WordNet framework improvements for NLP: Defining abstraction and scalability layers

Pedro Fialho Sérgio Curto Ana Cristina Mendes
Luísa Coheur
Instituto Superior Técnico, Technical University of Lisbon
Spoken Language Systems Lab./INESC-ID Lisboa

September 06, 2011

Abstract

The WordNet knowledge model is currently implemented in multiple software frameworks providing procedural access to language instances of it. Frameworks tend to be focused on structural/design aspects of the model thus describing low level interfaces for linguistic knowledge retrieval. Typically the only high level feature directly accessible is word lookup while traversal of semantic relations leads to verbose/complex combinations of data structures, pointers and indexes which are irrelevant in an NLP context. Here is described an extension to the JWNL framework that hides technical requirements of access to WordNet features with an essentially word/sense based API applying terminology from official interfaces. This high level API is applied to the original English version of WordNet as stored in filesystem, memory and database by JWNL's distinctive conversion capabilities. JWNL usage/implementation is also briefly described as a complement for available documentation.

1 Introduction

Linguistic thesaurus and dictionaries are useful in Natural Language Processing (NLP) tasks requiring meaning/context enrichment of single or multiple word expressions. WordNet[7] provides thesaurus and dictionary functionality through a knowledge model available freely in official interfaces (online/web and offline/filesystem). Frameworks (also known as local interfaces) are available in many programming languages and they all parse the offline knowledge representation into data structures (abstraction layer). These are mostly aimed at representing WordNet's knowledge model therefore lacking high level layers for word based information retrieval.

JWNL[1] (acronym for Java WordNet Library) is one such framework featuring tools for converting a WordNet filesystem representation into an in-memory

[1]`http://sourceforge.net/projects/jwordnet/`

map or SQL database accessible with a generic (storage independent) API. The main word level feature available in JWNL is lookup requiring a string and respective part-of-speech (POS) to retrieve the corresponding WordNet entry. Like other WordNet frameworks JWNL features word stemming, semantic relationship and word sense traversal without such a simple access method as such it's desirable to have a black-box approach to linguistic features already reachable in a too verbose/complex manner irrelevant for the NLP developer and/or the NLP application's logic.

Here is described an NLP component using linguistic information from WordNet with JWNL as access framework and development starting point. Existing features are compiled into procedures/getters (named with terminology from official interfaces) requiring only a word/sense. For textual code/API description some reading assumptions are made (such as *Name(properties)* for objects and *Object.name(parameters)* for procedures) therefore only relevant information is represented. Java class names are stated with full package description while JWNL's classes are single italic uppercased words (parameters may include JWNL's objects otherwise being lowercased). Section 2 briefly describes starting point concepts and components; section 3 details changes in the framework's code, developed getter's semantics and applied tests; section 4 refers intended usage, subjective value and future work.

2 Development baseline

The development starting point of this NLP targeted extension is focused on specific components of WordNet version 3.0 (web and filesystem) and JWNL version 1.4.1 as below described.

2.1 WordNet

WordNet is a directed acyclic graph composed of word forms and meanings related in a many to many manner. As thesaurus WordNet describes lemmas of single or multiple words grouped in synsets (sets of synonyms identifying concepts/senses) and interlinked by lexical/morphological and semantic relations. Synsets also contain linguistic information typically found on dictionaries (such as gloss, POS and usage examples) and are uniquely identified by WordNet specific offsets (database locations changeable among versions). Common morphological derivations are mapped to the correspondent lemmas in lists of exceptions.

2.2 JWNL

Usage of the developed extension requires acknowledgment of essential JWNL components below described and contextualized with other WordNet related components hidden by this extension. This description should complement available manuals[2], API[3] and related works[5, 2].

