Toward a semantic analyzer for Arabic language

Mohammed NASRI, Computer Science Department, Mohammadia School of Engineers,
Mohammed-V University-Agdal, Rabat, Morocco, mohammed.nasri@gmail.com
Lahsen ABOUENOUR, Computer Science Department, Mohammadia School of Engineers,
Mohammed-V University-Agdal, Rabat, Morocco, abouenour@yahoo.fr
Adil KABBAJ, Computer Science Department, INSEA, Rabat, Morocco, adil_kabbaj@yahoo.fr
Karim BOUZOUBAA, Computer Science Department, Mohammadia School of Engineers,
Mohammed-V University-Agdal, Rabat, Morocco, karim.bouzoubaa@emi.ac.ma

Abstract

Various research works have shown the requirements in terms of semantic processing to achieve more accurate Natural Language Processing (NLP) applications such as Information Retrieval (IR), Question Answering (QA), Reasoning, etc. This paper describes our approach of semantic analysis of the Arabic language. We adopt for this purpose a two-step approach; we firstly build an Arabic ontology, leveraging the content of the two linguistic resources Arabic WordNet and VerbNet and we secondly combine some Arabic NLP (ANLP) tools such as Stanford syntactic parser with the built ontology to automatically extract the semantics conveyed by the Arabic text. The extracted semantic is formulated in Conceptual Graph formalism, which is a promising and already proven powerful knowledge representation formalism, in reasoning and in many NLP fields.

Keywords: Arabic Natural Language Processing, Conceptual Graph, Semantic Analysis, Arabic ontology.

I. Introduction

In Natural Language Processing (NLP), unlike Information retrieval (IR) that are designed to return relevant documents related to the user's query, Question Answering (QA) is the task of automatically providing the exact answer to a question asked by the user in natural language.

QA field has reached an advanced level for the English language as well as for other Latin-based languages such as French. However, Arabic QA systems have not reached the same level yet. The cause behind this is that this task adds particular challenges mainly due to the specificities of the Arabic language (short vowels, absence of capital letters, complex morphology, etc.).
The QA task can be divided in most cases in three subtasks: Question Analysis, Passage Retrieval, and Answer Extraction. The difference between QA systems lies in how each one implements these subtasks.

Few implementations of Arabic QA exist: QARAB (Hammo et al., 2002) is a QA system that attempts to give a short answer to a question formulated in Natural Language. QARAB knowledge source is mainly based on a collection of Arabic newspaper text extracted from Al-Raya[1], a newspaper published in Qatar. This QA system does not semantically analyze the questions. AQAS (Mohammed et al., 1993), a knowledge-based QA system, extracts response only from structured data and does not process natural raw text. ArabiQA (Benajiba et al., 2007a) is an Arabic QA prototype based on the Java Information Retrieval System (JIRS) (Benajiba et al., 2007b), Passage Retrieval (PR) system and a Named Entities Recognition (NER) module. It embeds an Answer Extraction module dedicated especially to factoid questions. In order to implement this module, Benajiba and Rosso (2008) developed an Arabic NER system and a set of patterns manually built for each type of question. QASAL (Brini et al., 2009) is an Arabic QA which process with factoid questions and does not perform semantic analysis of the question.

The weakness of these systems is that they do not perform the semantic analysis of the documents and questions, which are often necessary, especially when the document containing the right answer does not include the keywords of the question.

In this paper, we present an approach for the semantic analysis of Arabic natural language that can be adopted to improve Arabic QA systems. Semantic analysis usually consists of two main components: a linguistic resource and an analyzing engine. Abouenour et al. (2013) have already introduced the linguistic resource, called Arabic ontology, built by leveraging the contents of Arabic WordNet (AWN) (Elkateb et al., 2006) and Arabic VerbNet (AVN) (Mousser, 2010 and 2011). This new resource (ontology) has the advantage of combining the high lexical coverage and semantic relations between words existing in AWN together with the formal representation of syntactic and semantic frames corresponding to verbs in AVN. This paper presents the semantic analyzing engine (so called analyzer). This analyzer begins by syntactically parsing the Arabic text with the Stanford parser (Klein and Manning, 2002 and 2003), which tags the constituents of the text with thematic roles and outputs the sentence syntactic patterns. This syntactic information combined with the semantic information provided by the built ontology helped to extract the semantic information of the given text.

A similar idea has already been introduced by Svetlana Hansman and John Dunnion (2005) for the English language. In their paper, they described a semi-automatic construction of a conceptual graph representation of English texts using a combination of VerbNet and WordNet. Our approach differs from Hansman and Dunnion's by the fact that it processes the Arabic language and the building of an Arabic ontology that can be used not only for the analysis of the Arabic language, but also for several semantic-based Arabic applications such as reasoning, information retrieval, etc.

We believe that the presented approach is very promising and presents at least two advantages, it allows to i) semantically analyze Arabic text and ii) formalize the semantic information in a powerful and promising formalism which is the conceptual graphs.

