Analyse syntaxique de l'Arabe: Le système MASPAR

Chafik ALOULOU
Faculté des Sciences Economiques et de Gestion de Sfax
Laboratoire de recherche LARIS, B.P. 1088
3018 - Sfax – TUNISIE
Tél. (216) 74 27 87 77, Fax (216) 74 27 91 39
chafik.aloulou@fsegs.rnu.tn

Mots-clefs

Analyse syntaxique, Robustesse d'analyse, HPSG, système multi-agent, AgentBuilder.

Syntactic Analysis, Robust analysis, HPSG, multi-agent system, AgentBuilder,.

Résumé

De nombreux systèmes de Traitement Automatique des Langues (TAL) utilisent une architecture séquentielle basée sur la transmission, à la fin de chaque phase d'analyse, des résultats trouvés à la phase d'analyse suivante. Ces types de systèmes séquentiels posent plusieurs problèmes (i.e. explosion combinatoire des solutions, lourdeur d'analyse, etc.). Pour remédier à ces problèmes, plusieurs solutions de remplacement ont vu le jour, nous pouvons citer par exemple, l'utilisation des approches multi-agent que nous avons adopté pour faire l'analyse syntaxique de textes Arabes, et que nous présentons dans cet article.

1 Introduction

Le traitement automatique de la langue naturelle (TALN) touche plusieurs domaines, telles que, les applications de correction grammaticale, les applications de communication homme/machine, les applications de traduction automatique, etc. L'automatisation de l'une de ces applications nécessite en général une étape d'analyse du texte ou document source, qui se fait à son tours par la décomposition de cette analyse en sous-tâches calquées sur les différents niveaux d'analyse linguistique à savoir l'analyse lexicale, l'analyse morphologique, l'analyse syntaxique, l'analyse sémantique et l'analyse pragmatique. Cette décomposition permet de se fixer un objectif moins ambitieux, mais il est clair, d'un point de vue linguistique, que les tâches ne sont pas indépendantes les unes des autres et qu'à chaque niveau, les connaissances des autres niveaux sont nécessaires (Abeillé, 2000). En plus, ces différents niveaux d'analyse sont couplés, en général de manière séquentielle, ce qui suppose que les phases d'analyse sont quasi-dépendantes.

Les systèmes basés sur un couplage séquentiel présentent des inconvénients majeurs pour l'analyse de la langue Arabe: absence d'interaction entre les niveaux de représentation, absence de distribution du contrôle et des connaissances, système difficilement évolutif. La principale conséquence étant une explosion combinatoire qui peut être résorbée par une coopération entre les différentes phases d'analyse ou les différents modules. Cette coopération doit permettre de trouver les bonnes analyses syntaxiques d'un morceau de texte tout en éliminant rapidement les solutions parasites et par la suite aboutir à une analyse robuste (Warren et Stefanini, 1996).

Dans cet article, nous nous intéressons principalement à la phase d'analyse syntaxique, qui constitue, à notre sens, la base du traitement automatique de tout énoncé écrit en langage naturel, et nous présentons les limites des méthodes séquentielles, dites classiques, de couplage des niveaux d'analyse linguistique. Nous mettons l'accent sur l'aspect robustesse, qui constitue une qualité très recherchée des analyseurs, étant donné que les textes bien formés sur le plan linguistique sont de plus en plus rares. Nous introduisons ensuite les systèmes multi-agents afin de pouvoir décrire notre système d'analyse de la langue Arabe, qui est déjà basé sur une approche multi-agent.

2 La robustesse d'analyse

Les textes libres peuvent contenir des erreurs de tout genre (i.e., des fautes d'orthographe, de grammaire, des omissions, des mots inconnus ou ambigus par rapport au lexique de l'analyseur). Les analyseurs robustes doivent fournir systématiquement une analyse syntaxique de chaque phrase contenue dans le texte concerné y compris celles qui sont mal formulées ou dont la structure n'est pas couverte par la grammaire du système (Giguet, 1997), (Shalen, 2000). En effet, vu la versatilité des langues, il est impossible d'écrire une grammaire complète, capable de traiter toutes les phrases imaginables dans une langue. Pour cette raison, une stratégie d'analyse partielle est souvent adoptée : l'analyseur examine autant que possible chaque phrase en entrée et renvoie une analyse partielle lorsqu'une analyse complète n'est pas possible.

