JWOLF: Java Free French Wordnet Library

Morad HAJJI[1], Mohammed QBADOU[2], Khalifa MANSOURI[3]
Laboratory SSDIA, ENSET Mohammedia
Hassan II University of Casablanca Mohammedia, Morocco

Abstract—**The electronic lexical databases WordNets have become essential for many computer applications, especially in linguistic research. Free French WordNet is an XML lexical database for French language based on Princeton WordNet for the English language and other multilingual resources. So far, research on Free French WordNet has focused on the construction and relevance of lexico-semantic information. However, no effort is made to facilitate the exploitation of this database under the Java language. In this context, this paper proposes our approach for the development of a new Java API based on Java Architecture for XML Binding. This Java API will make it easier for developers to exploit and use Free French WordNet to create applications for natural language processing. In order to assess the usefulness of our API, The API performance has been evaluated in the context of a Browser that we developed to extract semantic and lexical relations connecting synsets contained in this database, such as: the tree of hypernymy, the tree of hyponymy, synonyms, etc. The results showed that our API perfectly meets the needs of programmatically exploitation, exploration and consultation of this database in a Java application.**

Keywords—JAVA; API; WordNet; WOLF; JAXB; natural language processing

I. Introduction

The modern era is characterized by the importance of information to such an extent that we call it the information age. Indeed, we are witnessing the third industrial revolution: the digital revolution. This revolution radically transforms all areas. No one can deny the changes brought by this revolution to today's society. These changes affect every aspect of our lives to such an extent that it is difficult to identify an area where IT does not make its mark. The transformation is seen in the fields of economy, production, distribution, management, finance, marketing, consumption, health, agriculture, multimedia, education, etc.

In particular, the field of natural language processing is experiencing a rapid rise. This is due, on the one hand, to the technological evolution accomplished in the field of computers, on the other hand, to the progress made in the information processing models such as the electronic lexical databases (WordNets) and Artificial Intelligence.

Indeed, the field of electronic lexical databases (Wordnets) building has recently attracted increasing interest, because of their many applications in linguistic, psycholinguistic and computer research. Princeton University pioneered this field by developing the world's first WordNet namely Princeton WordNet (PWN) [2] for English. Over time, this lexical database has become unavoidable, the most developed and the most notorious. Indeed, the utility of this resource is confirmed in several areas, in this case the natural language processing, the extraction of information, the automatic construction of ontologies, the automatic translation, etc. Moreover, this base is embedded in the process of building the lexical databases of other languages by acting as a reference base. These WordNets have multiplied rapidly, the Global WordNet Association [6] lists more than 70 WordNets.

To our knowledge, research on French WordNet Free (WOLF) [7] has focused until now on the construction and relevance of the lexico-semantic information. Consequently, no effort is made to facilitate the programmatically use of this database under the Java language.

To overcome this problem, we propose through this paper a new approach for the development of JWOLF a Java API for access and exploitation of WOLF in order to facilitate the use of this database in programs developed with the Java language.

In addition, this article is part of our research work for the implementation of the systemic model that we have proposed in order to produce decision-making indicators from a corpus based on advances made in the of Semantic Web and Business Intelligence fields [1]. In particular, it is part of the "Construction of Ontology" phase of the proposed model in [1].

II. Princeton WordNet

Princeton WordNet is a project created and maintained by Princeton University [2]. It is an electronic lexical database for English Language. WordNet has the features of dictionaries and thesaurus. This base is built on the notion of 'concept'. In fact, words only serve to express messages conveying ideas composed of the senses. It is a semantic dictionary that lists words grouped by concepts related to each other. A group of words is called 'Synset' for 'Synonym Set' in English representing a given concept and connected to other groups of words. Hence, WordNet is more than just a thesaurus. Several semantic and lexical relations connect the synsets, among others are hyperonymy, hyponymy, meronymy, etc.

On the other hand, WordNet is a dictionary which the majority of synsets contain a definition of the concept they represent, in addition to simple examples of use in sentences. As part of WordNet, if a synset contains a definition, it is called 'gloss'. Therefore, WordNet is an electronic lexical semantic database for English. The WordNet database consists of several files listed by syntactic categories (grammatical classes). Typically, the data source files of this lexical database are divided into four syntactic categories, namely noun.dat, verb.dat, adj.dat and adv.dat. In fact, all synsets of the same syntactic category are listed in the same data file. Thus, the noun.dat file contains all the synsets representing the synonyms of the nouns, the verb.dat file contains all synsets representing the synonyms of the verbs, the adj.dat file contains all synsets representing the synonyms of the adjectives and finally the file adv.dat contains all synsets representing synonyms of adverbs.

