LREC 2012 Workshop on
Language Resource Merging

Workshop Programme

22 May 2012

2.00pm – 2.15pm – Welcome and Introduction by Núria Bel

2.15pm – 3.00pm – Invited talk
Iryna Gurevych, *How to UBY – a Large-Scale Unified Lexical-Semantic Resource*

3.00pm – 5.30pm – Oral Session

3.00pm – 3.30pm
Laura Rimell, Thierry Poibeau and Anna Korhonen, *Merging Lexicons for Higher Precision Subcategorization Frame Acquisition*

3.30pm – 4.00pm
Muntsa Padró, Núria Bel and Silvia Necşulescu, *Towards the Fully Automatic Merging of Lexical Resources: A Step Forward*

4.00pm – 4.30pm – Coffee break

4.30pm – 5.00pm
Benoît Sagot and Laurence Danlos, *Merging Syntactic Lexica: The Case for French Verbs*

5.00pm – 5.30pm
Cristina Bosco, Simonetta Montemagni and Maria Simi, *Harmonization and Merging of two Italian Dependency Treebanks*

5.30pm – 5.45pm – Short Break

5.45pm – 6.15pm – Poster Session

Riccardo Del Gratta, Francesca Frontini, Monica Monachini, Valeria Quochi, Francesco Rubino, Matteo Abrate and Angelica Lo Duca, *L-Leme: An Automatic Lexical Merger Based on the LMF Standard*

Anelia Belogay, Diman Karagiozov, Cristina Vertan, Svetla Koeva, Adam Przepiórkowski, Maciej Ogrodniczuk, Dan Cristea, Eugen Ignat and Polivios Raxis, *Merging Heterogeneous Resources and Tools in a Digital Library*

Thierry Declerck, Stefania Racioppa and Karlheinz Mörth, *Automatized Merging of Italian Lexical Resources*

Radu Simionescu and Dan Cristea, *Towards an Universal Automatic Corpus Format Interpreter Solution*

Editors

Núria Bel	Universitat Pompeu Fabra, Barcelona, Spain
Maria Gavrilidou	ILSP/Athena R.C., Athens, Greece
Monica Monachini	CNR-ILC, Pisa, Italy
Valeria Quochi	CNR-ILC, Pisa, Italy
Laura Rimell	University of Cambridge, UK

Workshop Organizers/Organizing Committee

Núria Bel	Universitat Pompeu Fabra, Barcelona, Spain
Maria Gavrilidou	ILSP/Athena R.C., Athens, Greece
Monica Monachini	CNR-ILC, Pisa, Italy
Valeria Quochi	CNR-ILC, Pisa, Italy
Laura Rimell	University of Cambridge, UK

Workshop Programme Committee

Victoria Arranz	ELDA, Paris, France
Paul Buitelaaar	National University of Ireland, Galway, Ireland
Nicoletta Calzolari	CNR-ILC, Pisa, Italy
Olivier Hamon	ELDA, Paris, France
Aleš Horák	Masaryk University, Brno, Czech Republic
Nancy Ide	Vassar College, Mass. USA
Bernardo Magnini	FBK, Trento, Italy
Paola Monachesi	Utrecht University, Utrecht, The Netherlands
Jan Odijk	Utrecht University, Utrecht, The Netherlands
Muntsa Padró	UPF-IULA, Barcelona, Spain
Karel Pala	Masaryk University, Brno, Czech Republic
Pavel Pecina	Charles University, Prague, Czech Republic.
Thierry Poibeau	University of Cambridge, UK and CNRS, Paris, France
Benoît Sagot	INRIA, Paris, France
Kiril Simov	Bulgarian Academy of Sciences, Sofia, Bulgaria
Claudia Soria	CNR-ILC, Pisa, Italy
Maurizio Tesconi	CNR-IIT, Pisa
Antonio Toral	DCU, Dublin, Ireland

Table of contents

Author Index

Introduction

The availability of adequate language resources has been a well-known bottleneck for most high-level language technology applications, e.g. Machine Translation, parsing, and Information Extraction, for at least 15 years, and the impact of the bottleneck is becoming all the more apparent with the availability of higher computational power and massive storage, since modern language technologies are capable of using far more resources than the community produces. The present landscape is characterized by the existence of numerous scattered resources, many of which have differing levels of coverage, types of information and granularity. Taken singularly, existing resources do not have sufficient coverage, quality or richness for robust large-scale applications, and yet they contain valuable information (Monachini et al. 2004 and 2006; Soria et al. 2006; Molinero, Sagot and Nicolas 2009; Necşulescu et al. 2011). Differing technology or application requirements, ignorance of the existence of certain resources, and difficulties in accessing and using them, has led to the proliferation of multiple, unconnected resources that, if merged, could constitute a much richer repository of information augmenting either coverage or granularity, or both, and consequently multiplying the number of potential language technology applications. Merging, combining and/or compiling larger resources from existing ones thus appear to be a promising direction to take.

The re-use and merging of existing resources is not altogether unknown. For example, WordNet (Fellbaum, 1998) has been successfully reused in a variety of applications. But this is the exception rather than the rule; in fact, merging, and enhancing existing resources is uncommon, probably because it is by no means a trivial task given the profound differences in formats, formalisms, metadata, and linguistic assumptions.

The language resource landscape is on the brink of a large change, however. With the proliferation of accessible metadata catalogues, and resource repositories (such as the new META-SHARE infrastructure), a potentially large number of existing resources will be more easily located, accessed and downloaded. Also, with the advent of distributed platforms for the automatic production of language resources, such as PANACEA, new language resources and linguistic information capable of being integrated into those resources will be produced more easily and at a lower cost. Thus, it is likely that researchers and application developers will seek out resources already available before developing new, costly ones, and will require methods for merging/combining various resources and adapting them to their specific needs.

Up to the present day, most resource merging has been done manually, with only a small number of attempts reported in the literature towards (semi-)automatic merging of resources (Crouch & King 2005; Pustejovsky et al. 2005; Molinero, Sagot and Nicolas 2009; Necsulescu et al. 2011, Gurevych et al. 2012, Eckle-Kohler and Gurevych 2012). In order to take a further step towards the scenario depicted above, in which resource merging and enhancing is a reliable and accessible first step for researchers and application developers, experience and best practices must be shared and discussed, as this will help the whole community avoid any waste of time and resources.

AIMS OF THE WORKSHOP

This half-day workshop is meant to be part of a series of meetings constituting an ongoing forum for sharing and evaluating the results of different methods and systems for the automatic production of language resources (the first one was the LREC 2010 Workshop on Methods for the Automatic Production of Language Resources and their Evaluation Methods). The main focus of this workshop is on (semi-)automatic means of merging language resources, such as lexicons, corpora and grammars. Merging makes it possible to re-use, adapt, and enhance existing resources, alongside new, automatically created ones, with the goal of reducing the manual intervention required in language resource production, and thus ultimately production costs.

REFERENCES

Dick Crouch and Tracy H. King. 2005. Unifying lexical resources. *Proceedings of Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*. Saarbruecken, Germany.

Judith Eckle-Kohler and Iryna Gurevych. 2012. Subcat-LMF: Fleshing out a standardized format for subcategorization frame interoperability. *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, April 2012.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer and Christian Wirth. 2012. Uby - A Large-Scale Unified Lexical-Semantic Resource. *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (EACL 2012), April 2012.

Monica Monachini, Nicoletta Calzolari, Khalid Choukri, Jochen Friedrich, Giulio Maltese, Michele Mammini, Jan Odijk & Marisa Ulivieri. 2006. Unified Lexicon and Unified Morphosyntactic Specifications for Written and Spoken Italian. In Calzolari et al. (eds.), *Proceedings of the LREC2006: 5th International Conference on Language Resources and Evaluation*, pp. 1852-1857, Genoa, Italy.

Miguel A. Molinero, Benoît Sagot and Nicolas Lionel. 2009. Building a morphological and syntactic lexicon by merging various linguistic resources. In *Proceedings of 17th Nordic Conference on Computational Linguistics (NODALIDA-09)*, Odense, Danemark

Silvia Necsulescu, Núria Bel, Muntsa Padró, Montserrat Marimon, Eva Revilla. 2011. Towards the Automatic Merging of Language Resources. *First international Workshop on Lexical Resources. Woler 2011*. Ljubljana, Slovenia: 1-5 August 2011.

Pustejovsky, J., M. Palmer and A. Meyers. Towards a Comprehensive Annotation of Linguistic Information. *Workshop on Frontiers in Corpus Annotation II. Pie in the Sky*, ACL, Ann Arbor, MI. 2005.

Claudia Soria, Maurizio Tesconi, Nicoletta Calzolari, Andrea Marchetti, Monica Monachini. 2006. Moving to dynamic computational lexicons with LeXFlow. In *Proceedings of the LREC2006: 5th International Conference on Language Resources and Evaluation*, Genoa, Italy (pp. 7–12).