Plusieurs stratégies sont adoptées afin de gérer ces problèmes. Il existe des analyseurs non déterministes qui génèrent plusieurs analyses par phrase et utilisent ensuite des statistiques ou des règles de grammaire basées sur des heuristiques afin de sélectionner l'analyse la plus probable parmi les réponses. D'autres analyseurs sont déterministes : ils ne rendent qu'une seule analyse par phrase même si plusieurs sont possibles en termes réels. Les grammaires sont limitées dans ces cas-là et un certain taux d'erreur est accepté d'avance.

La vitesse de la réponse est une autre contrainte importante pour les analyseurs car, afin d'être utile dans le monde réel, ils doivent retourner une réponse très rapidement.

La quête pour un analyseur syntaxique robuste, capable d'analyser des textes libres en profondeur, a mené au développement de toute une gamme d'analyseurs syntaxiques ces dernières années. Ces analyseurs varient en termes de stratégies (i.e., le déterminisme contre le non-déterminisme, l'analyse partielle contre l'analyse complète ou approfondie), de base théorique (i.e., les analyseurs statistiques contre les analyseurs symboliques et les analyseurs basés sur la grammaire syntagmatique contre ceux basés sur la grammaire de dépendances,

etc.) (Giguet, 1997). Cependant, alors que chaque analyseur a ses points forts et fonctionne bien pour certaines tâches, aucun d'entre eux n'est capable de faire une analyse complète de toutes les phrases dans un corpus de texte libre. Les méthodes statistiques, quoique prometteuses, ne suffisent pas, à elles seules, pour régler toutes les tâches du traitement automatique de la langue. Les grammaires symboliques sont également nécessaires afin d'obtenir une représentation précise et fiable de la sémantique, ce qui est crucial pour beaucoup de tâches de traitement automatique des langues. Récemment, il y a une tendance croissante à mélanger les approches différentes envers l'analyse syntaxique afin de profiter des points forts des différents types d'analyseurs.

Pour l'analyse de la langue Arabe, et afin d'avoir une analyse robuste, nous avons choisi d'adopter une autre approche d'analyse qui se base, d'une part sur la distribution des traitements[1] et d'autre part, par l'application du formalisme HPSG (Head-Driven Phrase Structure Grammar, ou Grammaire syntagmatique endocentrique[2]) qui privilégie une composante lexicale riche permettant de réduire au minimum le nombre de règles syntaxiques et qui propose des représentations lexicales riches et structurées. Ceci nous permettra de disposer du maximum d'informations sur les mots composants le texte en cours d'analyse et par la suite de réduire au maximum le nombre d'ambiguïtés qui se génèrent au cours de l'analyse. Nous présentons dans les paragraphes suivants une description de notre système d'analyse ainsi que la méthode adoptée.

3 L'approche multi-agent

Le terme « agent » est très répandu en informatique et il est utilisé dans plusieurs contextes. Plusieurs définitions d'agent existent aujourd'hui. D'après Ferber (Ferber 1995), " *On appelle agent une entité physique ou abstraite qui est capable d'agir sur elle-même et son environnement, qui dispose d'une représentation partielle de cet environnement, qui, dans un univers multi-agent, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents* ".

Cependant, les agents peuvent être décrits au moyen de plusieurs concepts. Les principaux concepts des agents 'logiciels' sont : l'autonomie, la réactivité , l'orienté-but (Goal-Oriented), la continuité, la communication, la coopération, l'anthropomorphisme, la mobilité, l'apprentissage, l'adaptabilité et l'auto-déclenchement (self-starting).

En respectant ces concepts, les agents peuvent communiquer entre eux, pour résoudre des problèmes, et forment par conséquent un système multi-agent. La communication entre les différents agents peut se faire par partage d'une zone de données communes dans laquelle les agents peuvent mettre et/ou trier des conclusions (tableau noir), ou aussi par envoi direct de messages à l'agent ou bien aux agents concernés. Dans ce type de communication les agents se connaissent les uns les autres.