Each synset includes an identifier, synonym words, relational pointers, a definition (gloss), and example for usage sentences.

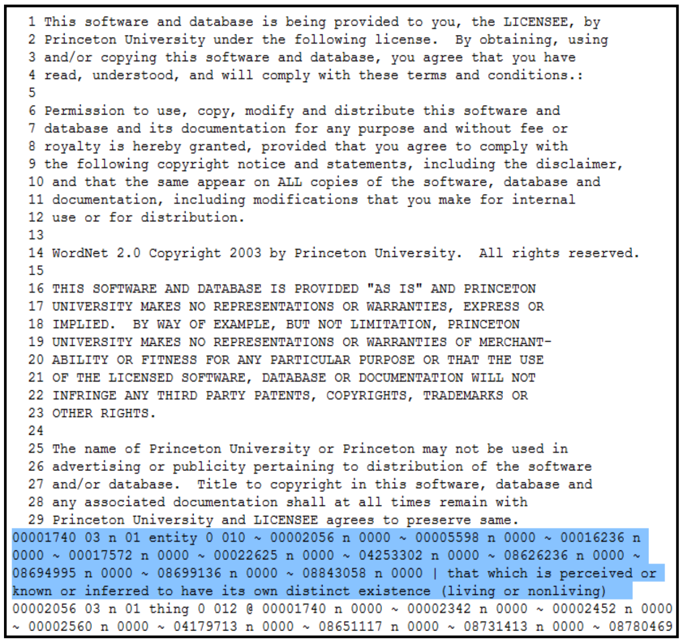
A word is represented in WordNet either in its orthographic form as individual word or in the form of a sequence of individual words linked by underscores. So, natural_object is a composed word that represents a unique concept.

The relationships between the synsets are encoded as relational pointers. Several relationships are defined within WordNet, among others are Antonym, Hyponym, Hypernym, Meronym, etc. Some of these relations are reflexive insofar as the existence of a relation between a synset X and another Y implies the existence of the inverse relation between Y and X. Indeed, if a synset X contains a relational pointer to another synset Y it implies that the synset Y contains the relational pointer opposite to X.

Synonymy is an implicit relation that links the words of the same synset since a synset is comprised of synonyms.

For each syntactic category, relational pointer types are represented by symbols. Indeed, for nouns, Antonymy is represented by the symbol '!', Hyponym is represented by the symbol '~', Hypernym is represented by the symbol '@ etc. In Fig. 1 we show the first synset belonging to the syntactic category representing nouns.

In addition, the WordNet database contains other files such as files with the extension .exc for exceptions, files with the extension .idx for indexes, and so on.



```
This software and database is being provided to you, the LICENSEE, by
Princeton University under the following license.  By obtaining, using
and/or copying this software and database, you agree that you have
read, understood, and will comply with these terms and conditions.:

Permission to use, copy, modify and distribute this software and
database and its documentation for any purpose and without fee or
royalty is hereby granted, provided that you agree to comply with
the following copyright notice and statements, including the disclaimer,
and that the same appear on ALL copies of the software, database and
documentation, including modifications that you make for internal
use or for distribution.

WordNet 2.0 Copyright 2003 by Princeton University.  All rights reserved.

THIS SOFTWARE AND DATABASE IS PROVIDED "AS IS" AND PRINCETON
UNIVERSITY MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, PRINCETON
UNIVERSITY MAKES NO REPRESENTATIONS OR WARRANTIES OF MERCHANT-
ABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE
OF THE LICENSED SOFTWARE, DATABASE OR DOCUMENTATION WILL NOT
INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR
OTHER RIGHTS.

The name of Princeton University or Princeton may not be used in
advertising or publicity pertaining to distribution of the software
and/or database.  Title to copyright in this software, database and
any associated documentation shall at all times remain with
Princeton University and LICENSEE agrees to preserve same.
00001740 03 n 01 entity 0 010 ~ 00002056 n 0000 ~ 00005598 n 0000 ~ 00016236 n
0000 ~ 00017572 n 0000 ~ 00022625 n 0000 ~ 04253302 n 0000 ~ 08626236 n 0000 ~
08694995 n 0000 ~ 08699136 n 0000 ~ 08843058 n 0000 | that which is perceived or
known or inferred to have its own distinct existence (living or nonliving)
00002056 03 n 01 thing 0 012 @ 00001740 n 0000 ~ 00002342 n 0000 ~ 00002452 n 0000
~ 00002560 n 0000 ~ 04179713 n 0000 ~ 08651117 n 0000 ~ 08731413 n 0000 ~ 08780469
```

Fig. 1. WordNet 2.0 Noun.Dat File.