How to UBY - a Large-Scale Unified Lexical-Semantic Resource

Iryna Gurevych

UKP-TUDA, Darmstadt Universität
TU Darmstadt - FB 20 Hochschulstraße 10 64289 Darmstadt
gurevych@ukp.informatik.tu-darmstadt.de

Abstract

The talk will present UBY, a large-scale resource integration project based on the Lexical Markup Framework (LMF, ISO 24613:2008). Currently, nine lexicons in two languages (English and German) have been integrated: WordNet, GermaNet, FrameNet, VerbNet, Wikipedia (DE/EN), Wiktionary (DE/EN), and OmegaWiki. All resources have been mapped to the LMF-based model and imported into an SQL-DB. The UBY-API, a common Java software library, provides access to all data in the database. The nine lexicons are densely interlinked using monolingual and cross-lingual sense alignments. These sense alignments yield enriched sense representations and increased coverage. A sense alignment framework has been developed for automatically aligning any pair of resources mono- or cross-lingually. As an example, the talk will report on the automatic alignment of WordNet and Wiktionary. Further information on UBY and UBY-API is available at: http://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/.

Merging Lexicons for Higher Precision Subcategorization Frame Acquisition

Laura Rimell*, **Thierry Poibeau**†, **Anna Korhonen***

* Dept of Theoretical and Applied Linguistics & Computer Laboratory, University of Cambridge, UK
† LaTTiCe, UMR8094, CNRS & ENS, France
laura.rimell@cl.cam.ac.uk, anna.korhonen@cl.cam.ac.uk, thierry.poibeau@ens.fr

Abstract

We present a new method for increasing the precision of an automatically acquired subcategorization lexicon, by merging two resources produced using different parsers. Although both lexicons on their own have about the same accuracy, using only sentences on which the two parsers agree results in a lexicon with higher precision, without too great loss of recall. This "intersective" resource merger is appropriate when both resources are automatically produced, hence noisy, or when precision is of primary importance, and may also be a useful approach for new domains where sophisticated filtering and smoothing methods are unavailable.

Keywords: verb subcategorization frames, subcategorization lexicon, parser ensemble, language resource merging

1. Introduction

Verb subcategorization frame (SCF) lexicons contain information about the subcategorization preferences of verbs, that is, the tendency of verbs to select the types of syntactic phrases with which they co-occur. For example, the verb *believe* can take a noun phrase complement, a clausal complement, or both together, while the verb *see* can take a noun phrase or a clausal complement, but not both together (Figure 1). SCF lexicons can serve as useful resources for applications requiring information about predicate-argument structure, including parsing (Carroll and Fang, 2004), semantic role labeling (Bharati et al., 2005), verb clustering (Schulte im Walde, 2006), information extraction (Surdeanu et al., 2003), and machine translation (Han et al., 2000).

Manually developed resources containing subcategorization information (Boguraev et al., 1987; Grishman et al., 1994) typically have high precision but suffer from a lack of coverage, making automatic acquisition desirable. The automatic acquisition of SCF information requires extraction of co-occurrence information from large amounts of unstructured text. A typical approach involves using a parser to discover the grammatical relations (GRs, i.e. dependencies) headed by each verb instance, then deciding which GR patterns constitute instances of various SCFs, either by heuristically matching a set of pre-defined patterns, or by accepting all patterns found within the data with a minimum frequency. The resulting set of SCF instances are amalgamated into an SCF lexicon, containing a probability distribution over SCFs for each verb lemma (Briscoe and Carroll, 1997; Korhonen, 2002; Preiss et al., 2007; Messiant et al., 2008; Lapesa and Lenci, 2011). Automatically acquired resources typically have higher coverage than manually developed ones, but suffer from a lack of precision.

A number of filtering and smoothing techniques have been proposed in order to improve the precision of automatically acquired SCF lexicons. Filtering SCFs which are attested below a relative frequency threshold for any given verb, where the threshold is applied uniformly across the whole lexicon, has been shown to be effective (Korhonen, 2002; Messiant et al., 2008). However, this technique relies on empirical tuning of the threshold, necessitating a gold standard in the appropriate textual domain, and it is insensitive to the fact that some SCFs are inherently rare. The most successful methods of increasing accuracy in SCF lexicons rely on language- and domain-specific dictionaries to provide back-off distributions for smoothing (Korhonen, 2002).

SCF	Example
NP	Mary *believed* [$_{NP}$Susan].
CCOMP	Mary *believed* [$_{CCOMP}$that the book had been returned].
NP-CCOMP	Mary *believed* [$_{NP}$Susan] [$_{CCOMP}$that the book had been returned].
NP	Mary *saw* [$_{NP}$Susan].
CCOMP	Mary *saw* [$_{CCOMP}$that the book had been returned].
NP-CCOMP	*Mary *saw* [$_{NP}$Susan] [$_{CCOMP}$that the book had been returned].

Figure 1: Sample subcategorization frames taken by two verbs. The asterisk represents an ungrammatical sentence.

This paper presents a different approach to acquiring a higher precision SCF resource, namely the merging of two automatically acquired resources by retaining only the information that the two resources agree on. Previous work on language resource merging has generally focused on increasing coverage by adding information from one resource to another, e.g. (Crouch and King, 2005; Molinero et al., 2009), which focus on merging multiple levels of information from disparate resources. More closely related to our work, (Necsulescu et al., 2011; Bel et al., 2011; Padró et al., 2011) merge two manually built SCF lexicons, unifying SCFs when possible but with the goal of retaining information from both lexicons. Treating language resource merger as (roughly) a union operation is appropriate for manually developed resources, or when coverage is a priority. However, when working with automatically acquired resources

it may be worthwhile to adopt the approach of merger by intersection.

We focus here on the fact that the tagger and parser are one source of noise in automatic SCF acquisition, and combine two lexicons built with different parsers. This approach is similar in spirit to parser ensembles, which have been used successfully to improve parsing accuracy (Sagae and Lavie, 2006; Sagae and Tsujii, 2007). We build two SCF lexicons using the framework of (Korhonen, 2002; Preiss et al., 2007), which was designed to classify the output of the RASP parser (Briscoe et al., 2006), and which we extend to classify the output of the unlexicalized Stanford parser (Klein and Manning, 2003). We then build a combined lexicon that includes only SCFs that are agreed on by both parsers. Using this simple combination approach, we obtain a lexicon with higher precision than the lexicon built with either parser alone.

2. Previous Work

Manually developed resources containing subcategorization information include ANLT (Boguraev et al., 1987) and COMLEX (Grishman et al., 1994). Automatically aquired SCF resources for English include (Briscoe and Carroll, 1997; Korhonen, 2002; Korhonen et al., 2006a; Preiss et al., 2007), and for other languages such resources as (Messiant et al., 2008) for French, and (Lapesa and Lenci, 2011) for Italian. The state of the art system for SCF acquisition in English is that of (Preiss et al., 2007), which we adopt and extend here. It uses manually defined rules to identify SCFs based on the output of the RASP parser.

The only previous work we are aware of on combining SCF lexicons is (Necsulescu et al., 2011; Bel et al., 2011; Padró et al., 2011). However, they combine manually developed lexicons. To our knowledge there is no previous work on combining automatically acquired SCF lexicons.

Parser ensembles have previously been used to improve parsing accuracy (Sagae and Lavie, 2006; Sagae and Tsujii, 2007), as well as for applications such as extraction of protein-protein interactions (Miyao et al., 2009).

3. System Description

We adapted the SCF acquisition system of (Preiss et al., 2007). First, corpus data is parsed to obtain GRs for each verb instance. We use the RASP parser and the unlexicalized Stanford parser. Second, a rule-based classifier matches the GRs for each verb instance with a corresponding SCF. The classifier of (Preiss et al., 2007) is based on the GR scheme of (Briscoe et al., 2006), used by the RASP parser. Since the Stanford parser produces output in the Stanford Dependencies (SD) scheme (de Marneffe et al., 2006), we developed a new version of the classifier for the Stanford output. We also made some minor modifications to the RASP classifier. At this stage we added a parser combination step, creating a new set of classified verb instances by retaining only instances for which the two classifiers agreed on the SCF. A lexicon builder then extracts relative frequencies from the classified data and builds lexical entries, and the resulting lexicons are filtered.

3.1. Parsing

SCF acquisition requires an unlexicalized parser, i.e. a parser that does not already have a notion of SCF probabilities conditioned on particular verb lemmas, so as not to bias the outcome towards the parser's existing knowledge. RASP is a modular statistical parsing system which includes a tokenizer, tagger, lemmatizer, and a wide-coverage unification-based tag-sequence parser, and has been used in a number of previous SCF acquisition experiments. The Stanford system includes a tokenizer, tagger, lemmatizer, and an unlexicalized[1] stochastic context-free grammar parser. We are unaware of any previous SCF acquisition using the Stanford parser.