This paper is organized as follows: section 2 introduces some tools used in the development of this work, section 3 describes our approach whereas the last section provides a conclusion and some future works.

II. Background

To achieve our goal, we used the Conceptual Graph formalism (Sowa, 2008) and had to integrate and use some tools/platforms that we introduce briefly in the following subsections.

1. Arabic WordNet

Arabic WordNet (AWN) is a free lexical resource for modern standard Arabic. It is based on the design and contents of Princeton WordNet (PWN) (Miller, 1995) and can be mapped onto PWN as well as a number of other wordnets, enabling translation on the lexical level to and from dozens of other languages. Arabic Wordnet groups Arabic words into sets of synonyms called synsets. It provides short, general definitions, and records the various semantic relations between these synonym sets.

2. Arabic VerbNet

Arabic VerbNet (AVN) is one of the first verb lexica, which classify the most used verbs of Modern Standard Arabic (MSA)[2]. The verb lexicon in its current state has 336 classes which contain 7744 verbs and 1399 frames providing information about verb root, verbal form, participle, thematic roles, sub-categorization frames and syntactic and semantic descriptions of verbs. Each AVN class defines a list of members (Arabic verbs), a list of possible thematic roles (Agent, Patient, etc.), and a list of frames (patterns) of how these semantic roles can be realized in a sentence.

3. *Stanford parser*

Stanford Parser[3] is a natural language parser developed by Dan Klein and Christopher D. Manning from The Stanford Natural Language Processing Group. The project contains a Java implementation of probabilistic natural language parsers; a graphical user interface is also available for parse tree visualization. Stanford Parser actually analyzes the three natural languages English, Chinese and Arabic. It generates a Treebank parse tree for the input sentence.

4. *Conceptual Graph*

Introduced by John Sowa (2008), Conceptual Graph (CG) is a powerful formalism for expressing knowledge and writing specifications, it allows expressing meaning in a form that is logically precise, humanly readable and computationally tractable. The CG formalism has been used in many areas (data base semantics, natural language processing, knowledge based systems, information systems, multi-agent systems, etc.).

Figure 1 shows an example of the CG for the sentence “The patient takes the food carefully.”.

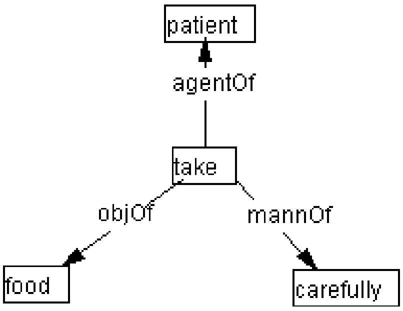


Fig 1. Example of a Conceptual Graph.

This CG can be read as: The “agent of” the action “take” is the “patient”, the “object” of the action is “the food” and the manner with which this action is performed is “carefully”.

Because of the power of this formalism and the advantages that can be gained in using it, it has been adopted in our work to express the extracted semantics.

5. *Amine Platform*

Amine Platform is a multi-layer integrated development environment (IDE), Developed by Kabbaj (2006, 2009), suited for symbolic programming, intelligent system programming and intelligent agents programming. It allows implementing many kinds of ontology-based intelligent systems such as Knowledge Based Systems, Ontology based applications, natural language processing applications, problem solving applications, planning applications, reasoning based applications, learning based applications and multi-agents applications.

Amine is a modular environment composed of seven layers: (i) Ontology layer; (ii) Knowledge-base layer, (iii) Algebraic layer; (iv) Memory-Based Inference and Learning Strategies Layer, (v) Programming layer; (vi) Agents and Multi-Agents Systems layer and (vii) Application layer. The reader is invited to refer to web site[4] of Amine for further information. Amine implements the Conceptual Graph formalism and provides APIs that can be used directly in any Java application, thanks to the high level of modularity and independence of its layers.

As mentioned by Kabbaj (2006), Amine is one of the best platforms that implement the CG formalism, it allows not only the creation and manipulation of CGs, but also their use in many ontology-based processes such as inferences, reasoning, etc. Furthermore, Amine is the platform developed and used in our research group.

III. The semantic analysis approach

Ontologies are among the resources that can allow computers understanding the meaning of texts and, in turn, leveraging their capabilities to develop more sophisticated applications for end users. This kind of resources are used in various fields including natural language processing, information retrieval, information system, semantic web, machine learning, data mining and knowledge representation.

As mentioned above, in our approach we start by building the Arabic ontology that leverages the contents of AWN and AVN. This ontology combines the high lexical coverage and semantic relations between words existing in AWN together with the formal representation of syntactic and semantic frames corresponding to verbs in AVN. We then proceed with the analysis of the Arabic sentences. For this purpose, we begin by using the Stanford Parser to syntactically parse and extract syntactic information of the Arabic text. Then the extracted (syntactic) information combined with the semantic information (provided by the built ontology) are used to extract the semantic information of the given Arabic text. This approach is detailed in the following two subsections.