---

[1] Plus précisément, une approche d'analyse multi-agent.

[2] Cette traduction de "Head-driven" décrit également les structures endocentriques, "headed phrase" en anglais, c'est-à-dire celles qui possèdent une seule tête.

Les domaines d'application des systèmes multi-agents sont très variés, tels que par exemple notre domaine d'application qui est le traitement automatique de la langue naturelle Arabe. Notre choix de l'approche multi-agent se justifie par nos propos de remédier aux problèmes de robustesse d'analyse déjà présentés dans le paragraphe précédent, et aussi de résoudre certains problèmes rencontrés dans les systèmes classiques d'analyse basés sur des approches séquentielles. Parmi ces problèmes nous pouvons citer :

- le manque d'interaction entre les niveaux de représentation,
- le manque de distribution des connaissances et du contrôle,
- la rigidité de l'architecture,
- la lourdeur de traitement,
- la difficulté d'évolution du système

Ainsi nous avons proposé dans (Aloulou 2002-A), (Aloulou 2002-B) un système d'analyse de textes Arabes que nous avons baptisé MASPAR (Multi-Agent System for Parsing ARabic). Dans ce qui suit, nous présentons ce système en exposant le rôle de chaque agent qui le constitue. Nous présentons ensuite la plateforme utilisée pour le développement de MASPAR en justifiant le choix de cette plateforme. Nous expliquons, ensuite le modèle agent de MASPAR et nous illustrons son fonctionnement par un exemple d'exécution.

4 Présentation du système MASPAR

Notre système d'analyse est composé, à l'état actuel, de six agents, à savoir l'agent 'Segmenteur', l'agent 'lexical', l'agent 'morphologie', l'agent 'syntax', l'agent 'anaphore' et l'agent 'ellipse' (figure 1) (Aloulou 2002-A):

- **l'agent 'Segmenteur':** cet agent à pour objectif de décomposer le texte brut en paragraphes et en phrases.

- **l'agent 'Lexical' :** cet agent propose une multitude de décomposition de mots en un ensemble de triades affixales (préfixe, affixe, suffixe) et de racines (Ben Hamadou 1993). Ensuite, il vérifie l'appartenance au langage de chaque mot, obtenu à partir de l'agent 'Segmenteur' en se basant sur les différentes décompositions obtenues.

- **l'agent 'Morphologie' :** cet agent à pour objet de déterminer les caractéristiques morpho-syntaxiques de chaque mot (Belguith 1999). Ainsi, par exemple, le mot 'حفظ'[3] (a appris) aura les caractéristiques suivantes :

    - **Type** : Verbe
    - **Personne** : 3ème
    - **Temps** : Accompli

3 Vu que nous travaillons sur l'Arabe non voyellé, le mot حفظ peut être interprété de la manière suivante : Type : Nom, Genre: Masculin, Nombre: singulier,

- **l'agent 'Syntaxe' :** cet agent détermine si une phrase (ou une succession de mots) appartient ou non au langage et respecte donc les règles de grammaire de la langue Arabe. Dans notre cas nous avons choisi de représenter notre grammaire selon le formalisme HPSG (Head-Driven Phrase Structure Grammar) qui s'inspire de plusieurs théories syntaxiques contemporaines, dont GPSG, la théorie gouvernement-liage et la grammaire catégorielle. HPSG se distingue, notamment, par un hyper-lexicalisme qui réduit au minimum le nombre de règles syntaxiques et propose des représentations lexicales riches et structurées. L'application de HPSG pour la langue Arabe a nécessité une modification des schémas standards pour pouvoir intégrer les particularités de la langue Arabe.

- **l'agent 'Ellipse' :** cet agent à pour but de reconstruire les différents types d'ellipses. Il détecte et reconstruit, suivant les cas, l'ellipse du sujet, du verbe, du complément ou de la proposition.