3.2. Classifying Verb Instances

The classifier attempts to match the set of GRs produced for each verb instance against its inventory of SCFs, using a set of rules which were manually developed by examining parser output on development sentences. The classifier is implemented in Lisp and examines the graph of GRs headed by the verb, finding the SCF which matches the greatest number of GRs. For example, if the verb has a direct object (NP) and an indirect object (PP), then the classifier will find SCF NP-PP, not NP. (Note that we do not include subjects in the SCF name, since they are obligatory in English.)

For the RASP parser, we used a re-implementation in Lisp of the rule set in (Preiss et al., 2007). We made some minor modifications to the rules based on examination of development data.

Despite commonalities between the GR scheme of (Briscoe et al., 2006) and the SD scheme, the realization of a particular SCF can nevertheless exhibit a number of differences across schemes. Rather than converting the SD output to (Briscoe et al., 2006) format, a complex many-to-many mapping that would likely lose information, we chose to develop a new version of the classifier, based on examination of development data parsed by the Stanford parser. Figure 2 shows an example of parser output in the two schemes.

3.3. Merging Classifier Output

For the combined lexicon, we merged the classifier output on a sentence-by-sentence basis. A sentence was considered to exemplify an SCF for a verb only if both classifiers, RASP and Stanford, agreed on that SCF based on the parser output. Note that we did not merge the results of the lexicon building step (Section 3.4.), which would mean accepting an SCF on a verb-by-verb basis, if the two lexicons agreed that the verb takes that SCF. We chose not to use this strategy since we believed it would allow more errors of both parsers to pass through the pipeline.[2]

[1] Though the Stanford parser is unlexicalized, the rules provided with the parser to generate GRs from a constituent parse are mildly lexicalized; for example, they can distinguish some raising verbs. This affects only a small number of SCFs. We made use of the information when it was available.

[2] We also did not combine parsers by voting on individual GRs, to generate a new *parse* with higher accuracy than the individual parser output; this would have been difficult due to the differences between the GR schemes.

SCF: EXTRAP-TO-NP-S
It matters to them that she left.
RASP
`ncsubj(matter it _)`
`iobj(matter to)`
`ccomp(that matter leave)`
`ncsubj(leave she _)`
Stanford
`nsubj(matter it)`
`prep(matter to)`
`ccomp(matter leave)`
`nsubj(leave she)`

Figure 2: Example sentence with RASP and SD GRs (incidental formatting has been normalized). The classifier rules identify this SCF only when the word *it* is in subject position and the preposition is *to*.

In some cases, differences in the two GR schemes allowed the parsers to take different views on the data. For example, RASP cannot distinguish the SCFs ADVP (*He meant well*) and PARTICLE (*She gave up*), since it analyzes both *well* and *up* as particle-type non-clausal modifiers. However, Stanford distinguishes the two as adverbial modifier and particle, respectively. In such cases we used the more fine-grained analysis in the resulting lexicon.

3.4. Lexicon Building and Filtering

The lexicon builder amalgamates the SCFs hypothesized by the classifier for each verb lemma. SCFs left underspecified by the classifier are also treated here. As the gold standard SCF inventory is very fine-grained, there are a number of distinctions which cannot be made based on parser output. For example, the gold standard distinguishes between transitive frame NP with a direct object interpretation (*She saw a fool*) and NP-PRED-RS with a raising interpretation (*She seemed a fool*), but parsers in general are unable to make this distinction. We used two different strategies at lexicon building time: weighting the underspecified SCFs by their frequency in general language, or choosing the single SCF which is most frequent in general language. For example, we either assign most of the weight to SCF NP with a small amount to NP-PRED-RS, or we assign all the weight to NP. The goal of the parser combination method is to increase the precision of the acquired lexicon, which is also the goal of the various filtering methods for removing noise from SCF lexicons. In order to investigate the role of filtering in the context of parser combination, we filtered all the acquired lexicons using uniform relative frequency thresholds of 0.01 and 0.02.

4. Experiments

4.1. Gold Standard

We used the gold standard of (Korhonen et al., 2006b), consisting of SCFs and relative frequencies for 183 general-language verbs, based on approximately 250 manually annotated sentences per verb. The verbs were selected randomly, subject to the restriction that they take multiple SCFs. The gold standard includes 116 SCFs. Because of the Zipfian nature of SCF distributions – a few SCFs are taken by most verbs, while a large number are taken by few verbs – only 36 of these SCFs are taken by more than ten verbs in the gold standard.

Filtering Method		RASP	Stanford	Comb.
Unfiltered	P	9.6	10.0	15.7
	R	95.8	95.4	90.3
	F	17.5	18.2	26.7
Uniform 0.01	P	42.7	38.6	50.8
	R	59.0	59.8	56.7
	F	49.6	46.9	53.6
Uniform 0.02	P	52.6	43.9	56.7
	R	48.8	47.2	46.6
	F	50.6	45.5	51.1

Table 1: Type precision, recall, and F-measure for 183 verbs. Underspecified SCFs weighted by frequency in general language.

4.2. Corpus Data

The input corpus consisted of up to 10,000 sentences for each of the 183 verbs, from the British National Corpus (BNC) (Leech, 1993), the North American News Text Corpus (NANT) (Graff, 1995), the Guardian corpus, the Reuters corpus (Rose et al., 2002), and TREC-4 and TREC-5 data. Data was taken preferentially from the BNC, using the other corpora when the BNC had insufficient examples.

4.3. Evaluation Measures

We used type precision, recall, and F-measure for lexicon evaluation, as well as the number of SCFs present in the gold standard but missing from the unfiltered lexicon (i.e. not acquired, rather than filtered out). We also measured the distributional similarity between the acquired lexicons and the gold standard using various measures.

5. Results and Discussion

Tables 1 and 2 show the overall results for each parser alone as well as the combination, using the two different methods of resolving underspecified SCFs. We note first that the single-parser systems show similar accuracy across the different filtering thresholds. In Table 1, both systems achieve an F-score of about 18 for the unfiltered lexicon, and between 45 and 50 for the uniform frequency thresholds of 0.01 and 0.02. In Table 2, the accuracy is slightly higher overall, with both systems achieveing F-scores of about 21-22 for the unfiltered lexicon, and between 51-57 for the uniform frequency thresholds. The RASP-based system achieves higher accuracy than the Stanford-based system across the board, due to higher precision. We attribute this difference to the fact that the RASP classifier rules have been through several generations of development, while the Stanford rule set was first developed for this paper and has had the benefit of less fine-tuning, rather than to any difference in suitability of the two parsers for the task.

The merged lexicon shows a notable increase in precision at each filtering threshold compared to the single-parser lexicons, with, in most cases, a corresponding increase in F-score. In Table 1, the unfiltered lexicon achieves an F-score

Filtering Method		RASP	Stanford	Comb.
Unfiltered	P	12.1	12.9	22.8
	R	83.6	86.8	82.4
	F	21.2	22.5	35.7
Uniform 0.01	P	48.6	42.8	59.9
	R	62.5	62.7	58.9
	F	54.7	50.9	59.4
Uniform 0.02	P	61.5	51.4	68.3
	R	52.8	51.3	48.6
	F	56.8	51.3	56.8

Table 2: Type precision, recall, and F-measure for 183 verbs. Underspecified SCFs by taking the single most frequent SCF from the set.

of 26.7, the lexicon with a uniform frequency threshold of 0.01 an F-score of 53.6, and with a uniform frequency threshold of 0.02 an F-score of 51.1. In Table 2, the unfiltered lexicon achieves an F-score of 35.7, the lexicon with a uniform frequency threshold of 0.01 an F-score of 59.4, and with a uniform frequency threshold of 0.02 an F-score of 56.8. Depending on the settings, the increase in precision over the higher of the single-parser lexicons ranges from about four points (Table 1, bottom row) to over 11 points (Table 2, middle row). This increase is achieved without developing any new classifier rules.

An interesting effect of merging can be observed in the unfiltered case. The unfiltered lexicons all have an extreme bias towards recall over precision. Because of noise in the parser and classifier output, most SCFs are hypothesized for each verb. However, the merged lexicon shows higher precision even in the unfiltered case: effectively, the merger acts as a kind of filter.

The combined lexicon does show somewhat lower recall than the single-parser lexicons. This is probably due to the fact that the intersection of the two classifier outputs resulted in a much smaller number of sentences in the input to the lexicon builder. Recall that the original dataset contained up to 10,000 sentences per verb. Not all of these sentences were classified in each pipeline, either due to parser errors or to the GRs failing to match the rules for any SCF. On average, the RASP classifier classified 6,500 sentences per verb, the Stanford classifier 5,594, and the combined classifier only 1,922. It should be noted that classifying more sentences does not necessarily mean better accuracy, since the classifications are noisy; in some cases it is preferential not to match on any SCF. In fact, the Stanford-based lexicon was based on fewer sentences than the RASP-based lexicon without loss of recall. However, the input corpus for the combined lexicon was effectively much smaller than the input corpus for the other two lexicons, which probably contributed to the loss of recall.