Storage systems

The feature that most distinguishes JWNL from other frameworks is the ability to convert a WordNet filesystem representation into SQL tables/views or a serialized *java.util.Map* through specific Java procedures/tools defined in the *net.didion.jwnl.utilities* package. Each storage system corresponds to a concrete implementation of the *Dictionary* abstract class eventually defining managers for storage specific resources. For database access the JDBC driver connection/registration is wrapped in a *ConnectionManager* for use by *DatabaseManagerImpl* (main database resource manager containing hardcoded SQL queries) while Map backed access does not require a resource manager since all information is on a single in-memory data structure thus being directly accessible. WordNet information is accessed through a storage independent API (due to usage of Java's inheritance mechanisms) thus hiding JDBC or Map details.

Initialization

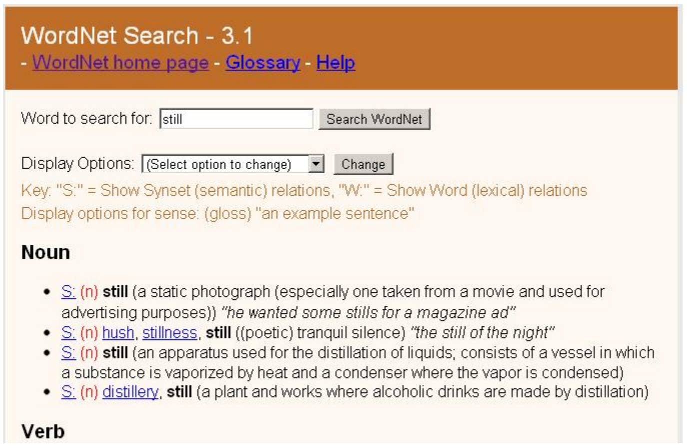
JWNL defines an initialization phase for loading appropriate Java constructs according to the WordNet access/manipulation profile specified in a mandatory configuration XML which defines target format, version, allowed morphological operations (language dependent) and storage specific parameters (such as database connection login or WordNet filesystem location). Some characters must be escaped due to XML parsing rules, namely "&" (used in a database URL) should be escaped as "&amp". Handlers for storage or language dependent operation/configuration are specified in XML elements with properties/arguments in child nodes although bundled examples should suffice for out-of-the-box usage/understanding.

Information retrieval

Upon initialization becomes available an instance of the *Dictionary* class which is the main access link to the WordNet knowledge base defining word based lookup procedures available in JWNL. These also require a POS and return an *IndexWord* object referencing the senses/synsets of the specified lemma and POS as described in the online interface example of figure 1. Senses are then obtained with *IndexWord.getSenses()* resulting an array of *Synset* objects which are a kind/ subclass of *PointerTarget*. A specific sense can be obtained with *Dictionary.getSynsetAt(POS, offset)* when a synset's offset is known. Words are also represented as a subclass of *PointerTarget* but can only be obtained by cast (to *Word*).

The set of *PointerTarget* objects representing WordNet's knowledge units are interlinked by lexical and semantic pointers. Available links are obtained with

[2]`http://sourceforge.net/apps/mediawiki/jwordnet/`
[3]`http://nlp.stanford.edu/nlp/javadoc/jwnl-docs/`


WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: still

Display Options: (Select option to change)

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **still** (a static photograph (especially one taken from a movie and used for advertising purposes)) *"he wanted some stills for a magazine ad"*
- S: (n) hush, stillness, **still** ((poetic) tranquil silence) *"the still of the night"*
- S: (n) **still** (an apparatus used for the distillation of liquids; consists of a vessel in which a substance is vaporized by heat and a condenser where the vapor is condensed)
- S: (n) distillery, **still** (a plant and works where alcoholic drinks are made by distillation)

Verb

Figure 1: Online interface lookup for "still" grouped by POS. In JWNL, the group "Noun" is referenced by an *IndexWord("still", POS.NOUN)* object with the illustrated senses as an array of *Synset* objects.

