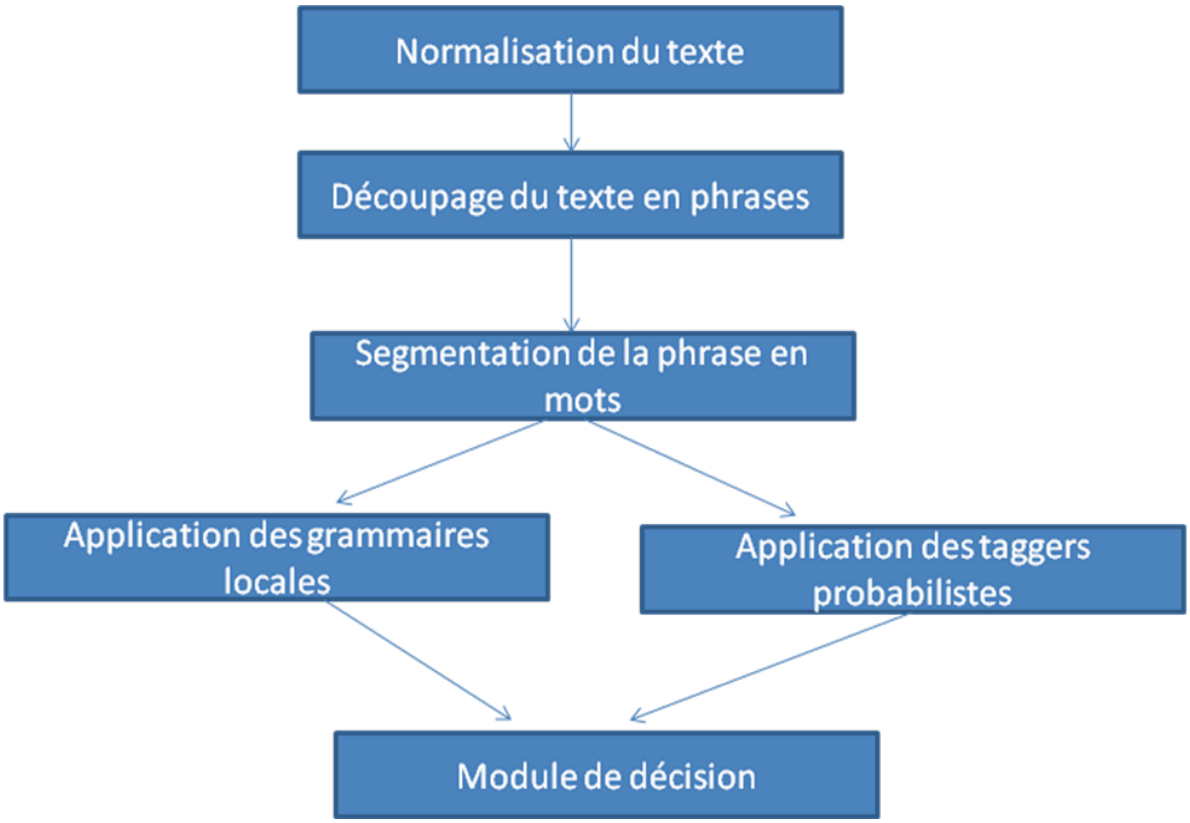

Fig. 6. *Chaine de traitement d'étiquetage morphosyntaxique automatique*

La figure 6 montre la chaine d'étiquetage morphosyntaxique, mais notre plateforme contient d'autres chaines de traitement qui ont des fonctionnalités différentes : chaîne d'annotation manuelle, chaine de détection des entités nommées, etc.

En ce qui concerne la chaine d'étiquetage morphosyntaxique, la première étape de la chaîne d'analyse linguistique est la normalisation du texte. Cette étape consiste à transformer le texte dans un format d'encodage UTF-8 et dans un format physique interne à la plateforme (.snt). Il faut ensuite appliquer des graphes qui permettent la segmentation des textes en phrases, puis des graphes permettant la segmentation des textes en mots. Puis, des taggers probabilistes telles que Mate et TreeTagger sont appliqués, ainsi que des grammaires locales telles que des règles morphologiques, syntaxiques et contextuelles. Enfin, un module de décision détermine la bonne étiquette morphosyntaxique selon les résultats donnés par les différents taggers.

La section suivante détaille la méthode utilisée pour l'étiquetage morphosyntaxique, qui couvre les trois dernières étapes de la chaine de traitement.

4. 2. Méthode hybride pour l'étiquetage morphosyntaxique des textes médiévaux

L'étiquetage morphosyntaxique est une opération qui permet, à partir d'un ensemble de couples (forme, étiquette morphosyntaxique), de choisir, pour chacun des mots du texte parmi ses étiquettes morphosyntaxiques associées celle(s) qui correspond(ent) au contexte[3].

Les méthodes d'étiquetage morphosyntaxique se divisent en deux grandes approches :

- Approche à base de règles : elle utilise généralement des « grammaires locales »[4] qui peuvent être modélisées sous forme d'automate ou de transducteur.