We found that the best results for the individual parsers were obtained with the higher threshold (0.02), and for the combination with the lower threshold (0.01). Again, this is probably due to the smaller effective number of sentences classified; rare SCFs were more likely to fall below the threshold. As the threshold value increases, the precision and F-score for the single-parser lexicons approach that of the combined lexicon, because increasing the threshold always has the effect of increasing precision at the expense of recall. Using a parser combination achieves the same effect without the need to tune the threshold.

	RASP	Stanford	Comb.
Number missing	0	1	7

Table 3: Missing SCFs in unfiltered lexicon.

The next measure we look at is the number of SCFs that were present in the gold standard but missing from the unfiltered lexicons, i.e. never identified at all by the SCF aquisition system (rather than filtered out). For this measure we use the weighting method of treating underspecified SCFs (as in Table 1); otherwise the assignment of probability mass to the most frequent SCF in the underspecified cases means that many more SCFs are missed. The results are shown in Table 3. The merged lexicon clearly suffers on this measure, as there were seven SCFs that it did not identify at all; however, these SCFs are all rare, so they are presumably not the most important ones for downstream applications. For example, the merged lexicon does not identify the frame PP-WHAT-TO-INF, e.g. *They deduced from Kim what to do*, or TO-INF-SUBJ, e.g. *To see them hurts*, both of which are rare in general language according to the ANLT dictionary. Sometimes SCFs were missed because each parser/classifier identified the SCF, but never both on the same sentence, and in other cases neither individual parser/classifier identified a true positive.

The one missing SCF for the unfiltered Stanford lexicon was POSS-ING, e.g. *She dismissed their writing novels*. The Stanford tagger consistently tags the gerund as NN rather than VVG, which makes the SCF impossible to identify.

On the other hand, the merged lexicon shows a clear increase in the number of SCFs it can identify accurately. Table 4 shows the SCFs identified with at least 50% accuracy (F-score) in the unfiltered lexicon; the combined system was able to do this for 17 SCFs, compared to 8 and 7 for the RASP- and Stanford-based systems, respectively. This includes the very important PP frame, e.g. *They apologized to him*, which is very frequent in general language and relies for its identification on accurate argument-adjunct discrimination. Several frames with *wh*-elements were also identified with greater than 50% accuracy in the combined lexicon but not the single-parser lexicons, such as WH-TO-INF, e.g. *He asked whether to clean the house*.

We next compare the acquired lexicons to the gold standard using various measures of distributional similarity: Kullback-Leibler divergence (KL), Jensen-Shannon divergence (JS), cross entropy (CE), skew divergence (SD), and rank correlation (RC). These measures all compare the SCF probability distributions learned by the SCF acquisition system for each verb lemma. Such measures are a useful complement to the type precision, recall, and F-score evaluation, because unlike the type-based measures, the distributional similarity measures compare the *frequencies* learned by the SCF acquisition system. We use several measures since they exhibit different sensitivity to noise in the data; see (Korhonen and Krymolowski, 2002) for a discussion of

	RASP	Stanford	Comb.
INTRANSITIVE	•	•	•
TRANSITIVE	•	•	•
NP-PP	•	•	•
PARTICLE	•	•	•
PARTICLE-NP	•	•	•
PARTICLE-NP-PP	•	•	•
PARTICLE-PP	•		•
WH-TO-INF	•	•	•
ADVP			•
EXTRAP-TO-NP-S			•
HOW-S			•
HOW-TO-INF			•
PP			•
PP-HOW-TO-INF			•
WH-S			•
WHAT-S			•
FIN-CLAUSE-SUBJ			•

Table 4: SCFs identified with F-score of at least 50 in unfiltered lexicon.

Measure	RASP	Stanf	Comb.
KL distance	0.376	0.376	0.337
JS divergence	0.072	0.083	0.059
cross entropy	1.683	1.680	1.619
skew divergence	0.345	0.358	0.297
rank correlation	0.627	0.599	0.666

Table 5: Distributional similarity measures comparing unfiltered lexicons to the gold standard, on SCFs common to both gold and acquired lexicon. Lower value means greater correlation: KL, JS, CE, SD. Higher value means greater correlation: RC.

the application of the various distributional similarity measures to SCF acquisition.

Table 5 shows the results of the distributional similarity comparisons on the unfiltered acquired lexicons. In each case the merged lexicon shows greater similarity to the gold standard than either of the single-parser lexicons.

Finally, an indication of how the parser combination acts as a kind of filter is given in Table 6, which shows the number of SCFs proposed for each verb lemma. The single-parser classifiers posit a higher number of SCFs: some genuine higher frequency SCFs, followed by a long noisy tail of false positives. The parser combination proposes only half the number of SCFs per verb lemma in the unfiltered lexicon.

6. Conclusion

We have combined the SCF classifier output for two parsers to produce a higher precision verb subcategorization lexicon than those resulting from the single-parser classifiers. This higher precision is achieved without the need for dictionaries or other external resources. Although there is a significant initial investment in defining the parser-specific SCF classifier rules for a particular unlexicalized parser to form part of the merged system, the resulting SCF acquisition system can subsequently be used across a variety of domains without additional effort. The improved precision is particularly interesting in the case of the unfiltered SCF lexicons, since the merger effectively acts as a kind of filter on incorrect SCFs. The unfiltered, merged lexicon is not accurate enough for downstream applications, but the filtered, merged lexicon also exhibits higher precision than the filtered single-parser lexicons. The interaction between parser combination and various filtering methods should be further investigated.

Future work should attempt to overcome the fact that the number of sentences successfully classified decreased dramatically with the parser combination, resulting in loss of recall. Using a larger input corpus would be a natural first step. Another natural extension which we leave for future work is to use a more nuanced version of the "intersective" merger; for example, increasing the likelihood of an SCF when the parsers/classifiers agree, but still retaining the sentences where they do not agree. It may also be possible to identify and leverage the particular strengths of each parser to aid in SCF identification.

	RASP	Stanford	Comb.
SCFs proposed	94.5	90.2	54.7

Table 6: Average number of SCFs proposed per verb in the unfiltered lexicons. Average over 183 verbs in gold standard.

Acknowledgements

This work was funded by the EU FP7 project 'PANACEA' and the Royal Society (UK). Thierry Poibeau is supported by the laboratoire d'excellence (labex) Empirical Foundation of Linguistics.

7. References

Núria Bel, Muntsa Padró, and Silvia Necsulescu. 2011. A method towards the fully automatic merging of lexical resources. In *Proceedings of the Workshop on Language Resources, Technology and Services in the Sharing Paradigm, IJCNLP-11*, Chiang Mai, Thailand.

Akshar Bharati, Sriram Venkatapathy, and Prashanth Reddy. 2005. Inferring semantic roles using subcategorization frames and maximum entropy model. In *Proceedings of CoNLL*, pages 165–168, Ann Arbor.

B. Boguraev, J. Carroll, E.J. Briscoe, D. Carter, and C. Grover. 1987. The derivation of a grammatically-indexed lexicon from the Longman Dictionary of Contemporary English. In *Proceedings of the 25th Annual Meeting of ACL*, pages 193–200, Stanford, CA.

E.J. Briscoe and J. Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Proceedings of the 5th ACL Conference on Applied Natural Language Processing*, pages 356–363, Washington, DC.

E.J. Briscoe, J. Carroll, and R. Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*.

J. Carroll and A. Fang. 2004. The automatic acquisition of verb subcategorisations and their impact on the performance of an HPSG parser. In *Proceedings of the 1st In-*

ternational Joint Conference on Natural Language Processing (IJCNLP), pages 107–114, Sanya City, China.

D. Crouch and T.H. King. 2005. Unifying lexical resources. In *Proceedings of the Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*, Saarbruecken, Germany.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

D. Graff, 1995. *North American News Text Corpus*. Linguistic Data Consortium.

R. Grishman, C. Macleod, and A. Meyers. 1994. COMLEX syntax: Building a computational lexicon. In *Proceedings of COLING*, Kyoto.

C. Han, Benoit Lavoie, Martha Palmer, Owen Rambow, Richard Kittredge, Tanya Korelsky, and Myunghee Kim. 2000. Handling structural divergences and recovering dropped arguments in a Korean/English machine translation system. In *In Proceedings of the AMTA*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430.

A. Korhonen and Y. Krymolowski. 2002. On the robustness of entropy-based similarity measures in evaluation of subcategorization acquisition systems. In *Proceedings of the Sixth CoNLL*, pages 91–97, Taipei, Taiwan.