- **l'agent 'Anaphore' :** le but de cet agent est de retrouver l'antécédent d'une forme anaphorique. Il établit aussi le lien entre la forme anaphorique et son antécédent.

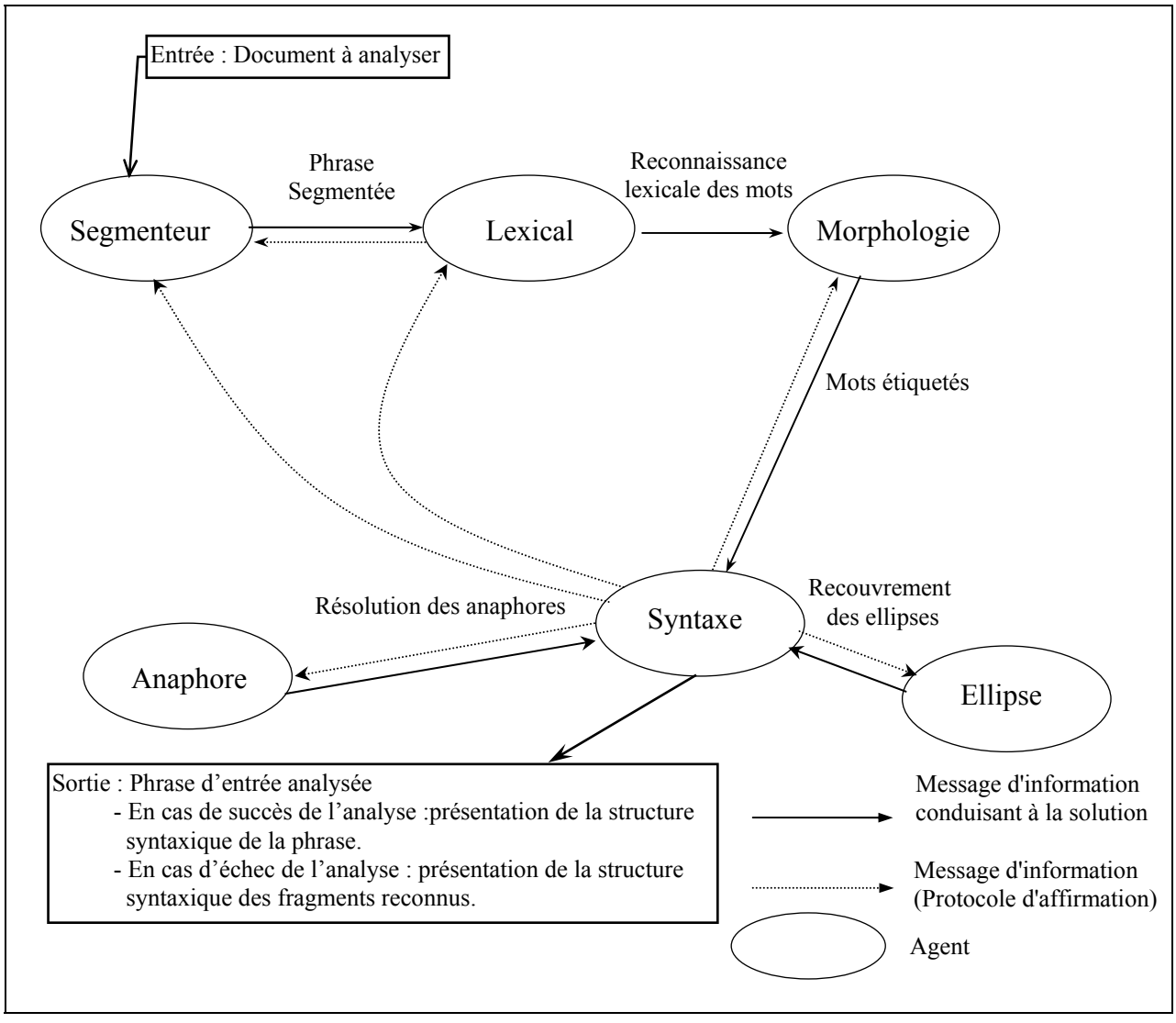

Figure 1 : Architecture générale du système MASPAR

Les différents agents de notre système communiquent entre eux par un échange dynamique de messages. Ces messages varient selon le type de document et le type de la phrase donnée en entrée. Le premier agent qui se lance, suite à la réception d'un document électronique à analyser, est l'agent de segmentation ('Segmenteur'). Suite à l'identification de la première phrase, et suite à l'identification du premier mot de cette phrase, un message est envoyé à l'agent 'Lexical'. Ce dernier génère toutes les décompositions possibles du mot en racine, préfixe, infixe et suffixe. Ainsi dès que l'agent 'Lexical' génère une décomposition, il l'envoie à l'agent 'Morphologie'. Cette tâche se répète au fur et à mesure qu'une décomposition est générée et pour chaque mot de la phrase[4].

L'agent 'Morphologie' utilise ses compétences pour déterminer les caractéristiques morphologiques du mot reçu. Signalons qu'une décomposition donnée peut générer plus qu'une seule liste de caractéristiques morphologiques possibles.

L'agent 'Syntax' sera lancé dès la réception des caractéristiques morphologiques du premier mot. A ce moment, cet agent commence par générer la matrice attribut/valeur relative au mot obtenu, et sélectionne par la suite, la liste des règles de la grammaire validant ce premier mot trouvé. Cette liste de règles, qui est déjà classée par ordre d'apparition des règles dans le corpus, sera raffinée au fur et à mesure de l'évolution de l'analyse. L'agent 'Syntax' peut orienter l'agent 'Morphologie' dans le choix des caractéristiques morphologiques des mots suivants. Ceci se fera par envoie de messages d'information présentant les caractéristiques morphologiques des mots attendus. Ainsi, les solutions parasites seront éliminées dans la progression de l'analyse. Dans le cas ou plusieurs familles de règles sont applicables, et afin d'éviter des retours en arrière dans l'analyse, nous appliquons un nombre de clonages de l'agent 'Syntax' en fonction des règles partiellement applicables. L'instance clonée non validée au cours de l'analyse sera détruite.