PointerTarget.getPointers() resulting an array of *Pointer* objects with link's properties such as group (lexical/semantic), type (one of the static fields in the *PointerType* class) and targets. Link retrieval procedures are overloaded for *PointerType* filtering although higher level relationship retrieval is available with *PointerUtils*. It is also possible to find relationships between two synsets with *RelationshipFinder.findRelationships(Synset, Synset, PointerType)* which returns a list of *Relationship* objects (essentially synset/pointer/synset triples). Terminology used in *PointerType* and *PointerUtils* may differ from official interfaces.

Stemming

For stemming purposes the *Dictionary* instance provides a *MorphologicalProcessor* allowing broader word coverage according to techniques specified in the configuration XML such as morphological reduction and/or exception lists lookup. Language specific garbage strings such as detachable suffixes (for gerund and plural resolution) and token delimiters can be defined directly in the XML along with lookup operations available for reduced word forms. Main procedure is *MorphologicalProcessor.lookupBaseForm(POS,expression)* returning the *IndexWord* for the first lemma found within the specified POS matching the specified expression (single or multi-word) according to the reduction/lookup rules defined in the XML.

3 Improvements

The developed extension is named *JWNLSimple* and works as a Java object requiring a configuration XML (used in JWNL) as argument thus allowing multiple WordNet formats to coexist on the same program by instantiation of *JWNLSimple* objects with distinct arguments. Essential JWNL resources (such as *Dictionary* and *MorphologicalProcessor*) are set with *JWNLSimple.init()* and released with *JWNLSimple.close()* (specially meaningful for databases) thus imposing a development/usage life cycle.

3.1 Patches

JWNL required patches on database access components which were the most reviewed since this was the main storage system used during development. Namely the "Exception" view was added to the SQL script bundled with JWNL (stated in the hardcoded SQL queries of *DatabaseManagerImpl*) and the "offset" field on a synset retrieval query was renamed to "file_offset" (according to the synset's table definition).

Connection reuse

Query execution for database operations (such as word lookup) involves connecting/disconnecting the database driver thus being inefficient and error prone on batch usage. For query execution a class must construct a new *ConnectionManager* object (requires driver, URL, username and password for database connection) and call *ConnectionManager.getQuery(sql)* to get a *Query* object containing a *java.sql.Connection* obtained (internally) from *ConnectionManager.getConnection()* — essentially a *java.sql.DriverManager.getConnection(url)* call. This getter always starts a new database connection that will be closed with *Query.close()* however sequential calls to lookup procedures have shown[4] failures.

In order to keep the connection alive a global variable was added to the *ConnectionManager* representing the database connection thus *ConnectionManager.getConnection()* only starts a new connection when this variable is *null*, the connection is closed or not valid (unresponsive within a time period) otherwise returning the global variable/connection. The *Query.close()* was also changed to avoid connection closing (maintaining statement and result set closure) upon each query. As such a single connection is now used for each database backed *JWNLSimple* that should be closed with *JWNLSimple.close()*.

3.2 Features

Alternative *Synset* retrieval procedures are made available with this extension since developed getters for semantic features require a *Synset* object. For usage

[4]Observed on (capable) low-end Hardware without a failure pattern nor constant code location.

independence from JWNL only basic WordNet data structures (such as *Synset* or *Word*) were used as input/output containers. Overall result organization and coverage is based on the online interface.

3.2.1 Lookup and Stemming

Developed stemming (and some word lookup) procedures cover all POS providing a larger search space. These are used internally by the below described word lookup procedures.