Anna Korhonen, Yuval Krymolowski, and Ted Briscoe. 2006a. A large subcategorization lexicon for natural language processing applications. In *In Proceedings of LREC*.

Anna Korhonen, Yuval Krymolowski, and Ted Briscoe. 2006b. A large subcategorization lexicon for natural language processing applications. In *Proceedings of LREC*.

Anna Korhonen. 2002. *Subcategorization Acquisition*. Ph.D. thesis, University of Cambridge.

Gabriella Lapesa and Alessandro Lenci. 2011. Modeling subcategorization through co-ocurrence. Presented at Explorations in Syntactic Government and Subcategorization, Cambridge, UK, September 2011.

G. Leech. 1993. 100 million words of English. *English Today*, 9(1):9–15.

Cédric Messiant, Anna Korhonen, and Thierry Poibeau. 2008. LexSchem: A large subcategorization lexicon for French verbs. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Marrakech.

Yusuke Miyao, Kenji Sagae, Rune Saetre, Takuya Matsuzaki, and Jun'ichi Tsujii. 2009. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25:394–400.

Miguel A. Molinero, Benoît Sagot, and Lionel Nicolas. 2009. A morphological and syntactic wide-coverage lexicon for Spanish: The Leffe. In *Proceedings of RANLP*, Borovets, Bulgaria.

Silvia Necsulescu, Núria Bel, Muntsa Padró, Montserrat Marimon, and Eva Revilla. 2011. Towards the automatic merging of language resources. In *Proceedings of the International Workshop on Lexical Resources (WoLeR)*, Ljubljana, Slovenia.

Muntsa Padró, Núria Bel, and Silvia Necsulescu. 2011. Towards the automatic merging of lexical resources: Automatic mapping. In *Proceedings of RANLP*, Hissar, Bulgaria.

Judita Preiss, Ted Briscoe, and Anna Korhonen. 2007. A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proceedings of ACL*.

T.G. Rose, M. Stevenson, and M. Whitehead. 2002. The Reuters Corpus volume 1 – from yesterday's news to tomorrow's language resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 29–31.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of NAACL HLT: Short Papers*, pages 129–132.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task*, pages 1044–1050.

Sabine Schulte im Walde. 2006. Experiments on the automatic induction of German semantic verb classes. *Computational Linguistics*, 32(2):159–194.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL*, Sapporo.

Towards the Fully Automatic Merging of Lexical Resources: a Step Forward

Muntsa Padró, Núria Bel and Silvia Necşulescu
Universitat Pompeu Fabra
Roc Boronat, 138, ES-08018-Barcelona
E-mail: muntsa.padro@upf.edu, nuria.bel@upf.edu, silvia.necsulescu@upf.edu

Abstract

This article reports on the results of the research done towards the fully automatically merging of lexical resources. Our main goal is to show the generality of the proposed approach, which have been previously applied to merge Spanish Subcategorization Frames lexica. In this work we extend and apply the same technique to perform the merging of morphosyntactic lexica encoded in LMF. The experiments showed that the technique is general enough to obtain good results in these two different tasks which is an important step towards performing the merging of lexical resources fully automatically.

Keywords: automatic merging of lexical resources, lmf, feature structures, graph unification

1. Introduction

The automatic production, updating, tuning and maintenance of Language Resources for Natural Language Processing is currently being considered as one of the most promising areas of advancement for the full deployment of Language Technologies. The reason is that these resources that describe, in one way or another, the characteristics of a particular language are necessary for Language Technologies to work for that particular language.

Although the re-use of existing resources such as WordNet (Fellbaum, 1998) in different applications has been a well known and successful case, it is not very frequent. The different technology or application requirements, or even the ignorance about the existence of other resources, has provoked the proliferation of different, unrelated resources that, if merged, could constitute a richer repository of information augmenting the number of potential uses. This is especially important for under-resourced languages, which normally suffer from the lack of broad coverage resources.

Several attempts of resource merging have been addressed and reported in the literature. Hughes et al. (1995) report on merging corpora with more than one annotation scheme. Ide and Bunt (2010) also report on the use of a common layer based on a graph representation for the merging of different annotated corpora. Teufel (1995) and Chan and Wu (1999) were concerned with the merging of several source lexica for part-of-speech tagging. The merging of more complex lexica has been addressed by Crouch and King (2005) who produced a Unified Lexicon with lexical entries for verbs based on their syntactic subcategorization in combination with their meaning, as described by WordNet (Fellbaum, 1998), Cyc (Lenat, 1995) and VerbNet (Kipper et al., 2000).

Despite the undeniable achievements of the research just mentioned, most of it reports the need for a significant amount of human intervention to extract the information of existing resources and to map it into a format in which both lexica can be compared. The cost of this manual effort might explain the lack of more merging attempts. Therefore, any cost reduction would have a high impact in the actual re-use of resources.

In this context, a proposal such as the Lexical Markup Framework, LMF (Francopoulo et al. 2008) is also an attempt to standardize the format of computational lexica as a way to reduce the complexities of merging lexica. However, LMF (ISO-24613:2008) “does not specify the structures, data constraints, and vocabularies to be used in the design of specific electronic lexical resources”. Therefore, the merging of two LMF lexica is certainly easier, but only if both also share the structure and vocabularies, if not, mapping has still to be done by hand. Our aim is to work towards the full automatization of the whole merging process. This constituted the main challenge of the research reported in Bel et al. (2011), where a method to perform the merging of two different lexical resources fully automatically was proposed. They applied the proposed method to the particular case of merging two very different subcategorization frame (SCF) lexica for Spanish obtaining encouraging results.

The aim of the research we present here was to assess to what extent the actual merging of information contained in different LMF lexica can be done automatically, following the mentioned method, in two cases: when the lexica to be merged share structure and labels, and when they do not. Besides, our second goal was to prove the generality of the approach, i.e. if it could be applied to different types of lexical resources.

Therefore, for this work we applied the method presented in Bel et al. (2011) to merge different Spanish morphosyntactic dictionaries. A first experiment tackled the merging of a number of dictionaries of the same

family: Apertium monolingual lexica developed independently for different bilingual MT modules. A second experiment merged the results of the first experiments with the Spanish morphosyntactic FreeLing lexicon. All the lexica were already in the LMF format, although Apertium and FreeLing have different structure and tagset. In addition, note that these morphosyntactic lexica contain very different information than SCF lexica of the first experiments, and that what we present here can be considered a further proof of the good performance and generality of the proposed automatic merging method.

The current results have shown that the availability of the lexica to be merged in a common format such as LMF indeed alleviates the problem of merging. In our experiment with different Apertium lexica it was possible to merge three different monolingual morphosyntactic lexica with the method proposed as to achieve a larger resource. We have also obtained good results in the merging of different tag set based lexica.

2. Methodology

Basically, the merging of lexica has two well defined steps (Crouch and King, 2005):

1. Mapping step: because information about the same phenomenon can be expressed differently, the information in the existing resources has to be extracted and mapped into a common format.
2. Combination Step: once the information in both lexica is encoded in the same way, this information from both lexica is mechanically compared and combined to create the new resource.

Thus, our goal is to carry out the two steps of the merging process in a fully automatic way. This is to perform both mapping and combination steps without any human supervision.

In this section, we will first describe the lexica we wanted to merge, after we will discuss the problems of the combination step, which is simpler and motivates the mapping, which we will explain later.

2.1. The lexica

We have worked with Apertium lexica. Apertium (Armentano-Oller et al., 2007) is an open source rule-based MT system. In this framework, bilingual MT systems are developed independently (and by different people), and this also holds for the lexica for the same language that belong to different bilingual systems. These lexica that share format and tags can differ in the number of entries and the particular encoding of particular entries. For our experiments we merged three Spanish monolingual lexica coming from the Catalan-Spanish with 39,072 entries, English-Spanish with 30,490 entries and French-Spanish with 21,408 entries. In table 1 we further describe details of these lexica. Thus, we found numerous cases of common entries, missing entries in some of them, and also some phenomena related to homography (i.e. the same lemma with different morphological paradigm) as it is the case of *contador* that in one lexicon appears as the machine (‘meter’), only

```
Apertium source:
    tenebrosísimo:tenebroso<adj><sup><m><sg>
Apertium LMF:
    <LexicalEntry  id="id20588-s">
    <feat att="partOfSpeech" val="adj"/>
    <Lemma>
        <feat att="writtenForm" val="tenebroso"/>
    </Lemma>
    <WordForm>
        <feat att="writtenForm" val="tenebrosísimo"/>
        <feat att="type" val="sup"/>
        <feat att="gender" val="m"/>
        <feat att="number" val="sg"/>
    </WordForm>

FreeLing source:
    tenebroso tenebroso AQ0MS0
FreeLing LMF:
    <LexicalEntry>
    <feat att="partOfSpeech" val="adjectiveQualifier"/>
    <Lemma>
        <feat att="writtenForm" val="tenebroso"/>
    </Lemma>
    <WordForm>
        <feat att="writtenForm" val="tenebroso"/>
        <feat att="grade" val="-"/>
        <feat att="grammaticalGender"
        val="masculine"/>
        <feat att="grammaticalNumber"
        val="singular"/>
        <feat att="function" val="-"/>
    </WordForm>
```

Figure 1. Source and LMF versions for the adjective “tenebroso” (‘gloomy’) in Apertium and FreeLing lexica

masculine forms, and in other as the person with both feminine and masculine forms.