III. Free French WordNet

The Free French WordNet (WOLF) is in XML (Extensible Markup Language) format used by the DebVisDic in the BalkaNet project. This format uses a scheme of document type definition (DTD). The graphical representation of this DTD is depicted in Fig. 2 as a hierarchical graph. This figure represents the hierarchy describing the structure of the debvisdic-strict.dtd file available on paper [3] converted to XML Schema.

However, the WOLF scheme is limited to the WN, SYNSET, ID, POS, SYNONYM, ILR, BCS, DEF and USAGE elements. The structure of WOLF is illustrated in Fig. 3.

Thus, WOLF is composed of a set of one or more SYNSETS. Each SYNSET includes an ID element, a SYNONYM element, and a DEF element. In addition, it can contain zero or more ILR elements, zero or one BCS element, zero or more USAGE elements. In Table 1, we give the meaning and use of the elements constituting the schema of the WOLF structure.

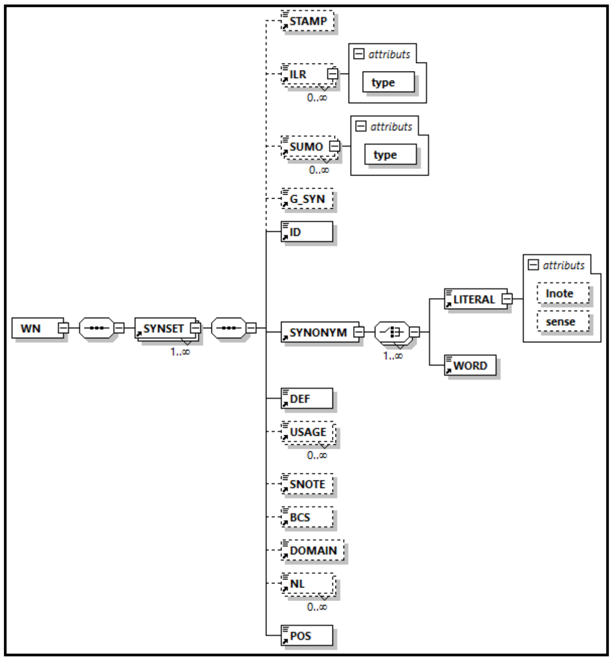


Fig. 2. Debvisdic-Strict Structure.

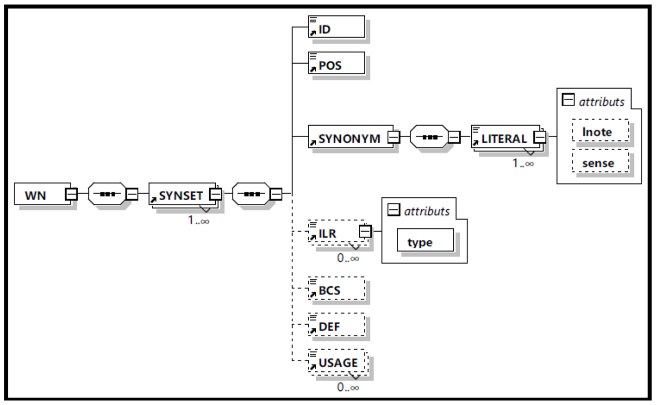


Fig. 3. WOLF Structure.

TABLE I. WOLF ELEMENTS

Element	Usage
WN	The root node representing the WordNet database itself.
SYNSET	Represents a set of synonyms.
ID	Represents the identifier of a SYNSET identical to that in Princeton's WordNet.
POS	Part of Speach (POS): Represents the grammatical nature of the synset words (syntactic category). Takes values N for noun, V for verb, A for adjective and b for adverb.
SYNONYM	Contains a list of synonyms words representing the same meaning.
ILR	Internal Language Relations (ILR): Includes a list of semantic links to other synsets. The links are based on the identifiers of the synsets. This tag has a 'type' attribute specifying the nature of the link between two synsets. See Fig. 4.
BCS	Basic Concept Sets (BCS): Sets of Basic Concepts that represents the importance of a synset.
DEF	Definition: Includes a small definition of the meaning that a synset represents.
USAGE	Includes a usage examples of the meaning that a synset represents.

The notion of pointers is adopted by WOLF and represented by relational links. While in the case of Princeton WordNet, the pointers are represented by symbols, within the context of the WOLF the relational links are filled via the ILR tag. Technically, the type of a relational link is indicated in the 'type' attribute of this tag in the form of a string. The value of this string gives the ID of the synset pointed to by this relation with the synset which contains this ILR tag. Fig. 4 depicts all types of relational links implemented by WOLF while Table 2 represents their meanings.