Toutefois, et dans le cas de blocage de l'analyse, l'agent 'Syntax' prendra l'initiative de débloquer la situation. Selon les caractéristiques morphologiques des mots déjà traités et selon le groupe de règles de la grammaire déjà sélectionnées, l'agent 'Syntax' peut, ou bien demander l'aide de l'agent Ellipse, ou celui de l'agent Anaphore qui vont utiliser leurs compétences pour recouvrir les structures elliptiques ou anaphoriques, ou peut même demander à l'agent 'Morphologie' de proposer d'autres caractéristiques morphologiques qui à son tour peut demander à l'agent découpage d'autres alternatives de décomposition (c'est le cas par exemple de groupement de mots). L'analyse s'achève par la reconnaissance d'une structure syntaxique valide, ou bien se limite à la présentation des fragments syntaxiques reconnus.

L'agent de segmentation ('Segmenteur'), qui se base en grande partie sur les indicateurs de surfaces pour identifier les phrases et les paragraphes, nécessite dans certains cas de connaître le type de quelques mots du texte ou même leurs enchaînements. Ceci nécessite d'anticiper dans l'analyse et de déclencher le phénomène d'analyse pour agir localement sur une partie du texte. Cette liste d'alternatives ou de modèles de communication entre les différents agents n'est pas exhaustive, et d'autres cas sont envisageables selon la complexité des textes traités et les ambiguïtés qui s'y trouvent.

---

[4] Ceci concerne uniquement les mots qui vont subir une décomposition, sont exclus les particules, tel que, les pronoms personnels, etc.

Cette suite de modèles de communications a débouché sur la conception détaillée de notre système. Cette conception respecte au niveau des spécifications, les différents modèles donnés par AUML (Agent Unified Modeling Language) qui représente une extension de la méthode UML, elle est employée principalement pour la modélisation des interactions multi-agent, elle introduit principalement deux nouveaux concepts au niveau de la représentation des protocoles et au niveau de la représentation des interactions entre agents (Odell, 2000).