- wordnetStemmer(expr) — a *MorphologicalProcessor.lookupBaseForm(POS,expression)* for all *POS* returning the first found stemmed form (having the same number of words in case of a multi-word expression) as a *java.lang.String* or the original expression if none found.
- wordnetStemmerAll(expr) — similar to *JWNLSimple.wordnetStemmer(expr)* instead returning all found forms as a *java.util.LinkedHashSet* of *IndexWord* (input excluded).
- getAllSynsets(expr) — a *Dictionary.getIndexWord(POS,expr)* for all *POS* covering the original and stemmed form of the specified expression (single or multi-word) and returning a *java.util.LinkedHashSet* of *Synset* objects.
- getAllSynsets(pos, expr) — a *JWNLSimple.getAllSynsets(expr)* for the specified *POS*.
- getAllSynsetsByPOS(expr) — similar to *JWNLSimple.getAllSynsets(expr)* instead returning a *java.util.HashMap* with *POS* objects as keys and a *Synset* set as value.
- getAllSynsetsLiteral(expr) — a *JWNLSimple.getAllSynsetsByPOS(expr)* using *POS* labels and *Synset* lemmas (both *java.lang.String*, space separated in case of a multi-word expression) in the resulting map.
- findRelationshipsAll(Synset, Synset) — a *RelationshipFinder.findRelationships(Synset, Synset, PointerType)* for all pointer types returning a *java.util.ArrayList* of *Relationship*.

3.2.2 Semantic/Linguistic

Getters for relation's targets were developed having in common the argument and result type (*Synset* and *java.util.LinkedHashSet* respectively) with the latter being composed of (unique) *Word* or *Synset* objects. Applied terminology and result coverage are based on the online interface as such each getter is named with "get" followed by a relation's name (as appearing in WordNet and below itemized) and to each relation of a synset corresponds a set of words or synsets (each element related to a synset's lemma). Some relations observed on the online interface required using *JWNLSimple.findRelationshipsAll(Synset, Synset)* for pointer name matching in JWNL (*Synset* objects obtained by offset lookup).

PointerTarget list parsers

The following relation's targets were obtained from existent JWNL procedures by parsing the returned *PointerTarget* list representation into a *java.util.LinkedHashSet*.

- Member Holonym — group where the sense belongs; Example: "forest" is a member holonym of "tree". JWNL provides *PointerUtils.getMemberHolonyms(Synset)* returning a JWNL specific list here parsed into a *java.util.LinkedHashSet*.
- Member Meronym — the inverse of a member holonym. A *java.util.LinkedHashSet* version of *PointerUtils.getMemberMeronyms(Synset)*.
- Part Holonym — whole element made with the sense; Example: "car" is a part holonym of "window". A *java.util.LinkedHashSet* version of *PointerUtils.getPartHolonyms(Synset)*.
- Part Meronym — the inverse of a part holonym. A *java.util.LinkedHashSet* version of *PointerUtils.getPartMeronyms(Synset)*.
- Entailment — implication of a sense; Example: "inhale" is an entailment of "smoke". A *java.util.LinkedHashSet* version of *PointerUtils.getEntailments(Synset)*.
- Attribute — quantifier of a sense; Example: "weight" is an attribute of "heavy". A *java.util.LinkedHashSet* version of *PointerUtils.getAttributes(Synset)*.
- Verb Group — equivalent verb sense (usually with the same lemmas) denoting an higher abstraction level; Example: "trace" is a verb group of "watch". A *java.util.LinkedHashSet* version of *PointerUtils.getVerbGroup(Synset)*.
- Cause — non causative counterpart of a verb; Example: "burn" is a cause of "ignite". A *java.util.LinkedHashSet* version of *PointerUtils.getCauses(Synset)*.
- See Also — alternate/equivalent version of a sense; Example: "many" is a see also of "more". A *java.util.LinkedHashSet* version of *PointerUtils.getAlsoSees(Synset)*.
- Direct Hypernym — category of a sense; Example: "tree" is a direct hypernym of "palm". A *java.util.LinkedHashSet* version of *PointerUtils.getDirectHypernyms(Synset)*.
- Direct Hyponym — type/kind of a sense; Example: "tiger" is a direct hyponym of "cat". A *java.util.LinkedHashSet* version of *PointerUtils.getDirectHyponyms(Synset)*.
- Sister Term — member of the direct hyponynms of some sense where the input sense also belongs; Example: "man" is a sister term of "woman" (both hyponyms of "adult"). A *java.util.LinkedHashSet* version of *PointerUtils.getCoordinateTerms(Synset)*.