The WOLF lexical database relates the synsets according to relationships listed by syntactic categories in Table 3.

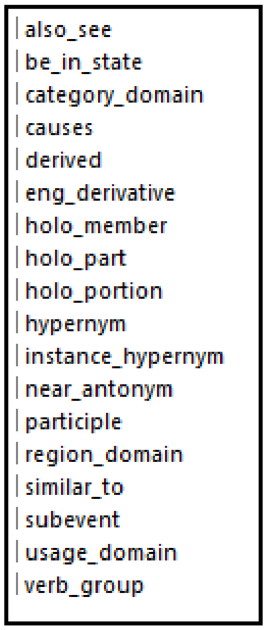

also_see
be_in_state
category_domain
causes
derived
eng_derivative
holo_member
holo_part
holo_portion
hypernym
instance_hypernym
near_antonym
participle
region_domain
similar_to
subevent
usage_domain
verb_group

Fig. 4. ILR Types List.

TABLE II. ILR TYPE SIGNIFICATION

ILR Type	Signification
verb_group	Verbs are grouped according to their meanings.
usage_domain	A relation between two concepts X and Y, insofar as Y represents the domain of use of X.
holo_part	A relation between two concepts X and Y as far as X is a part of Y.
also_see	also, a reference of weak meaning
instance_hypernym	A relation between two concepts X and Y with: Y is a type of X and Y is a root node of the hierarchy.
category_domain	Indicates the category of this word.
be_in_state	A relation between two concepts X and Y insofar as the concept X is qualified by the concept Y.
region_domain	A relation between two concepts X and Y to the extent where Y is a geographical or cultural domain of the concept X.
participle	A relation between an adjective X and a verb Y to the extent that X is the participle form of the verb Y.
near_antonym	A relation that links two concepts X and Y to the extent that X is the oposed of Y.
causes	A relation that links two verbs X and Y in which Y derives from X (causal relation).
hypernym	A relation between two concepts X and Y in which X is a type of Y. X is kind of Y.
eng_derivative	A word is derived from English.
similar_to	A relation between two synsets X and Y having the same meaning. As X is similar to Y.
subevent	A relation between two concepts X and Y in which X induces Y.
holo_member	A relationship between two concepts X and Y in which X is an element of Y.
derived	A relationship between two words X and Y in which X is derived from Y.
holo_portion	A relation between two concepts X and Y in which Y represents a portion of X.

TABLE III. RELATIONS PER SYNTACTIC CATEGORY

Syntactic category	Relations
Noms (Nouns)	holo_part, usage_domain, instance_hypernym, category_domain, near_antonym, be_in_state, hypernym, eng_derivative, holo_member, region_domain, holo_portion,
Verbes (Verbs)	verb_group, usage_domain, also_see, category_domain, near_antonym, causes, hypernym, eng_derivative, subevent, region_domain.
Adjectifs (Adjectifs)	usage_domain, also_see, participle, category_domain, near_antonym, be_in_state, eng_derivative, similar_to, region_domain, derived.
Adverbes (Adverbes)	usage_domain, category_domain, near_antonym, eng_derivative, region_domain, derived.

TABLE IV. SYNTACTIC CATEGORIES PER RELATIONSHIP

Relationship	Syntactic Categories
verb_group	Verb.
usage_domain	Noun, Verb, Adjective, Adverb.
holo_part	Noun
also_see	Verb, Adjective
instance_hypernym	Noun
category_domain	Noun, Verb, Adjective, Adverb.
be_in_state	Noun, Adjective
region_domain	Noun, Verb, Adjective, Adverb.
participle	Adjective,.
near_antonym	Noun,Verb, Adjective, Adverb.
causes	Verb
hypernym	Noun, Verb.
eng_derivative	Noun, Verb, Adjective, Adverb.
similar_to	Adjective.
subevent	Verb.
holo_member	Noun
derived	Adjective, Adverb.
holo_portion	Noun

The exploitation of these relations between the synsets makes it possible to construct tree structures. For example, the tree structure of hypernymy linking a word to these ancestors or categories according to all the senses that it has and all the syntactic categories to which it belongs. As WordNet, WOLF can be seen as a huge semantic network whose basic unit is synsets linked by lexical and semantic relations. Table 4 lists the syntactic categories by relationship. Thus the relationship 'hypernym' concerns only the category of nouns and the category of verbs.