[3] Serge Fleury, site plurital : http://www.tal.univ-paris3.fr/cours/masterproj.htm

[4] Maxi Silberztein, manuel d'Intex : http://mshe.univ-fcomte.fr/intex/downloads/Manuel.pdf

Ils prennent en entrée le graphe correspondant à la projection de différentes étiquettes sur le texte et éliminent une partie des chemins de ce graphe qui rajoute des chemins[5].

- Approche probabiliste : C'est l'approche la plus utilisée actuellement, elle s'appuie sur l'ordre des mots en faisant un graphe orienté des étiquettes possibles pour chacun des mots. elle peut se mettre en œuvre à l'aide de modèles de Markov cachés à titre d'exemple TreeTagger.[6]

Ces approches classiques suffisent-elles pour étiqueter des textes médiévaux ?

Une étude réalisée par E. Tzourkermann en 1996 sur deux corpus en français moderne extraits du journal *Le Monde* montre que plus que la moitié des mots ne sont pas ambigus et qu'une grosse part de l'ambiguïté est détenue par un petit nombre des mots fréquents qui sont généralement des mots outils. [5]
Si le taux d'ambiguïté pour les langues modernes se limite à un nombre fini de mots, en langues médiévales, il existe un grand nombre de possibilités pour une même forme morphosyntaxique et une liste complète des formes est impossible à établir.[7]

Les langues médiévales sont des langues en pleine évolution dont l'orthographe et le système flexionnel ne sont pas stables.

Afin de répondre aux spécificités et aux diversités des langues médiévales, nous proposons une méthode hybride qui combine un tagger à base des règles réalisées en utilisant les normes, modules, automate et transducteur du système Intex, et la puissance des taggers probabilistes telles que TreeTagger et Mate.

Cette méthode est indépendante de la langue, elle peut être appliquée à plusieurs langues si nous disposons des ressources linguistiques nécessaires (corpus étiqueté, dictionnaires, règles de morphologie, etc.).

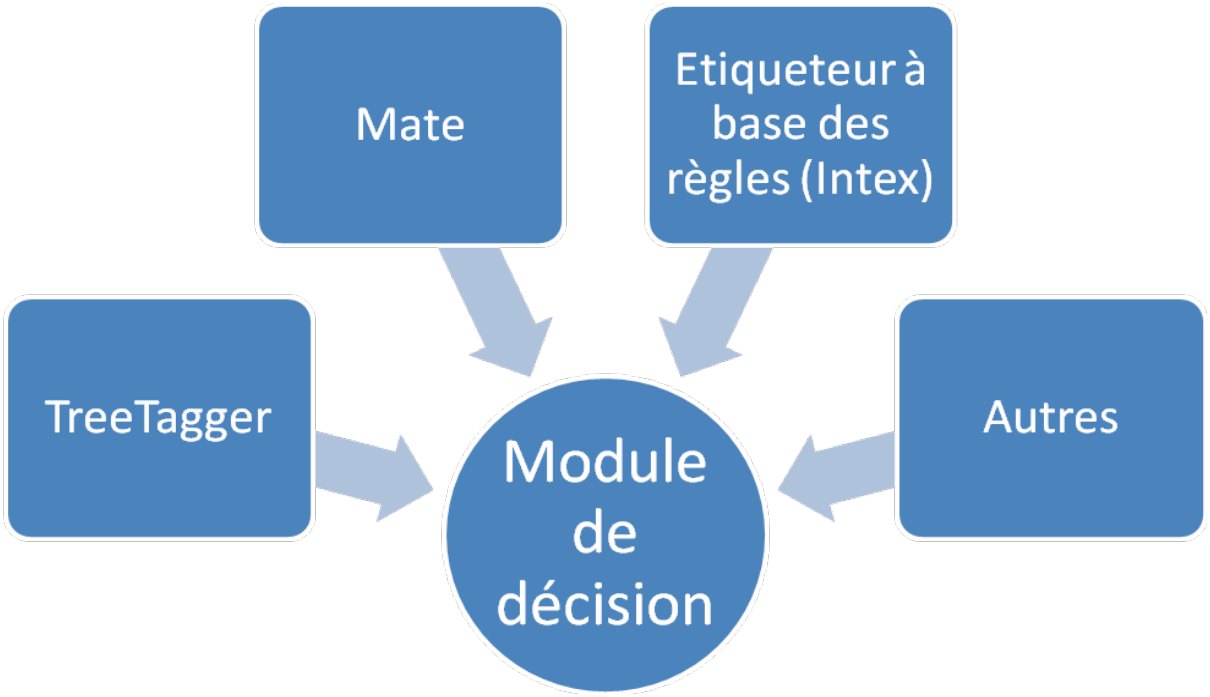

Fig. 7. *Méthode hybride pour l'étiquetage morphosyntaxique*

Mourad AOUINI

[5] Benoît Habert, Adeline Nazarenko et André Salem, *Les linguistiques de corpus*, Paris, 1997.
[6] Helmut Schmid, « Probabilistic Part-Of-Speech Tagging Using Decision Trees », dans *Proceedings of the International Conference on New Methods in Language Processing*, 1994, p. 44-49.
[7] Gilles Souvay et Jean-Marie Pierrel, « Lemmatisation des mots en moyen français », *Traitement automatique des langues*, 50, 2009, en ligne à l'adresse suivante : http://www.atala.org/LGeRM