1. The construction of the Arabic ontology

This section presents the first part of our work; we have developed an ontology with a high coverage of lexical terms as well as semantic representation of concepts. This ontology is useful for NLP applications especially Arabic QA systems. The main objective of this ontology is to make available an ontology allowing semantic representation of key concept meanings to support semantic reasoning.

The design of our ontology is structured around a concept hierarchy, lexical information and semantic frames (so called situations) about these concepts. The former are extracted from AWN, whereas the latter are represented based on the transformation of syntactic and semantic AVN frames into CGs.

The advantages of using ontology rather than a lexical database (like AWN) is that, in addition to concepts hierarchy and the relations between them, ontologies allow adding to each concept some extra semantic information such as definition, canon, rules and situations, which is very important for NLP applications such as question answering or reasoning. Furthermore, many ontology-based processes have already been developed within Amine platform, and the use of Amine ontology would save time and effort.
Figure 2 provides the top level of our Amine Arabic ontology. Below, a brief commentary on the specified categories:

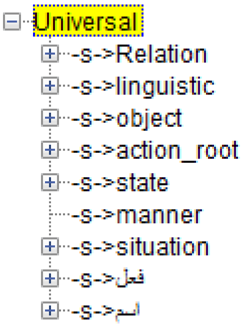
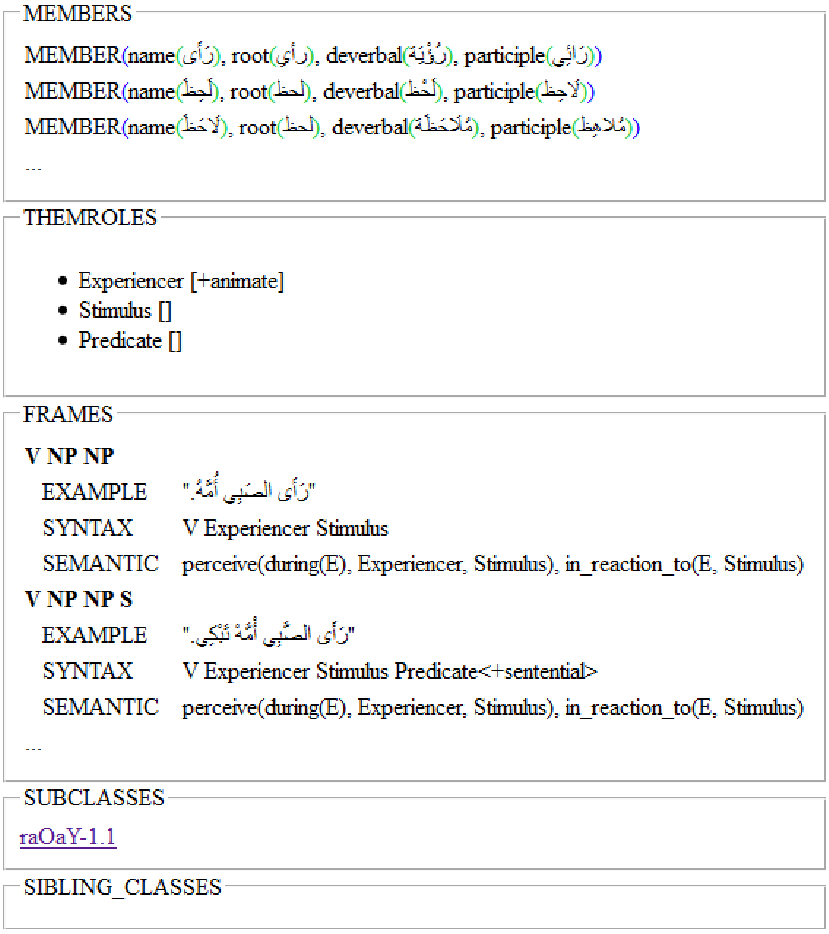


Fig 2. A snapshot of the Amine ontology

1. "فعل" branch encloses all verbs extracted from AWN,
2. "اسم" branch encloses all nouns extracted from AWN,
3. "State" branch contains all states used in the verbs situations (such as forbidden, alive, visible, etc.),

4. “Situation” branch contains all situations used in the verbs situations (such as legal, illegal, etc.),
5. “Manner”, an empty branch, designed to contain the manners.
6. “Action_root”, contains the elementary actions, used by AVN as predicate values (desire, transfer_info, perceive, etc.),
7. “Linguistic”, contains all linguistic concepts used in the verbs situations (verb, prep, adjective, etc.),
8. “Object”, contains all non-linguistic concepts used in the verbs situations (event, location, stimulus, etc.), and
9. “Relation” branch that contains all the relations used in the verb situations.

This ontology contains, for example, the verb “رأى” (to see) extracted from AWN with situations formulated in CG formalism corresponding to its syntactic frames extracted from AVN. Figures 3 and 4 respectively present a part of the AVN class “رأى” [raOaY-1] and a snapshot of the built ontology showing the node “رأى” with the content of one of its situations.