IV. JAVA ARCHITECTURE FOR XML BINDING

Data binding refers to the mapping between classes of a program and data in an XML document. Just like object-relational mapping (ORM) solutions that perform the mapping between classes of a program and tables within a relational database, JAXB is a Java API that interfaces with an application program and an XML file to simulate the data contained in this file at the Java object-oriented level. It defines mappings between an XML schema or a DTD and classes in a Java program. JAXB is an abstraction layer between the object model and the XML model. JAXB provides a transparent way to link XML schemas with classes in Java programs that make it easy to manipulate and process XML data with Java. As illustrated in Fig. 5.

The binding process is based on two notions of marshalling and unmarshalling. The unmarshalling operation matches the contents of an XML file with the contents of a Java tree. While the marshalling operation is the reverse operation and it matches the contents of a Java tree with the contents of an XML file. As shown in Fig. 5. The linking process typically consists of seven steps: Class Generation, Class Compilation, Unmarshal, Generate Content Tree, Validate (Optional), Content Processing, and Marshal.

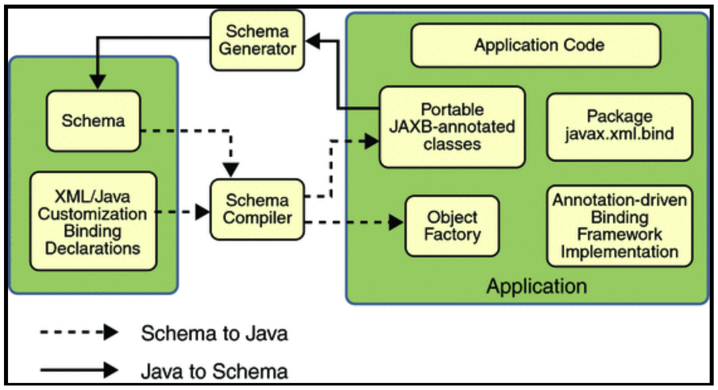


Fig. 5. JAXB Architectural Overview [5].

JavaBeans are generated with the XJC command (Xml-Java Compiler) shown in Fig. 5 by 'Portable JAXB-annotated' classes. In what follows, we present a part of the generated code relating to the class 'SYNSET'.

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "id",
    "pos",
    "synonym",
    "ilr",
    "bcs",
    "def",
    "usage"
})
@XmlRootElement(name = "SYNSET")
public class SYNSET {

    @XmlElement(name = "ID", required = true)
    protected String id;
    @XmlElement(name = "POS", required = true)
    protected String pos;
    @XmlElement(name = "SYNONYM", required = true)
    protected SYNONYM synonym;
    @XmlElement(name = "ILR")
    protected List<ILR> ilr;
    @XmlElement(name = "BCS")
    protected String bcs;
    @XmlElement(name = "DEF")
    protected String def;
    @XmlElement(name = "USAGE")
    protected List<USAGE> usage;

    /**
     * Gets the value of the id property.
     *
     * @return
     *     possible object is
     *     {@link String }
     *
     */
    public String getID() {
        return id;
    }
    …… // other methodes
}
```

The JAXB API entry point is materialized by the main JAXBContext class that provides transparent access to manage and manipulate unmarshal, marshal, and validate XML (which refers to Java binding operations).