PointerTarget tree parsers

The following relation's targets were also obtained from already existent JWNL procedures instead returning an internal representation of a *PointerTarget* tree here traversed with overflow precautions. JWNL trees are subgraphs of WordNet for a single relation (no multiple inheritance) although developed getters return a set/bag of words thus discarding tree structure.

- Similar To — characterization of a sense; Example: "calm" is a similar of "quiet". Observed similars match the root node's childs of *PointerUtils.getIndirectAntonyms(Synset)* (also match *PointerType.SIMILAR_TO*) here gathered into a *java.util.LinkedHashSet*.

- Antonym — opposite of a sense (direct or through a similar); Example: "naive" is an antonym of "sophisticated". Obtained from *PointerUtils.getAntonyms(Synset)* if the resulting list is not empty (direct antonyms) otherwise a recursively parse of *PointerUtils.getIndirectAntonyms(Synset)* with similars removed (antonyms of similars).

- Inherited Hypernym — the category of a sense eventually with other hypernyms in between; Example: "time period" is an inherited hypernym of "May". All the synsets recursively gathered from *PointerUtils.getHypernymTree(Synset, PointerUtils.INFINITY)* if result overflow is not reached otherwise a *JWNLSimple.getDirectHypernyms(Synset)*.

- Full Hyponym — the type/kind of a sense eventually with other hyponyms in between; Example: "magenta" is a full hyponym of "color". A *java.util.LinkedHashSet* version of *PointerUtils.getHyponymTree(Synset)*.

PointerType based

The following relation's targets were obtained by extracting words or synsets (as observed in the online interface) related to an input synset by the stated pointer types.

- Pertainym — the broad category of a sense; Example: "music" is a pertainym of "musical". A *java.util.LinkedHashSet* of *Word* objects connected to the input *Synset* by a *PointerType.DERIVED* link.

- Derivationally Related Form — the result of attaching derivational affixes to a stem (a subset of fuzzynyms[4]); Example: "fraternity" is a derivationally related form of "fraternal". A *java.util.LinkedHashSet* of *Word* objects connected to the input *Synset* by a *PointerType.NOMINALIZATION* link.

- Phrasal Verb — a multi word expression containing the verb (original meaning may change); Example: "run away" is a phrasal verb of "run". Phrasal verbs observed in the online interface match *PointerType.SEE_ALSO* link targets.

- Domain Usage — the environment/discourse type where a sense is applied[1]; Example: “irony” is a domain usage of “pretty” (as in “pretty mess”). Domain usages are obtained from *PointerType.USAGE* links.
- Domain Term Usage — the inverse of a domain usage. Obtained from *PointerType.USAGE_MEMBER* links.
- Domain Category — the type of domain where a sense is applied; Example: “physics” is a domain category of “light”. Domain categories correspond to *PointerType.CATEGORY*.
- Domain Term Category — the inverse of a domain category. Obtained from *PointerType.CATEGORY_MEMBER* links.
- Instance — a concrete implementation[6] (applies to proper nouns); Example: “Google” is an instance of “search engine”. Obtained from *PointerType.INSTANCES_HYPERNYM.*
- Has Instance — the inverse of the instance relation. This relation is obtained from *PointerType.INSTANCES_HYPONYM* and with the previous are equivalent to hyponymy/hypernymy applied to proper nouns (in some cases finding instances is equivalent to retrieving the full hyponym tree) hence the pointer terminology.