MEMBERS

MEMBER(name(رَأَى), root(رأي), deverbal(رُؤْيَة), participle(رَائِي))

MEMBER(name(لَحَظ), root(لحظ), deverbal(لَحْظ), participle(لاَحِظ))

MEMBER(name(لاَحَظ), root(لحظ), deverbal(مُلاَحَظَة), participle(مُلاَحِظ))

...

THEMROLES

- Experiencer [+animate]
- Stimulus []
- Predicate []

FRAMES

V NP NP

EXAMPLE "رَأَى الصَبِي أُمَّهُ."

SYNTAX V Experiencer Stimulus

SEMANTIC perceive(during(E), Experiencer, Stimulus), in_reaction_to(E, Stimulus)

V NP NP S

EXAMPLE "رَأَى الصَّبِي أُمَّهُ تَبْكِي."

SYNTAX V Experiencer Stimulus Predicate<+sentential>

SEMANTIC perceive(during(E), Experiencer, Stimulus), in_reaction_to(E, Stimulus)

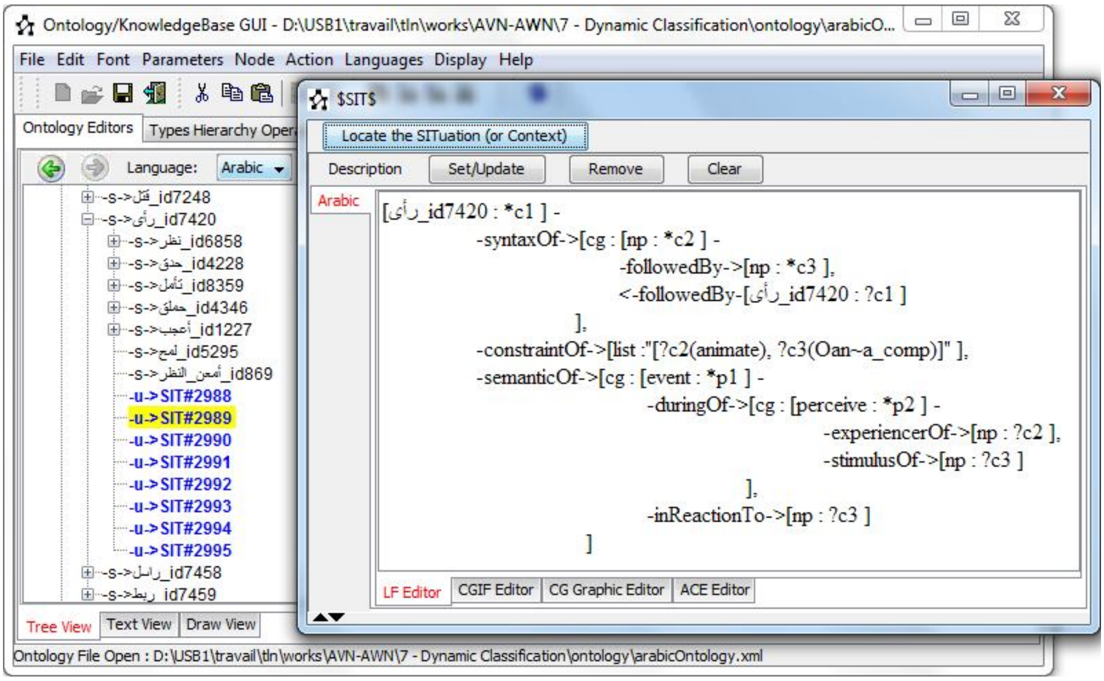
...

SUBCLASSES

raOaY-1.1

SIBLING_CLASSES

Fig 3. A snapshot of the AVN class “رأى” [raOaY-1]


Ontology/KnowledgeBase GUI - D:\USB1\travail\tln\works\AVN-AWN\7 - Dynamic Classification\ontology\arabicO...
File Edit Font Parameters Node Action Languages Display Help
Ontology Editors
Types Hierarchy Oper
Language: Arabic
Tree View
Text View
Draw View
SIT
Locate the SITuation (or Context)
Description
Set/Update
Remove
Clear
Arabic
LF Editor
CGIF Editor
CG Graphic Editor
ACE Editor

```
-s->قتل_id7248
-s->رأى_id7420
  -s->نظر_id6858
  -s->حدق_id4228
  -s->تأمل_id8359
  -s->حملق_id4346
  -s->أعجب_id1227
  -s->لمح_id5295
  -s->أمعن_النظر_id869
  -u->SIT#2988
  -u->SIT#2989
  -u->SIT#2990
  -u->SIT#2991
  -u->SIT#2992
  -u->SIT#2993
  -u->SIT#2994
  -u->SIT#2995
-s->راسل_id7458
-s->ربط_id7459
```

```
[رأى_id7420 : *c1 ] -
          -syntaxOf->[cg : [np : *c2 ] -
                        -followedBy->[np : *c3 ],
                        <-followedBy-[رأى_id7420 : ?c1 ]
                 ],
          -constraintOf->[list :"[?c2(animate), ?c3(Oan~a_comp)]" ],
          -semanticOf->[cg : [event : *p1 ] -
                        -duringOf->[cg : [perceive : *p2 ] -
                                        -experiencerOf->[np : ?c2 ],
                                        -stimulusOf->[np : ?c3 ]
                                    ],
                        -inReactionTo->[np : ?c3 ]
                 ]
```

Ontology File Open : D:\USB1\travail\tln\works\AVN-AWN\7 - Dynamic Classification\ontology\arabicOntology.xml

Fig 4. Snapshot of the Arabic ontology showing the node “رأى” with the content of one of its situations.