```
public WOLF getWolfFromXml() {
   try {
         JAXBContext jc = JAXBContext.newInstance("org.hajji.jwolf.model");
         Unmarshaller unmarshaller = jc.createUnmarshaller();
         System.setProperty("javax.xml.accessExternalDTD", "all");
         WOLF wolf = (WOLF) unmarshaller.unmarshal(new File("wolf-
1.0b4.xml"));
         return wolf;
   } catch (JAXBException e) {
            e.printStackTrace();
            return null;
   }
}
```

V. JWOLF

The goal of JWOL is to simplify and accelerate the common tasks of language processing. It provides application support for seamless access to WOLF data. In fact, the understanding and perception of the WOLF structure and the notions that it is built up is the most important task.

A set of classes making up the library by providing a set of functions to access WOLF data and explore lexical and semantic relationships. During the development of this API we were inspired by JWNL [4] a Java API to access Princeton WordNet.

We have adopted the layered development model to benefit from the advantages it provides, namely the isolation of technical and business concerns, the substitution between layer implementations, the promotion of dependency management, etc. In Fig. 6, we illustrate the architectural hierarchy of JWOLF layers. Each layer uses and exploits the services offered by the layer that lie below it.

The JWOLF layer offers an abstraction of the notion of electronic lexical database whose class diagram of its model is illustrated in Fig. 7.

While developing the architecture of this API, we have targeted the following general design goals:

- Simplify the configuration and use of the API by requiring as little code as possible. One of the basic concepts of our approach is to make a self-configuration of this API via a property file written in XML.
- Provide transparent access to WOLF data by adopting an abstraction layer.

In Fig. 8, we show the representative classes of the JWOLF layered model.

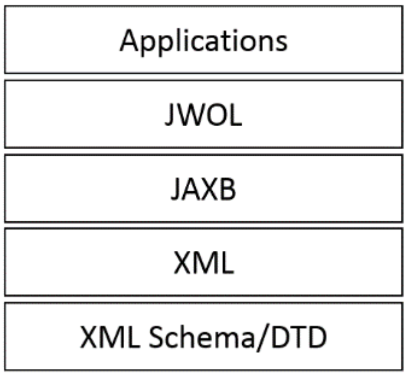


Fig. 6. JWOLF Architecture.

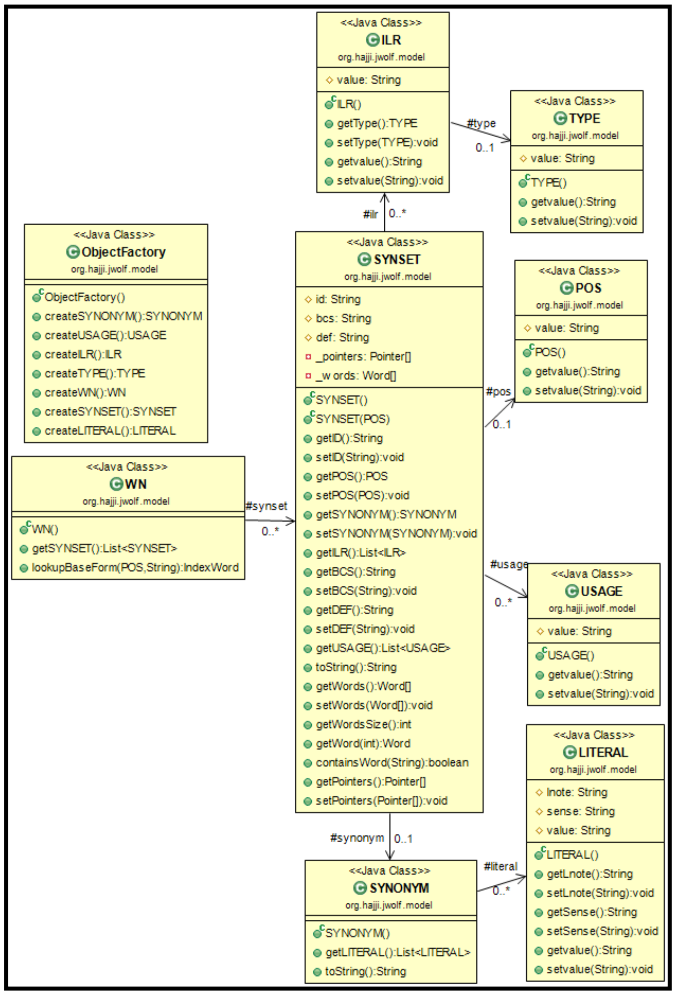


Fig. 7. JWOLF Model Class Diagram.

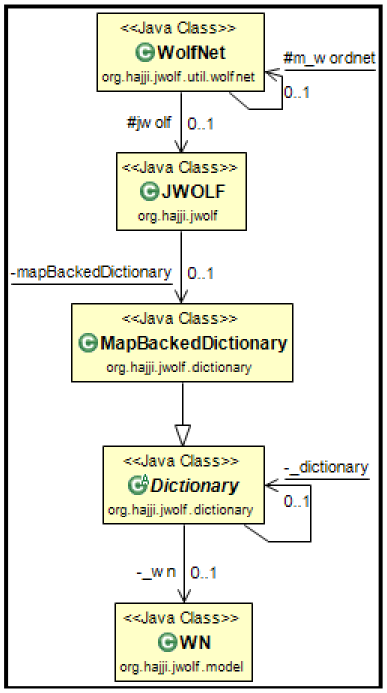

<<Java Class>> WolfNet org.hajji.jwolf.util.wolfnet
#m_wordnet 0..1
#jwolf 0..1
<<Java Class>> JWOLF org.hajji.jwolf
-mapBackedDictionary 0..1
<<Java Class>> MapBackedDictionary org.hajji.jwolf.dictionary
<<Java Class>> Dictionary org.hajji.jwolf.dictionary
-_dictionary 0..1
-_wn 0..1
<<Java Class>> WN org.hajji.jwolf.model

Fig. 8. Simplified Class Diagram of the JWOLF Layered Model.