3.3 A Portuguese WordNet representation

The developed extension was also applied to a Portuguese WordNet represented in an SQL database accepted by JWNL. The contained lexical knowledge was originally extracted[3] from TemaNet[5], Papel[6] and MWN.PT[7] into an SQL database whose schema was extended using a specifically designed SQL script mapping original structures (tables, indexes and views) into knowledge containers required for JWNL interpretation. This database contains word forms grouped in senses/synsets with lexical information such as definition, usage examples, domain and subdomain concatenated by the developed script into a WordNet gloss (eventually) containing more information than defined on the original English version. These senses are linked by semantic pointers describing synonym, hypernym and hyponym relations. Availability of described lexical information slots/types and semantic relations is dependent on the originating lexical resource as such the resulting WordNet does not describe the same information (particularly on glosses) slots for all synsets. Some of the information required by JWNL was also not existent in this database thus being filled with nullable characters/numbers avoiding errors on programmatic access with JWNL.

[5]http://www.instituto-camoes.pt/temanet/inicio.html
[6]http://www.linguateca.pt/PAPEL
[7]http://mwnpt.di.fc.ul.pt/features.html

3.4 Results

The developed getters were manually tested for correspondence with official WordNet interfaces using words in the examples above which apply at least one of the described linguistic features. These words were manually selected from educational linguistic resources about the covered relations having all their senses applied to the developed getters as retrieved by *JWNLSimple.getAllSynsets(expression)*. Some words have senses or features assigned in JWNL (and therefore in WordNet) not shown in official interfaces namely:

- sophisticated — on *JWNLSimple.getAllSynsets(expression)* JWNL's word lookup returns the noun “sophisticate” while official interfaces only show verbs and adjectives although the same stemming rule is applied since both show the verb “sophisticate”.

- still — on *JWNLSimple.getSeeAlso(Synset)* JWNL's *PointerUtils.getAlsoSees(Synset)* has results for the synset with lemmas “hush, quieten, silence, still, shut up, hush up” and glosses “cause to be quiet or not talk, please silence the children in the church” while official interfaces doesn't contain the see also relation on this sense of “still”.

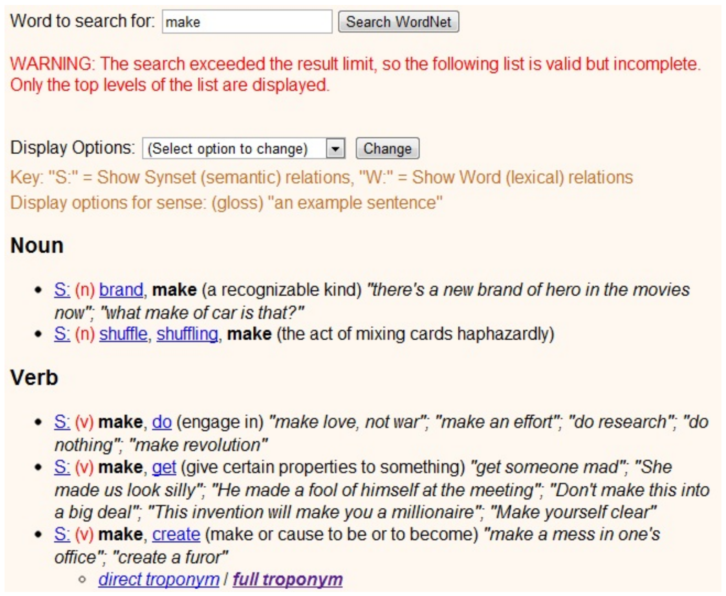
Stress tests were also applied to this extension (and inherently to JWNL) in filesystem, map and database backed WordNet versions. These were repeated in a loop where semantic/linguistic getters are invoked in randomized order and applied to randomized versions of the test set. As such each getter is repeatedly applied to each sense in the test set always surrounded by different procedures in the invocation pipeline thus discarding result fetching from fixed memory locations. The connection reuse problem above described arised during these stress tests since it rarely occurs on paced invocations.