Each situation CG is made up of the main verb, linked to three subCGs conventionally named syntacticCg, constraintCg and semanticCg respectively with the three relations syntaxOf, constraintOf and semanticOf. Figure 5 shows the general design of each situation:

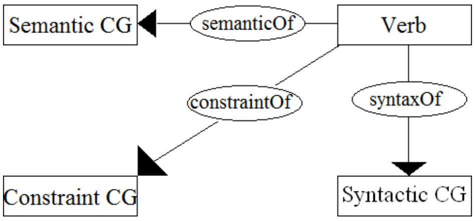


Fig 5. General design of the situation CGs

These subCGs include all information extracted from AVN:

- The syntacticCG represents the syntactic pattern of a given frame, such as:
  [verb: *c1]-followedBy->[NP: *c2]-followedBy->[NP: *c3]

- The constraintCG carries the constraints about the constituents of the syntacticCg, such as:
  [list : "[?c2(animate)]"]

- The semanticCG gives the semantic information (extracted from the semantic node of the AVN frame), for example:

```
[event : *p1 ] -
      -duringOf->[cg : [perceive : *p2 ] -
                        -experiencerOf->[np : ?c2 ],
                        -stimulusOf->[np : ?c3 ]
                  ],
      -inReactionTo->[np : ?c3 ]
```

The global CG is composed of these three subCGs (to be considered as situation of the verb “رأى”) is as follows:

```
[رأى : *c1 ] -
        -syntaxOf->[cg : [np : *c2 ] -
                    -followedBy->[np : *c3 ],
                    <-followedBy-[رأى : ?c1 ]
                 ],
        -constraintOf->[list :"[?c2(animate)]" ],
        -semanticOf->[cg : [event : *p1 ] -
                      -duringOf->[cg : [perceive : *p2 ] -
                                   -experiencerOf->[np : ?c2 ],
                                   -stimulusOf->[np : ?c3 ]
                               ],
                      -inReactionTo->[np : ?c3 ]
```

This CG can be read as follows: In a given sentence, if the **verb** (رأى) is **followed by** a Noun Phrase (NP) referenced by **c2** and this latter is **followed by** another NP referenced by **c3** (syntacticCG) and if the NP referenced by **c2** is an **animate** (constraintCG) then **c2 perceives c3 in reaction to c3** (semanticCG).

Here are some examples of Arabic sentences having this syntactic pattern:

Table 1: Examples of Arabic sentences having the syntactic pattern V NP NP.

Arabic sentences	**English translation**
رأى المتفرج المشهد	The spectator saw the scene
رأى الطفل القمر	The boy saw the moon.
قتل المجرم الضحية	The murder killed the victim.

2. Semantic analysis

The main goal of the semantic analyzer is to extract the semantics that corresponds to the given text. For this purpose, we opt for a five-step approach:

1. For each clause in the Arabic sentence we identify the main verb and build a syntactic CG which represents a sentence pattern using the parse tree.
2. For each clause, we localize the verb in the ontology and we extract all its situations (CGs), which contain the possible syntactic frames extracted from AVN together with the selectional restrictions (constraints).
3. We match the syntactic CG (that represents the sentence pattern) to each syntactic CG of the available situation CGs.
4. The result subset of situations (of the previous step) undergoes a filtering based on the constraint CG to retain at most one situation CG.
5. We extract the semantic CG from the situation and we instantiate it, by substituting the concepts called “np” in the (semantic) CG with the CG corresponding to the semantics of the noun phrase in the sentence. We consider in this paper simple sentences that have NPs restricted to only one word.

We describe in more details these steps in the following subsections:

2.1 Construction of the syntactic CG representing the pattern of the sentence

This step begins by syntactically parsing the sentence and producing the parse trees representing their clauses. We used for this purpose the Stanford parser.

For each clause of the sentence we construct a syntactic pattern, which identifies the main verb and the other main categories of the clause. For example, from the parse tree of the sentence:

رَأى الطِّفْلُ الْقَمَرَ.

(The boy saw the moon.)

We construct the following pattern:

V NP NP

From this pattern, we construct a syntactic CG as follows:

[verb: *c1] -followedBy->[NP: *c2] -followedBy->[NP: *c3]

During the construction of the syntactic CG, We keep in memory the mapping between the references of the concepts in the CG (c1, c2 and c3) and the sentence constituents they replace in the sentence (رأى, الطفل, القمر). This is useful for the other steps.