FreeLing morphosyntactic lexicon is used for morphological analysis and PoS disambiguation modules of the FreeLing NLP suite (Padró et al., 2010). It uses an adapted version of the EAGLES tag set (Leech and Wilson, 1999). The lexica, originally in the format shown in 1 as source, were converted into LMF in the framework of the METANET4U[1] project (the converted lexica are available in META-SHARE repository) but without changing the tag set labels. Although semantically very close (they both are describing morphosytactic data), main differences between the Apertium and FreeLing tag sets are in the way the information is encoded. For instance adjectives in FreeLing encode ‘grade’ and

[1] An EU PSP-CIP funded project whose aim is to make available and accessible updated and standardized language resources. The META-SHARE repository will be used for the distribution of such resources. www.meta-share.eu

'function', while in Apertium the grade was converted into a "type". The spelling of the name of attributes and values also vary in the source and in the converted files. Note that the conversion into LMF was done independently for each lexicon and followed the information supplied by the available documentation where the semantics of the tags was explained. The order of the features is maintained as well as the number of features.

2.2. Combination of lexica using graph unification

Necsulescu et al. (2011) and Bel et al. (2011) proposed to perform the combination step using graph unification (Kay, 1979). This single operation which is based on set union of compatible feature values, makes it possible to validate the common information, exclude the inconsistent one and to add, if desired, the unique information that each lexicon contained for building a richer resource. For graph unification in our experiments, we used the NLTK unification mechanism (Bird, 2006).

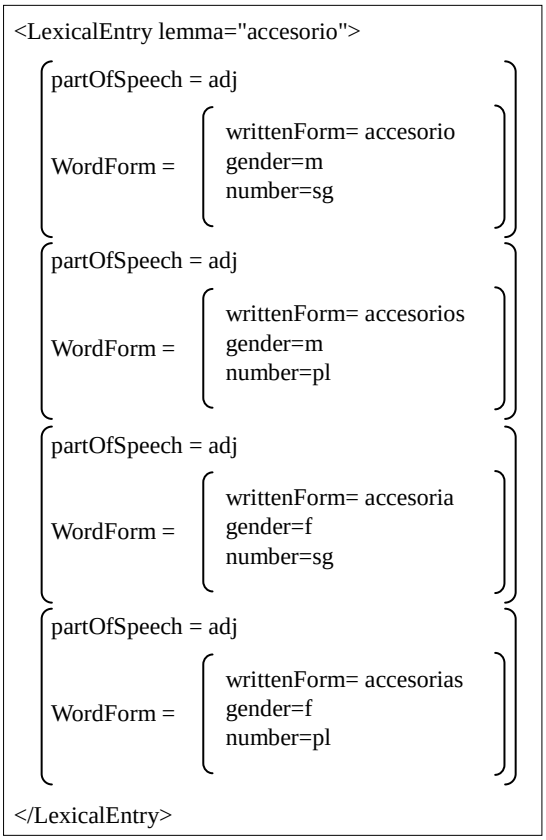

Figure 2. An Apertium LMF entry represented as a feature structure for graph unification.

In order to use graph unification, the LMF lexica had to be represented as feature structures (graphs, see figure 2 for a sample). Because LMF lexica already identified attributes and values in a structured way, this step was straightforward. Note that in converting LMF into feature structures, a lexical entry contains all its <WordForm> as present in the original lexicon together with the part of speech., while the lemma information is encoded as a special feature outside the feature structure in order to guide the unification process. This process is carried out by the system along the following steps:

1) For each lemma that is common to both lexica, it gathers all lexical entries with that lemma in both lexica (cases of homography are taken into account).

2) For the set of entries got in (1), it tries to unify every entry in one lexicon with all the entries in the other lexicon. This step implies checking unification for all feature structures included in the entries.

3) When having a successful unification, create an entry in the resulting lexicon. Unification operation will deliver as feature structures in the resulting entry those that resulting from the common information and also those present in one entry but not in the other.

4) When a lexical entry does not unify with anyone of the other lexicon, it creates an entry in the resulting lexicon as well, because it is considered to contain unique information.

5) For those lemmas that only are in one of the lexica, it creates a lexical entry in the resulting lexicon.

In order to be able to inspect the results, information about the operation that originated the entries in the resulting lexicon is registered in a log file.

2.3. Semantic Preserving Mapping

The proposal to avoid manual intervention when converting two lexica into a common format with a blind, semantic preserving method (Bel et al., 2011) departs from the idea of Chan and Wu (1999) of comparing information contained in common entries of different lexica and looking for significant equivalences in terms of consistent repetition. The basic requirement for this automatic mapping is to have a number of common entries encoded in the two lexica to be compared. Chan and Wu (1999) were working only with single part-of-speech tags, but the lexica we address here handle more complex and structured information, which has to be identified as units by the algorithm. In order to avoid the necessity of defining the significant pieces of information to be mapped by hand, Bel et al. (2011) proposed a method to first automatically identify such pieces ("minimal units") in each lexicon and secondly, to automatically learn the correspondence of such pieces between the two lexica. Their results showed that it is possible to assess that a piece of the code in lexicon A corresponds to a piece of code in lexicon B since a significant number of different lexical entries hold the same correspondence. Then, when a correspondence is found, the relevant piece in A is substituted by the piece in B, performing the conversion into the target format to allow for comparison and, eventually, merging as explained in section 2.2. Note that the task is defined in terms of automatically learning correspondences among both, labels and structure since both may differ across lexica. For example, in FreeLing the verb tense and mood are encoded in two different attributes (e.g. mood=subjunctive, tense=present), while Apertium encodes both tense and mood in a sole attribute (e.g.

tense=prs).

Lexicon	Lexical Entries	Av. Word Forms per entry	Lexical Entries per PoS				
			Nouns	Verbs	Adjectives	Adverbs	Proper nouns
Apertium							
Apertium ca-es	39,072	7.35	16,054	4,074	5,883	4,369	8,293
Apertium en-es	30,490	6.41	11,296	2,702	4,135	1,675	10,084
Apertium fr-es	21,408	6.78	7,575	2,122	2,283	729	8,274
Aperium unified (all)	60,444	6.14	19,824	5,127	7,312	5,340	21,917
FreeLing							
FreeLing	76,318	8.76	49,519	7,658	18,473	169	0
Apertium and Freeling							
Apertium and FreeLing unified (mapping to FreeLing)	112,621	7.03	54,830	8,970	20,162	5,406	21,917

Table 1: Original and unified lexicon sizes

The algorithm used in this work to learn the mapping between two lexica is basically the same used by Bel et al. (2011) although two changes were introduced in order to gain in generality. The main difference is due to the fact that in the first experiments with SCF lexica no attribute had an open list of values (for instance, the value of the attribute for 'writtenForm' does not have a closed number of possible values). We have made the algorithm more general, able to deal with a larger number of possible resource types by adding a different treatment for open and closed feature value types. The identification and special treatment of open values is made fully automatically and affect the step of finding units and learning the correspondence between lexica.

The identification of the open values is now the algorithm first step. By counting the different values in the lexicon, the system decides a feature value to be open when a relative large number of values are encountered. Open values are substituted with a variable in order to find the repetitions that are learnt as a pattern.

The other difference is that, because of the LMF source, we can work from the beginning with a feature structure version of the lexica, while in Bel et al. (2011) they worked with the source formats. Therefore, our algorithm splits each feature structure into feature-value pairs and looks for the elements that always occur together in order to identify the "minimal units" of information. This step is necessary in order to gain in generality when learning correspondences. Note that the probability of finding significant correspondences of larger units is lower. For instance, the system must learn that in FreeLing, *tense* and *mood* features always occur together, and that they both correspond to the information that is a value of the feature *tense* in Apertium.

In order to learn such mapping, for each possible pair of minimal units that are a potential mapping rule, the system measures the similarity in terms of the lemmas that contain a member of the pair in the corresponding lexica. That is, the list of lemmas that contain each minimal unit is represented as a binary vector and the Jaccard distance measure is used to compute similarity between vectors[2] (as Chan and Wu, 1999). The system chooses as correspondences those that maximize similarity, i.e., those with a larger number of lemmas that contain the minimal units to be mapped. In case that there is more than one correspondence, all are considered possible mappings.