5 Présentation de la plateforme de développement

Afin de choisir une plate-forme de développement de notre système, nous avons mené une étude comparative de quelques plates-formes disponibles. Notre étude s'est portée principalement sur les trois plates-formes suivantes :

- Dima : Développement et Implémentation de Systèmes Multi-Agents
  http://www-poleia.lip6.fr/~guessoum/dima.html
- EMAC : Environnement Multi-Agent à mémoire Collective
- AgentBuilder
  http://www.agentbuilder.com/

Le tableau ci-dessous résume les différents critères d'évaluation que nous avons pris en considération pour comparer les trois plates-formes, ainsi que les valeurs attribuées pour chacun de ces critères.

Critères	**DIMA**	**EMAC**	**AgentBuilder**
Architecture	Centralisée	Décentralisée	Décentralisée
Type d'agent	Hybrides	Réactifs	Cognitifs
Compétences	Réduites	Etendues	Très étendues
Langage de communication	Non	Oui	Oui
Contrôle	Centralisé	Distribué	Distribué
Niveau de coopération	Coopération faible	Coopération moyenne	Coopération complète
Langage de programmation	Java	C++	Java
Système d'exploitation	Win-NT, Win95	Unix	Win-NT, Win95, Unix
Réutilisabilité	Classement et importation ou modification du code source	Par modification du code source	Par copie

Suite à cette étude comparative, nous avons remarqué que la plate-forme à agent hybride présente un cycle de conception et d'implémentation non complet, quant à la plate-forme EMAC, elle présente un cycle de conception et d'implémentation acceptable alors que la plate-forme AgentBuilder présente un cycle de conception et d'implémentation relativement complet. Notre choix s'est fixé donc sur la plate-forme AgentBuider, que nous essayons de décrire d'avantage.

AgentBuilder est une suite d'outils intégrés permettant de construire des agents intelligents, développée par Reticular Systems Inc.

L'étape d'analyse consiste en la spécification en OMT (Object Modelling Technique) des objets du domaine et des opérations qu'ils peuvent effectuer, puis en la production d'une ontologie du domaine. Des outils graphiques sont fournis pour effectuer ces tâches.

L'étape de conception avec AgentBuilder consiste en la décomposition du problème en fonctionnalités que les agents doivent avoir. Puis les agents sont identifiés, leurs rôles et leurs caractéristiques sont définis, puis les protocoles d'interactions qu'ils utiliseront sont spécifiés.

L'étape de développement consiste en la définition du comportement des agents, par l'intermédiaire de leurs règles de comportement, croyances initiales, intentions et capacités.

La figure 2 représente le modèle de protocole agent de notre système de traitement automatique de la langue naturelle