2.2 Extraction of all situations of the main verb from the Arabic ontology

In this step, we locate the verb in the ontology, we extract all its situations and consider them to be possible situations of the verb in the sentence. To the verb "رأى" are assigned, for example, 8 situations as shown in Table 2.

Table 2: Situations of the verb رأى in the Arabic ontology.

Situation 1:

```
[رأى : *c1 ] -
            -syntaxOf->[cg : [np : *c2 ] -
                              -followedBy->[np : *c3 ],
                              <-followedBy-[رأى : ?c1 ]
                         ],
        -constraintOf->[list:"[?c2(animate),?c3(Oan~a_comp)]"],
        -semanticOf->[cg : [event : *p1 ] -
                             -duringOf->[cg : [perceive : *p2 ] -
                                                -experiencerOf->[np : ?c2 ],
                                                -stimulusOf->[np : ?c3 ]
                                              ],
                             -inReactionTo->[np : ?c3 ]
                           ]
```

Situation 2:

```
[رأى : *c1 ] -
      -syntaxOf->[cg : [np : *c2 ] -
                              -followedBy->[np : *c3 ],
                              <-followedBy-[رأى : ?c1 ]
                         ],
      -constraintOf->[list:"[?c2(animate),?c3(kayofa_extract)]"],
      -semanticOf->[cg : [event : *p1 ] -
                                 -duringOf->[cg : [perceive : *p2 ] -
                                                -experiencerOf->[np : ?c2 ],
                                                -stimulusOf->[np : ?c3 ]
                                                ],
                                 -inReactionTo->[np : ?c3 ]
                       ]
```

Situation 3:

```
[رأى : *c1 ] -
          -syntaxOf->[cg : [np : *c2 ] -
                         -followedBy->[np : *c3 ],
                         <-followedBy-[رأى : ?c1 ]
                    ],
      -constraintOf->[list:"[?c2(animate),?c3(maA_extract)]"],
      -semanticOf->[cg : [event : *p1 ] -
                       -duringOf->[cg : [perceive : *p2 ] -
                                    -experiencerOf->[np : ?c2 ],
                                    -stimulusOf->[np : ?c3 ]
                                 ],
                       -inReactionTo->[np : ?c3 ]
                     ]
```

Situation 4:

```
[رأى : *c1 ] -
          -syntaxOf->[cg : [np : *c2 ] -
                         -followedBy->[np : *c3 ],
                         <-followedBy-[رأى : ?c1 ]
                    ],
          -constraintOf->[list :"[?c2(animate)]" ],
          -semanticOf->[cg : [event : *p1 ] -
                            -duringOf->[cg : [perceive : *p2 ] -
                                     -experiencerOf->[np : ?c2 ],
                                      -stimulusOf->[np : ?c3 ]
                                   ],
                            -inReactionTo->[np : ?c3 ]
                     ]
```

Situation 5:

```
[رأى : *c1 ] -
          -syntaxOf->[cg : [np : *c2 ] -
                         -followedBy->[np : *c3 ]-followedBy->[prep :"كَ" ]-followedBy->[np : *c5 ],
                         <-followedBy-[رأى : ?c1 ]
                    ],
          -constraintOf->[list :"[?c2(animate|organization), ?c5(!sentential), ?c3(concrete|organization)]" ],
          -semanticOf->[cg : [event : *p1 ] -
                            -duringOf->[cg : [characterize : *p2 ] -
                                                  -themeOf->[np : ?c3 ],
                                                  -predicateOf->[np : ?c5 ]
                                       ],
                            -causeOf->[np : ?c2 ]
                     ]
```

Situation 6:

```
[رأى : *c1 ] -
          -syntaxOf->[cg : [np : *c2 ] -
                         -followedBy->[np : *c3 ]-followedBy->[np],
                         <-followedBy-[رأى : ?c1 ]
                    ],
          -constraintOf->[list :"[?c2(animate), ?c4(participle)]" ],
          -semanticOf->[cg : [event : *p1 ] -
                            -duringOf->[cg : [perceive : *p2 ] -
                                                -experiencerOf->[np : ?c2 ],
                                                -stimulusOf->[np : ?c3 ]
                                       ],
                            -inReactionTo->[np : ?c3 ]
                     ]
```

Situation 7:

```
[رأى : *c1 ] -
          -syntaxOf->[cg : [np : *c2 ] -
                         -followedBy->[np : *c3 ]-followedBy->[np],
                         <-followedBy-[رأى : ?c1 ]
                    ],
          -constraintOf->[list :"[?c2(animate), ?c4(sentential)]" ],
          -semanticOf->[cg : [event : *p1 ] -
                              -duringOf->[cg : [perceive : *p2 ] -
                                                  -experiencerOf->[np : ?c2 ],
                                                  -stimulusOf->[np : ?c3 ]
                                         ],
                              -inReactionTo->[np : ?c3 ]
                      ]
```