Once the corresponding units have been identified, a new feature structure is created substituting units in lexicon A with the corresponding units of lexicon B. This operation results in a lexicon A encoded with the tagset of lexicon B. Now, both lexica can be compared and merged as explained in section 2.2. Note that the mapping should preserve the semantic of the feature value pair. Furthermore, this procedure also identifies the differences between the two lexica when no mapping for a particular minimal unit is found. This information can be later used for creating patches that systematically carry out corrective actions: direct transformation, deletion of feature structures, etc.

3. Experiments and Results

Our experiments were the following. We first merged the three Apertium lexica, and we evaluated the success of the combination step. For these three lexica, no mapping was required because they all use the same tagset. Once this merged lexicon was created, it was mapped and merged with the FreeLing lexicon. The results of the merging are

[2] If the minimal unit is a sibling of an open valued feature, the elements in the vector are the values of this feature instead of the lemmas. E.g. "gender=m", is a sibling of "writtenForm", so the vector used will contain the values of "writtenForm".

presented in table 1.
From the results of merging all Apertium lexica, it is noticeable that the resulting Apertium lexicon has two times the entries (in average) of the source lexica, and that the part of speech that supplied more entries was proper noun. One can explain this if takes into account the independent development of the lexica and that each one probably took different reference test corpora. For the other parts of speech, there is a general increase of number of entries.

As for the merging with FreeLing lexicon experiment, in order to validate the results, both conversion senses were tested giving similar results. We will only comment on the Apertium into FreeLing as we have only closely inspected that experiment. From the data in table 1, we can see that again proper nouns but also adverbs are the main source of new entries. Because FreeLing did not include proper nouns, all the Apertium ones are added. Adverbs are also a major source of new elements, which can be explained because FreeLing handles derivate adverbs (adjective with the *–mente* suffix) differently to Apertium.

In what follows, we present separatedly the results of the two different steps, mapping and merging, for the Apertium into FreeLing lexica experiment. Also, concrete examples of the different cases are discussed.

In the mapping experiment from Apertium into FreeLing 127 and 152 minimal units were automatically identified respectively. The found mapping correspondences between them are shown in table 2.

# possible mappings	#units
0	19
1	99
2	8
3	1
Total	127

Table 2: Number of units that receive a concrete number of correspondences (mappings)

Note that mapping correspondences are learnt only if enough examples are seen. A threshold mechanism over the similarity measures controls the selection of the mapping rules to be applied. The most common cases were learnt satisfactorily, and the mapping of units with the lowest frequency had different results. For instance, the mapping of Apertium "type=sup" for superlative adjectives was not found to be correlated with the FreeLing "grade=superlative", mainly due to the little number of examples in FreeLing. On the other hand, Apertium lexicon contained only two examples of "future of subjunctive" but in FreeLing lexicon all verbs do have these forms and the system correctly learnt the mapping. There were also incorrect mappings, which, however, affected only few cases which could be traced back after the inspection of the inferred mapping rules.

Finally, there were some cases where no correspondence was found and a manual inspection of these cases confirmed that, indeed, they should not have a mapping. For example, there were some PoS tags in Apertium that had no correspondence in FreeLing: *proper noun* and *acronym*. The merging mechanism was the responsible of adding the entries with these tags to the resulting lexica.

As we said before, the lexical entries in the resulting lexicon may have three different origins: from unification of an entry in lexicon A and in lexicon B; from entries that did not unify although having the same lemma, and from entries whose lemma was not in one of the lexica. In the following tables a summary of the results of the different unification results are given.

PoS	# LE	PoS	# LE
adjectiveQualifier	5,206	interjection	13
adpositionPreposition	24	nounCommon	14,147
adverbGeneral	112	pronoun	4
conjunctionCoordinated	4	pronounExclamative	8
conjunctionSubordinated	8	pronounIndefinite	12
determinantExclamative	4	pronounRelative	9
determinantIndefinite	12		

Table 3: Number of entries with the same information in lexicon A and in lexicon B per categories

PoS	# LE	PoS	# LE
adjectiveQualifier	561	determinantIndefinite	4
adpositionPreposition	0	interjection	2
adverbGeneral	11	nounCommon	792
conjunctionCoordinated	1	pronounDemonstrative	3
conjunctionSubordinated	1	pronounExclamative	2
determinantIndefinite	4	pronounIndefinite	1
determinantExclamative	0	pronounPersonal	4
verbAuxiliary	1	pronounPossesive	7
verbMain	3,929	pronounRelative	1

Table 4: Entries that gained information with the unification per categories

PoS	#LE	PoS	#LE
adjectiveOrdinal	4	num	11
adjectiveQualifier	1,138	preadv	11
adpositionPreposition	1	pronoun	2
adverbGeneral	41	pronounDemonstrative	2
adverbNegative	1	pronounExclamative	2
cnjsub	1	pronounIndefinite	33
conjunctionCoordinated	5	pronounPersonal	13
conjunctionSubordinated	13	pronounPossessive	3
determinantArticle	1	pronounRelative	3
determinantDemonstrative	5	np	5
determinantExclamative	1	punctuation	1
determinantIndefinite	28	vbmod	2
determinantPossessive	2	verbAuxiliary	1
interjection	51	verbMain	8
nounCommon	1,978	predet	1

Table 5: Lexical Entries in both lexica that did not unify

As explained before, for the cases in table 5 where, although having the same lemma, the entries did not unify

the system creates a new entry. This step might cause some undesirable results. This is the case of *no*, encoded as negative adverb in FreeLing with a special tag, where in Apertium it is encoded as a normal adverb. The system creates a new entry, and therefore a duplication. These cases can be traced back when inspecting the log information. The most numerous cases, common nouns and adjectives, mostly correspond to the case of nouns that can also be adjectives, for instance *accesorio* ('incidental' when adjective and 'accessory' when noun). In that case unification fails because of the different PoS value. The system creates a new entry in the resulting lexica, in that case correctly.

4. Dicussion

From the results presented above, we can see that using graph unification as merging technique is a successful approach. This method combines compatible information and detects incompatible one, allowing us to keep track of possible merging errors.

Furthermore, the results showed that the technique proposed by Bel et al. (2011) to automatically learn a mapping between lexica that originally encoded information in different ways, have a very good performance in this task. The algorithm correctly learned mapping rules between most of the elements, including those that imply a change in the structure or those that have very few examples in one of the lexica.

In this work we have focused in the use of LMF lexica to test the merging technique, which eases the conversion to feature structures. Though the use of other formats or the conversion to such formats to LMF is an interesting line to be studied in the future, in our opinion the use of LMF is very interesting for different reasons: first of all, because it is a standard format and secondly because it allows the encoding of very complex structures and the possible relations among them. If such structures are encoded in LMF, it is still possible to convert them to feature structures and to perform the automatic mapping and merging, but if these structures are encoded in other formats, discovering them automatically and converting them to a common format with a blind process is much more difficult.

As with respect with previous work, one difference between the application of this technique to SCFs lexica and to morphological lexica is that in the first case, the feature structures obtained after applying the automatic mapping were often incomplete in the sense that some parts of the SCF were partially translated to feature structures and some information was lost. This was overcome in most of the cases at unification step, where the missing information was obtained by subsumption from the target lexicon. Nevertheless, this is not the case in the experiments presented here. In this case, most of the feature structures obtained after applying the mapping are complete and keep all information encoded in the original lexicon. This is partly due to the fact that morphological dictionaries are probably more systematic than SCF lexica, where the SCFs assigned to each verb often have an important variability among lexica. Nevertheless, the improvement observed in the task of merging morphological lexica is also associated to the fact of working with LMF lexica, which allows us to perform a more systematic conversion to feature structures and eases the step of comparing elements of the two lexica. Thus, we can conclude that working with LMF lexica leads to a better performance of our algorithm.

The evaluation we presented here is only qualitative. A proper quantitative intrinsic evaluation will be done by manual inspection as no gold-standard is available. It is also pending to evaluate more accurately the obtained results in an extrinsic evaluation, that is, by assessing the consequences of the errors in another task for comparison.

Summarizing, we have presented an adaptation of the merging mechanism proposed by Bel et al. (2011) to work with LMF lexica that performs fully automatically all steps involved in the merging of two lexica. Furthermore, we have generalized this model to deal with open valued features. This issue was not tackled in the previous work, but is crucial to apply the method to different kind of lexica where this kind of features will be found. The obtained results showed the feasibility of the approach and confirm that this technique can be successfully applied to different kind of lexica.

5. Conclusions and Further Work

In this work we have applied the method for automatically merging lexical resources proposed by Bel et al. (2011) to the case of merging morphological dictionaries. These dictionaries were encoded in LMF format, so first of all we adapted the method to work with this standard format and generalized it to deal with open valued features.