The class 'WN' represents the notion of data binding provided by the API JAXB. The class 'Dictionary' represents an abstraction of the notion of the dictionary as much as the class 'MapBackedDictionary' constitutes the concretization and the implementation of this notion. The conceptual aspects of this API have evolved as it has been implemented. However, some basic principles persist inevitably. These can be summarized as follows:

- Interfaces providing read-only access to WOLF data.
- Offer an abstraction layer of the manipulation of these data
- Offer special features, such as searching for the meanings of a word, syntactic categories of a word, etc.
- Provide to the programmers, the access to networks of lexical and semantic relations of a word.

Since we aim to explore WOLF, the JWOLF API model provides access to this database through a number of read-only interfaces and class definitions. As a result, it does not provide functionality for changing WOLF data.

VI. WOLF Browser

In order to evaluate the elaborated JWOLF API and give an overview of the features it offers, we have developed a WOLF Browser, a tool for WOLF exploration. In fact, this tool constitutes the 'Presentation' layer according to the decomposition of a system according to a five-layer model whose architecture is presented in Fig. 9.

The search for the synonyms of a word using WOLF Browser is carried out via a graphical explorer interface designed specifically for this purpose. Using this explorer, users can access the sense trees represented by words in the form of synsets, the trees of lexical relations and the trees of semantic relations linking words with each other's according to their syntactic categories.

We show in Fig. 10 the use of this tool to extract the hypernymy tree for the word 'Reporter' as part of the syntactic category of nouns. In fact, this word belongs to two syntactic categories namely 'noun' and 'verb'. This word is a homograph to the extent that it has different meanings while having the same graphic form.

WOLF can be used for the development of natural language processing (NLP) applications. Thus, JWOLF as Java API will allow developers to more easily use Java to create NLP applications. In fact, JWOLF provides other features such as relationship discovery and morphological processing. It can be used for searching the synonyms of a given word, the extraction of the relations of a given type linking a given word to the synsets contained in WOLF (for example, obtain the tree of hypernymy of a given word).

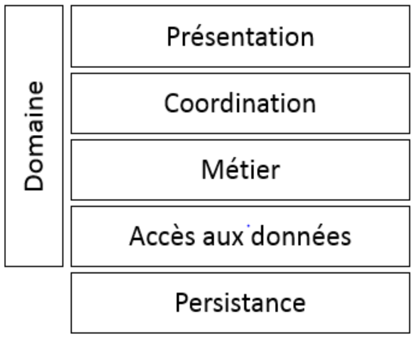
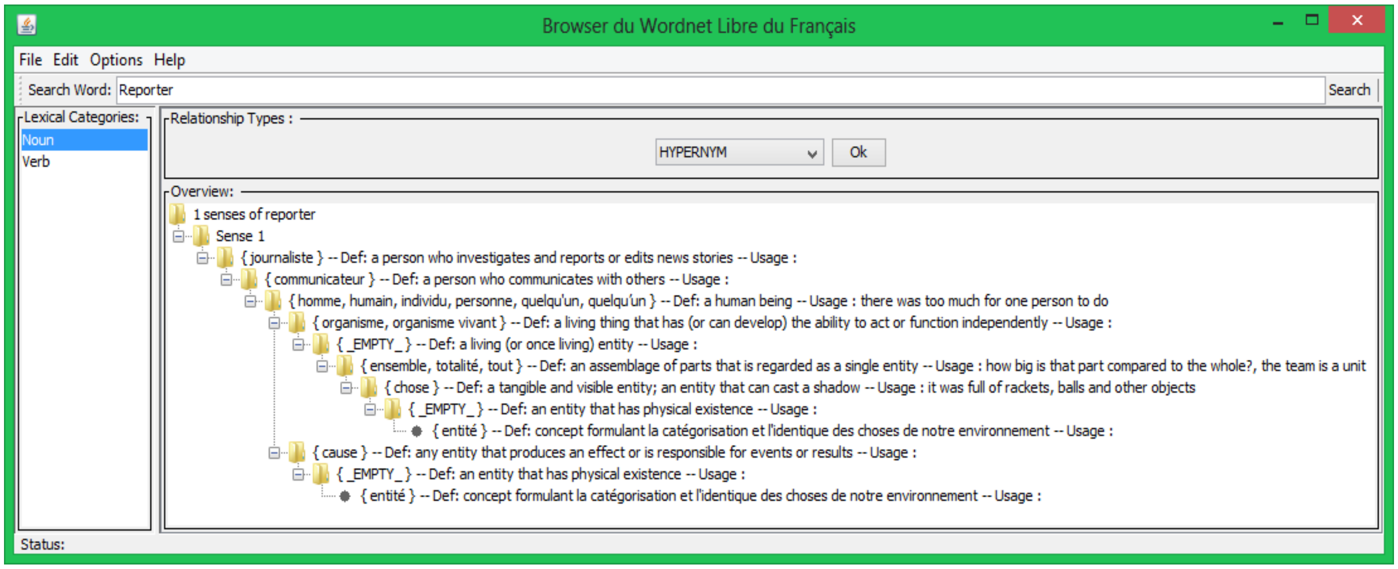


Fig. 9. Layered Model.



Search Word: Reporter

Lexical Categories:
Noun
Verb

Relationship Types : HYPERNYM

Overview:

```
1 senses of reporter
  Sense 1
    { journaliste } -- Def: a person who investigates and reports or edits news stories -- Usage :
      { communicateur } -- Def: a person who communicates with others -- Usage :
        { homme, humain, individu, personne, quelqu'un, quelqu'un } -- Def: a human being -- Usage : there was too much for one person to do
          { organisme, organisme vivant } -- Def: a living thing that has (or can develop) the ability to act or function independently -- Usage :
            { _EMPTY_ } -- Def: a living (or once living) entity -- Usage :
              { ensemble, totalité, tout } -- Def: an assemblage of parts that is regarded as a single entity -- Usage : how big is that part compared to the whole?, the team is a unit
                { chose } -- Def: a tangible and visible entity; an entity that can cast a shadow -- Usage : it was full of rackets, balls and other objects
                  { _EMPTY_ } -- Def: an entity that has physical existence -- Usage :
                    { entité } -- Def: concept formulant la catégorisation et l'identique des choses de notre environnement -- Usage :
          { cause } -- Def: any entity that produces an effect or is responsible for events or results -- Usage :
            { _EMPTY_ } -- Def: an entity that has physical existence -- Usage :
              { entité } -- Def: concept formulant la catégorisation et l'identique des choses de notre environnement -- Usage :
```

Fig. 10. WOLF Browser.

VII. Conclusion

This research paper provided an approach for the development of JWOLF; a Java API to improve and facilitate access and exploration of the data listed in WOLF.

The proposed approach consisted of identifying the structural features of Princeton WordNet and that of WOLF. The study of all the architectural constituents of WOLF highlighted the compositional specificities of this linguistic base. Hence, a presentation of the data binding approach offered by the JAXB Java API for manipulating data in XML files with the Java programming language has been elaborated.

The research methodology adopted in this paper was concretized by the implementation of the JWOLF to explore the WOLF data, extract lexical relations and semantic relations between the synsets it contains.

In order to assess the usefulness and the benefits offered by our API, we developed WOLF Browser, a tool allowing to access the linguistic data contained in WOLF and to explore relational trees that this API allows to extract.

The results of this work showed that our API represents a significant improvement in the exploration manner of the WOLF and a considerable optimization of using this database programmatically. It can be seen as a brick that can be added to the extraordinary building of Java libraries. It has been developed to provide a higher level of abstraction, reducing the effort for using WOLF in a Java programming environment.

In future works, we will utilize this API in the ontology extraction process from a corpus.

References

[1] M. HAJJI, M. QBADOU, K. MANSOURI, "Proposal for a new Systemic Approach of Analitical Processing of Specific Ontology to Documentary resources: Case of Educational Documents", Journal of Theoretical and Applied Information Technology, July 2016, Vol.89, No.2, pp. 481-51.

[2] Fellbaum, C., WordNet. An Electronic Lexical Database. MIT Press. 1998.

[3] “InriaForge : Wordnet Libre du Français : Liste de fichiers du projet Wordnet Libre du Français.” [Online]. Available: https://gforge.inria.fr/frs/?group_id=1177&release_id=7690. [Accessed: 31-Jan-2019].

[4] B. Walenz and J. Didion, “JWNL (Java WordNet Library),” SourceForge. [Online]. Available: https://sourceforge.net/projects/jwordnet/. [Accessed: 31-Jan-2019].

[5] “Chapter 17 Binding between XML Schema and Java Classes (The Java EE 5 Tutorial),” Oracle. [Online]. Available: https://docs.oracle.com/cd/E19316-01/819-3669/bnazf/index.html. [Accessed: 31-Jan-2019].

[6] “The Global WordNet Association,” The Global WordNet Association. [Online]. Available: http://globalwordnet.org/. [Accessed: 31-Jan-2019].

[7] B. Sagot, D. Fiser, "Building a free French wordnet from multilingual resources", OntoLex. Marrakech Morocco, *2008.*