4 Conclusion

Although other frameworks exist for WordNet access JWNL is the only featuring translation of filesystem versions of WordNet to other storage systems. This feature allows usage of WordNet on environments unsuitable for the filesystem version such as distributed access to a central version (for sharing an updatable domain/language specific WordNet) or stateless/diskless systems.

More semantic relations can be retrieved with JWNL although not covered by this extension due to lack of visibility on synsets of the test set (such as troponymy) or appropriate object mapping on JWNL (such as sentence frames) although intended for future work. As in inherited hypernyms some of these can lead to large result sets when applied recursively as shown in the online interface example of figure 2. Unless all the results are required typically large result sets (according to hardware and Java configuration) of inherited/full relations should be replaced with their direct/top level equivalents.

Some of the getters in *PointerUtils* already use WordNet terminology but return JWNL specific data types requiring traversal/usage knowledge from


Word to search for: make Search WordNet

WARNING: The search exceeded the result limit, so the following list is valid but incomplete. Only the top levels of the list are displayed.

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) brand, **make** (a recognizable kind) *"there's a new brand of hero in the movies now"; "what make of car is that?"*
- S: (n) shuffle, shuffling, **make** (the act of mixing cards haphazardly)

Verb

- S: (v) **make**, do (engage in) *"make love, not war"; "make an effort"; "do research"; "do nothing"; "make revolution"*
- S: (v) **make**, get (give certain properties to something) *"get someone mad"; "She made us look silly"; "He made a fool of himself at the meeting"; "Don't make this into a big deal"; "This invention will make you a millionaire"; "Make yourself clear"*
- S: (v) **make**, create (make or cause to be or to become) *"make a mess in one's office"; "create a furor"*
  - *direct troponym* / ***full troponym***

Figure 2: Message displayed when attempting to retrieve full troponyms on a verb sense of "make" illustrating result overflow.

an NLP oriented developer intending to use WordNet. Therefore this extension reduces this knowledge requirements to basic/essential JWNL structures (*Synset* and *Word*) containing only WordNet related information easily accessible/understood with their API's terminology. These WordNet information containers are then grouped in Java's standard datatypes (such as *java.util.LinkedHashSet*) with well known efficiency, organization and usage.

The larger search space in lookup/stemming procedures enables strictly word based usage (as in the online interface) by covering all POS or pointer types but requires the resulting set to be parsed when filtering desired senses. However this filtering can be done according to any of the information types in *Synset* such as gloss or lemma pattern matching.

5 Acknowledgments

This work was supported by Fundação para a Ciência e a Tecnologia (INESC-ID multiannual funding) through the PIDDAC Program funds, and also through the project FALACOMIGO (ProjectoVII em co-promoção, QREN n 13449) that supports Sérgio Curto's fellowship.

References

[1] Luisa Bentivogli, Pamela Forner, Bernardo Magnini, and Emanuele Pianta. Revising the wordnet domains hierarchy: semantics, coverage and balancing. In *Proc. Workshop on Multilingual Linguistic Resources*, MLR '04, pages 101–108, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[2] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. *Text Processing with GATE (Version 6).* GATE, 2011.

[3] Rui Pedro dos Santos Correia. Automatic question generation for reap.pt tutoring system, July 2010.

[4] Armando Stellato Maria Teresa Pazienza and Alexandra Tudorache. A bottom-up comparative study of eurowordnet and wordnet 3.0 lexical and semantic relations. In Bente Maegaard Joseph Mariani Jan Odjik Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

[5] Armando Stellato Maria Teresa Pazienza and Alexandra Tudorache. Jmwnl: an extensible multilingual library for accessing wordnets in different languages. In Bente Maegaard Joseph Mariani Jan Odjik Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

[6] George A. Miller and Florentina Hristea. Wordnet nouns: Classes and instances. *Comput. Linguist.*, 32:1–3, March 2006.

[7] Princeton University. About wordnet. http://wordnet.princeton.edu, 2010.