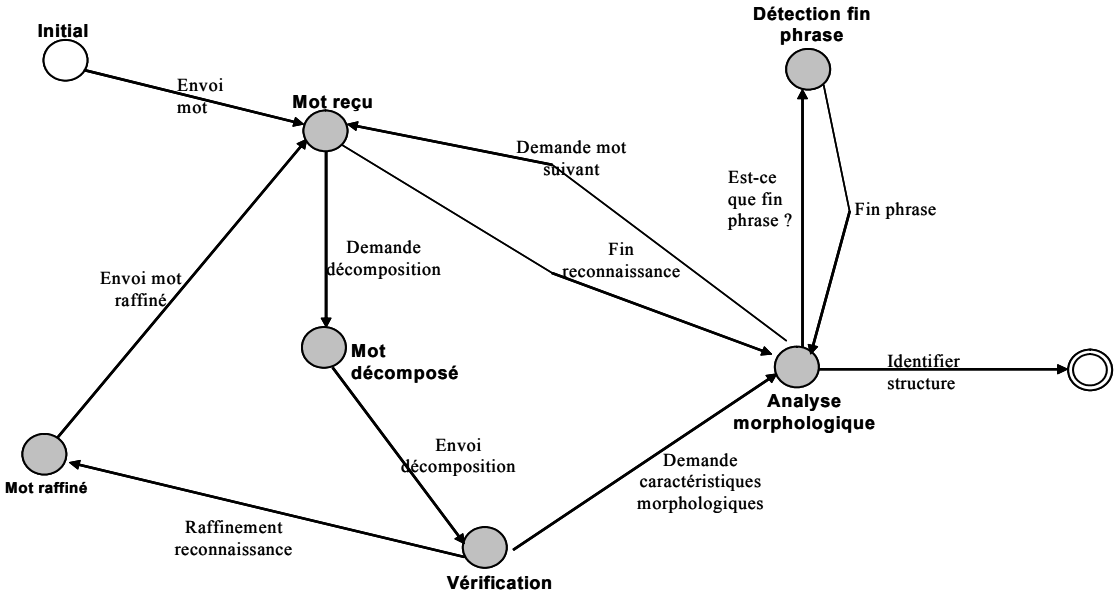

Figure 2 : Modèle de protocole du système MASPAR

Les agents engendrés au cours de la phase précédente sont exécutés par un Run-Time Agent Engine, qui interprète le code des agents. Un outil de déboguage est également fourni, permettant le suivi des états mentaux des agents et de leurs interactions. A cause du modèle purement BDI choisi, seuls des systèmes multi-agents composés d'un petit nombre d'agents intelligents sont envisageables. La complexité du déploiement est réduite par la nature conviviale des outils de déploiement.

L'exécution de notre système par AgentBuider nous donne un ensemble de fenêtres dont chacune représente un agent. Comme ça nous pouvons suivre l'état d'évolution de l'analyse. Nous donnons à titre d'exemple dans la figure suivante (figure 3), l'état initial et l'état final de l'analyse de la phrase suivante :

رجع المهاجر إلى الوطن.

(est revenu l'émigrant au le patrie.)[5]

Ainsi, la première fenêtre représente tous les agents de notre système qui sont à l'état inactif, dans l'attente de la fin de la saisie de la phrase à analyser. La deuxième fenêtre représente notre système après la fin de l'analyse. Nous remarquons que nous avons obtenu l'arbre syntaxique de la phrase analysée.

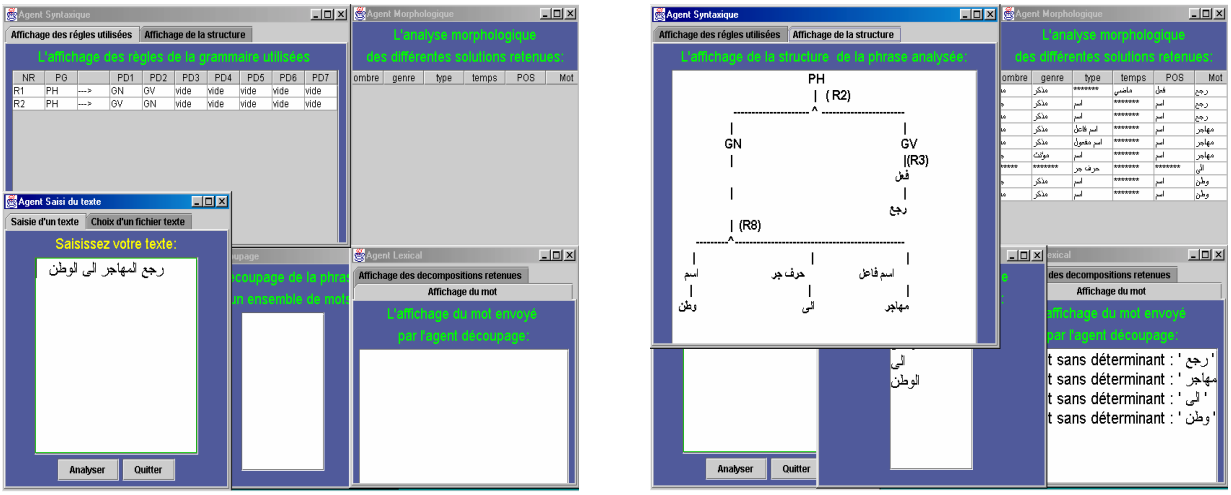

Figure 3 : Exemple d'exécution du système MASPAR

6 Conclusion

Dans cet article, nous avons présenté le système MASPAR d'analyse de textes Arabes non voyellés permettant de reconnaître les structures syntaxiques. Nous avons commencé par définir l'analyse syntaxique et son utilité dans les systèmes de TALN, ensuite nous avons défini la robustesse d'analyse qui est une qualité exigée au niveau de notre système d'analyse. Avant de définir l'architecture de notre système, nous avons décrit le modèle de communication qui peut se dérouler lors de l'analyse d'un texte. Signalons que ce système a été implémenté en utilisant la plate-forme AgentBuilder.