Situation 8:

```
[رأى : *c1 ] -
          -syntaxOf->[cg : [np : *c2 ] -
                         -followedBy->[np : *c3 ]-followedBy->[prep :"فِي" ]-followedBy->[np],
                         <-followedBy-[رأى : ?c1 ]
                    ],
          -constraintOf->[list :"[?c2(animate)]" ],
          -semanticOf->[cg : [event : *p1 ] -
                              -duringOf->[cg : [perceive : *p2 ] -
                                                  -experiencerOf->[np : ?c2 ],
                                                  -stimulusOf->[np : ?c3 ]
                                         ],
                              -inReactionTo->[np : ?c3 ]
                      ]
```

2.3 Matching algorithm

The matching algorithm is based on two different steps: syntactic match and restriction match.

In the first step, we match the syntactic CG (generated from the sentence) against the syntactic CG of each situation (of the verb) extracted from the ontology. If the two syntactic CGs are equal then the algorithm considers this a possible match. Note that the algorithm may often return more than one match. This step gives as result in our example situations 1, 2, 3 and 4 only.

During the second step, we should filter the previous selected situations to keep only one (at most). This is done by applying the constraints contained in the constraint CG of each situation to the constituents of the generated syntactic CG. For example, a common requirement for the NP that immediately follows the verb is to be animate or organization, this check is done based on ontology super type relations.

The constraint CG of the previous example ([list: "[? c2(animate)]"]) means that the word referenced by c2 has to be an animate. From the kept mapping between the references and the concepts, we get the concept referenced by c2 and then we check that this concept is or not an animate.

The constraint parts of the retained situations are:

- Constraint CG of the situation 1: **list :"[?c2(animate), ?c3(Oan~a_comp)]"**
- Constraint CG of the situation 2: **list:"[?c2(animate),?c3(kayofa_extract)]"**
- Constraint CG of the situation 3: **list :"[?c2(animate), ?c3(maA_extract)]"**
- Constraint CG of the situation 4: **list :"[?c2(animate)]"**

This second step goal is to check whether:

The NP referenced by c2 (الطفل-boy) is or not animate and the NP referenced by c3 (القمر-moon) is a phrase that begins with أن (that) [Oan~a], begins with كيف (how) [kayofa] or begins with ما (what) [maA].

In our example, relying to the sentence structure and the ontology hierarchy, the word الطفل [The boy] is animate and القمر [the moon] is none of the three mentioned phrase types. So the constraint that matches with the types is the fourth one. We conclude then that the situation that matches with this sentence is the fourth one:

```
[رأى : *c1 ] -
        -syntaxOf->[cg : [np : *c2 ] -
                    -followedBy->[np : *c3 ],
                    <-followedBy-[رأى : ?c1 ]
                ],
        -constraintOf->[list :"[?c2(animate)]" ],
        -semanticOf->[cg : [event : *p1 ] -
                        -duringOf->[cg : [perceive : *p2 ] -
                              -experiencerOf->[np : ?c2 ],
                               -stimulusOf->[np : ?c3 ]
                             ],
                        -inReactionTo->[np : ?c3 ]
                    ]
```

2.4 Construction of the semantic CG

The situation CG that results from the previous process is made of three parts, the last one represents the semantic part. In this step, we firstly extract this part, then we replace the references that refer to concepts in the other parts (syntactic or constraint CGs) with copies of these concepts and we finally instantiate this CG, by substituting the concepts called “np” in the CGs with the CGs corresponding to the semantic of the noun phrases in the sentence. For this example, we simply substitute the NP concepts with the corresponding words in the sentence basing on the kept mapping between the references of the concepts in the CG (c1, c2 and c3) and the words in the sentence (رأى, الطفل, القمر).

The semantic CG extracted from the selected situation is:

```
[event : *p1 ] -
          -duringOf->[cg : [perceive : *p2 ] -
                          -experiencerOf->[np : ?c2 ],
                          -stimulusOf->[np : ?c3 ]
                        ],
         -inReactionTo->[np : ?c3 ]
```

After the instantiation of this semantic CG, it becomes:

```
[event : *p1 ] -
          -duringOf->[cg : [perceive : *p2 ] -
                          -experiencerOf->[الطفل : *c2 ],
                          -stimulusOf->[القمر  : *c3 ]
                        ],
         -inReactionTo->[القمر  : ?c3 ]
```

Which means that الطفل [the boy] is the experiencer of the action "perceive", under a stimulus which is القمر [to moon].

Conclusion and future works

In this paper, we presented our approach of semantic analysis of the Arabic texts. We opted for a two-step approach: In the first step we built an Arabic ontology leveraging the contents of Arabic WordNet and Arabic VerbNet, which presents the advantage to combine the high lexical coverage and semantic relations between words existing in AWN together with the formal representation of syntactic and semantic frames corresponding to verbs in AVN. In the second step, we semantically analyze the Arabic texts by using Stanford parser for syntactically parsing the Arabic text, tagging the constituents of the text with thematic roles and giving the sentence syntactic pattern. This syntactic information combined with the semantic information provided by the built ontology helped us to extract the semantic information of the given text.

The system is still in development and has not been tested on real cases. We are currently testing and improving it.

The presented approach is very promising and presents many advantages, such as:

- Allowing to semantically analyze Arabic texts.
- Giving the semantic information in conceptual graphs which is a powerful and promising formalism.
- Its ability to be adapted and used in question analysis and answer extraction components of QA systems.

We do not deny that this approach presents, on the other hand, some drawbacks:

- It highly depends on the Stanford parser which is a probabilistic parser.
- It depends on the semantics provided by Arabic VerbNet frames.
- It processes only with the use of Arabic VerbNet verbs which cover only 60.53% of Arabic WordNet verbs.
- The semantics in AVN frames uses a limited set of predicate values and does not retain the original verb, which makes the task harder for paraphrasing the extracted answer to communicate to the end user.

As future works, we aim to improve the semantic analyzer in order to process with questions too and integrate it in a semantic-based Question Answering system.

References

Abouenour, L., Nasri, M., Bouzoubaa, K., Kabbaj, A., and Rosso, P. (2013). ‘Construction of an ontology for intelligent Arabic QA systems leveraging the Conceptual Graphs representation’. Submitted to: *the Journal of Intelligent and Fuzzy Systems*.

Benajiba, Y. and Rosso, P. (2008). ‘Arabic named entity recognition using conditional random fields’ Proceedings of Workshop on HLT & NLP within the Arabic World, Marrakech, Morocco. Volume 8, 143-153.

Benajiba, Y., Rosso, P. and Lyhyaoui, A. (2007a). ‘Implementation of the ArabiQA Question Answering System's components’ Proceedings of Workshop on Arabic Natural Language Processing, 2nd Information Communication Technologies Int. Symposium, Fez, Morocco.

Benajiba, Y., Rosso, P. and Soriano, J. M. G. (2007b). ‘Adapting the JIRS Passage Retrieval System to the Arabic Language’ Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing, Mexico City, Mexico, 530-541.

Brini, W., Ellouze, M., Trigui, O., Mesfar, S., Belguith, HL. and Rosso, P. (2009). ‘Factoid and definitional Arabic question answering system’ Proceedings of NOOJ-2009, June 8-10, Tozeur, Tunisia.

Elkateb, S., Black, W., Vossen, P., Farwell, D., Rodríguez, H., Pease, A. and Alkhalifa, M. (2006). ‘Arabic WordNet and the challenges of Arabic’ Proceedings of Arabic NLP/MT Conference, London, UK.

Hammo, B., Abu-Salem, H. and Lytinen, S. (2002). 'QARAB: A question answering system to support the Arabic language' Proceedings of the ACL-02 workshop on Computational approaches to semitic languages, Association for Computational Linguistics, Philadelphia, PA, USA, 1-11.

Hensman, S. and Dunnion, J. (2005). 'Constructing conceptual graphs using linguistic resources' Proceedings of the 4th WSEAS international Conference on Telecommunications and informatics, Prague, Czech Republic, 1-6.

Kabbaj, A. (2006). 'Development of intelligent systems and multi-agents systems with amine platform' Proceedings of the 14th International Conference on Conceptual Structures, Conceptual Structures: Inspiration and Application, Aalborg, Denmark, 286-299.

Kabbaj, A. (2009). An overview of Amine, Conceptual Structures in practice, CRC Press, Chapman & Hall Book, (ed) P. Hitzler and H. Schärfe.

Klein, D. and Manning, CD. (2002). 'Fast exact inference with a factored model for natural language parsing' Proceedings of the 15th annual conference on Neural Information Processing Systems, British Columbia, Canada, 3-10.

Klein, D. and Manning, CD. (2003). 'Accurate unlexicalized parsing' Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, Sapporo Convention Center, Sapporo, Japan, Volume 1, 423-430.

Mohammed, F. A., Nasser, K. and Harb, HM. (1993). A knowledge based Arabic question answering system (AQAS), ACM SIGART Bulletin, 4(4), 21-30.

Mousser, J. (2010). 'A Large Coverage Verb Taxonomy for Arabic', Proceedings of 7th International Conference on Language Resources and Evaluation, Valletta, Malta.

Mousser, J. (2011). 'Classifying Arabic verbs using sibling classes' Proceedings of the 9th International Conference on Computational Semantics, p. 355-359, Association for Computational Linguistics, Oxford, UK.

Miller, GA. (1995). 'WordNet: a lexical database for English', *Communications of the ACM*, 38(11), 39-41.

Sowa, JF. (2008). Conceptual Graphs, Handbook of Knowledge Representation. 213-237.

[1] http://www.raya.com

[2] http://ling.uni-konstanz.de/pages/home/mousser/

[3] http://nlp.stanford.edu/software/lex-parser.shtml

[4] http://amine-platform.sourceforge.net/.