La première version de ce système s'est limitée au développement de l'agent 'Lexical', 'Morphologie' et 'Syntax', et accepte en entrée une phrase. Les résultats obtenus sont encourageants. Nous sommes actuellement en train d'implémenter une deuxième version de ce système en intégrer les agents 'Ségmenteur', 'Anaphore' et 'Ellipse'. Ainsi notre système sera capable d'analyser un texte électronique composé de plusieurs phrases et paragraphes.

[5] La traduction de la phrase est faite mot à mot.

7 Références

Abeillé A., Blache P., (2000) "*Grammaires et analyseurs syntaxiques*", dans J.-M. Pierrel édition Ingénierie des Langues, Paris, Hermès.

Aloulou C., Belguith Hadrich L. et Ben Hamadou A. (2002-A) MASPAR: "*Multi-agent System for Parsing Arabic*", IEEE international conference on systems, man and cybernetics, du 6 au 9 octobre 2002 à Hammamet - Tunisie.

Aloulou C., Belguith Hadrich L. et Ben Hamadou A. (2002-B), "*Utilisation des grammaires HPSG pour l'analyse de l'Arabe*", JEI'2002, 2ème journée des jeunes chercheurs en électronique et Informatique, 26 – 28 Mars 2002, Hammamet, Tunisie.

Belguith L. (1999), "*Traitement des erreurs d'accord de l'arabe basé sur une analyse syntagmatique étendue pour la vérification et une analyse multicritère pour la correction*", Thèse de doctorat en Informatique, Faculté des Sciences de Tunis, Février 1999.

Ben Hamadou A. (1993), "*Vérification et correction automatique par analyse affixale des textes écrits en langage naturel : le cas de l'arabe non voyellé*", Thèse d'Etat en Informatique, Faculté des Sciences de Tunis, Mars 1993.

Delisle S., Boufaden N, Moulin B. (1998), "*L'analyse syntaxique robuste de dialogues transcrits : de la parole aux actes en passant par l'écrit*", Actes des XXIIèmes Journées d'Études sur la Parole (JEP 98), Martigny (Suisse), 15-19 juin'98, 359-362, 1998.

Ferber J.,(1995), "*Les systèmes multi-agents vers une intelligence collective*". InterEdition, Paris, 1995.

Giguet E., Vergne J. (1997), "*Syntactic analysis of unrestricted French*", Actes de la conférence "International Conference on Recent Advances in Natural Languages Processing" (RANLP'97), pages 276-281, Tzigov Chark, Bulgaria, 11-13Septembre 1997.

Jacobs P. (1990), "*To parse or not to parse: Relation-driven text skimming*", Actes de 13ème conference COLING, Volume 2, pp 194-198, Helsinki , 1990.

Odell, J., Van Dyke Parunak, H., Bauer, B. (2000): "*Extending UML for Agents*". Workshop AOIS dans la conférence AAAI 2000. Austin, Texas, Juillet 2000.

Shaalan K., Farouk A et Rafea A. (2000), "*Towards An Arabic Parser For Modern Scientific Text*", Actes de la conference "International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications" : ACIDCA '2000, pp 228 – 235, Monastir, Tunisie,.Mars 2000.

Warren K., Stefanini M.H.. (1996), "*Modélisation et validation de protocoles de communication dans l'architecture TALISMAN*", Actes de la conference "Natural Language Processing and Industrial Applications" : NLP+IA'96, pp 270 – 276, Moncton, Canada,.Juin 1996.