The presented experiments showed, on the one hand, that using this method to automatically map two lexica into a common format and then merge them using graph unification mechanism performed satisfactorily in the task tackled in this work. This allows us to make an important step forward to demonstrate the generality of this approach, since it lead to satisfactory results in two very different scenarios: the merging of SCF lexica and the merging of morphological dictionaries.

On the other hand, we have also shown that using LMF as the source encoding format eases the merging process and probably contributes to a better performance of the system.

For that reason, one interesting future line is to study the feasibility of using an approach similar to the mapping technique presented here to convert lexica in any format into LMF. This will help lexicon developers to have their lexica in a standard format.

Another line to be studied in the future is the development of patch rules to refine the obtained results. These patch rules would be the only part of the method dependent on the concrete lexica to be merged and will be developed after systematically detecting possible errors. Besides, we also expect that in order to maximize the usability of the resulting lexica, some scripts can tune the richer and larger lexicon achieved by automatic merging to the requirements of a particular application.

Finally, we are also interested in testing the method for merging other LMF lexica with more or different information (e.g. containing sense information) and especially to apply the proposed technique to the merging of lexica with different levels of information, for example combining the morphological dictionaries with SCF information to obtain a richer, multi-level lexicon.

6. Acknowledgements

This work was funded by the EU 7FP project 248064 PANACEA and has greatly benefited of the results of the work done by Marta Villegas in METANET4U project.

7. References

Carme Armentano-Oller, Antonio M. Corbí-Bellot, Mikel L. Forcada, Mireia Ginestí-Rosell, Marco A. Montava, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Gema Ramírez-Sánchez, Felipe Sánchez-Martínez. 2007. Apertium, una plataforma de código abierto para el desarrollo de sistemas de traducción automática. *In Proceedings of FLOSS (Free/Libre/Open Source Systems) International Conference*, p. 5-20. Jerez de la Frontera, Spain.

Núria Bel, Muntsa Padró and Silvia Necsulescu. 2011. A Method Towards the Fully Automatic Merging of Lexical Resources. *In Proceedings of Workshop on Language Resources, Technology and Services in the Sharing Paradigm, at IJCNLP 2011*. Chiang Mai, Thailand.

Steven Bird. 2006. NLTK: the natural language toolkit. *In Proceedings of the COLING/ACL on Interactive presentation sessions.* Association for Computational Linguistics, Morristown, NJ, USA.

Daniel K. Chan and Dekai Wu. 1999. Automatically Merging Lexicons that have Incompatible Part-of-Speech Categories. *In Proceedings of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*. Maryland.

Dick Crouch and Tracy H. King. 2005. Unifying lexical resources. *In Proceedings of Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes.* Saarbruecken; Germany.

Christiane Fellbaum. 1998. WordNet: An Electronic Lexical Database. *MIT Press*.

Gil Francopoulo, Núria Bel, Monte George, Nicoletta Calzolari, Mandy Pet, and Claudia Soria. 2008. Multilingual resources for NLP in the lexical markup framework (LMF). *Journal of Language Resources and Evaluation, 43 (1).*

John Hughes, Clive Souter, and E. Atwell. 1995. Automatic Extraction of Tagset Mappings from Parallel-Annotated Corpora. *Computation and Language.*

Nancy Ide and Harry Bunt. 2010. Anatomy of Annotation Schemes: Mapping to GrAF. *In Proceedings of the Fourth Linguistic Annotation Workshop, ACL 2010.*

Martin Kay. 1979. Functional Grammar. *In Proceedings of the Berkeley Linguistics Society,* pp 142-158.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. *In Proceedings of AAAI/IAAI.*

Geoffrey Leech and Andrew Wilson. 1999. Recommendations for the Morphosyntactic Annotation of Corpora. *EAGLES Report EAG-TCWG-MAC/R.*

Doug Lenat. 1995. Cyc: a large-scale investment in knowledge infrastructure. *In CACM 38*, n.11.

Silvia Necsulescu, Núria Bel, Muntsa Padró, Montserrat Marimon and Eva Revilla: Towards the Automatic Merging of Language Resources. *In Proceedings of WoLeR 2011*. Ljubljana, Slovenia.

Lluís Padró, Miquel Collado and Samuel Reese and Marina Lloberes and Irene Castellón. FreeLing 2.1: Five Years of Open-Source Language Processing Tools. *Proceedings of 7th Language Resources and Evaluation Conference (LREC 2010), ELRA*. La Valletta, Malta. May, 2010.

Simone Teufel. 1995. A Support Tool for Tagset Mapping. *In Proceedings of EACL-Sigdat 95*.

Merging syntactic lexica: the case for French verbs

Benoît Sagot, Laurence Danlos

Alpage, INRIA Paris-Rocquencourt & Université Paris Diderot, 175 rue du Chevaleret, 75013 Paris, France
benoit.sagot@inria.fr, laurence.danlos@linguist.jussieu.fr

Abstract

Syntactic lexicons, which associate each lexical entry with information such as valency, are crucial for several natural language processing tasks, such as parsing. However, because they contain a rich and complex information, they are very costly to develop. In this paper, we show how syntactic lexical resources can be merged, in order to take benefit from their respective strong points, and despite the disparities in the way they represent syntactic lexical information. We illustrate our methodology with the example of French verbs. We describe four large-coverage syntactic lexicons for this language, among which the Le*fff*, and show how we were able, using our merging algorithm, to extend and improve the Le*fff*.

1. Introduction

Syntactic lexicons are crucial for several natural language processing tasks, such as parsing, be it symbolic (Riezler et al., 2002; Thomasset and Éric de La Clergerie, 2005) or even statistical (Collins, 1997; Versley and Rehbein, 2009). Syntactic lexicons are rich and complex resources, and their development is a costly task. Although a lot of work has been published on the automatic acquisition of syntactic lexica, the resources that have a coverage and an accuracy large enough for being used as linguistic descriptions, e.g., in symbolic parsers, have been developed manually or semi-automatically, sometimes for several decades.

In this paper, we focus our study on French verbs. There exist today four large-coverage syntactic lexical resources for French, that provide information about the valency of lexical entries, i.e., subcategorization frames and other syntactic information relevant for describing the syntactic behaviour of predicative lexical units. These resources are Lexicon-Grammar tables (Gross, 1975; Boons et al., 1976b; Boons et al., 1976a; Guillet and Leclère, 1992), the verb valency lexicon Dicovalence (van den Eynde and Mertens, 2006), the verbal syntactico-semantic lexicon LVF (Dubois and Dubois-Charlier, 1997), and the Le*fff* (Sagot et al., 2006; Sagot, 2010). All these resources use both syntactic and semantic criteria for defining either one or several entries for the same verb lemma. Therefore, these lexicons can be considered as an inventory of *lexemes* (as opposed to verb lemmas) associated with syntactic information.

The objective of this paper is to show how these diverse resources can be leveraged for improving one of them, the Le*fff*, by developing and applying merging techniques for valency lexicons. In this paper, we limit ourselves to verbal entries, for at least two reasons. First, they are best covered in terms of syntactic information than other categories. For example, LVF and Dicovalence only cover verbs. Second, verb valency is crucial in the first NLP application of syntactic lexicons, namely parsing systems.

Merging syntactic lexicons is not a straightforward task. Indeed, there is no real consensus on the way syntactic information should be modeled and formalized. There are discrepancies among resources, which differ in various ways:

- coverage: for example, Dicovalence has focused on reasonably frequent entries of fairly frequent verbal lemmas, wheras LVF has tried to have as large a coverage as possible;
- level of granularity of the set of entries for a given lemma (i.e., level of granularity used for distinguishing lexemes from one another): for example, LVF can distinguish two entries which differ only at a very fine-grained semantic level, whereas other resources will contain only one corresponding entry (see examples below);
- nature and level of granularity of the syntactic properties they describe: for example, Lexicon-Grammar tables include a large amount of non-standard syntactic information (e.g., symetric verbs), but does not really cover reflexive and reciprocal realizations using the pronoun *se*, whereas Dicovalence only describes pronominal realizations of syntactic functions which include the reciprocal and reflexive *se* realizations,
- level of formalization: Dicovalence and the Le*fff* are immediately usable in NLP tools, contrarily to LVF or Lexicon-Grammar tables;
- definition of what is considered syntactic argument as opposed to an adjunct: Dicovalence considers as arguments complements that other resources sometimes consider as adjuncts.

The methodology we have developed for merging syntactic lexicons has been developed in the last years (Sagot and Danlos, 2008; Sagot and Fort, 2009; Sagot and Danlos, 2009; Molinero et al., 2009). Other teams have worked on this task, such as Crouch and King (2005) for English and Necşulescu et al. (2011) for Spanish. They address in different ways the issue of mapping lexical entries for a given lemma from various input lexicons to the one another, although these entries might have been defined at least in part using semantic criteria. In the work by Crouch and King (2005), the authors rely on the fact that, in (some of) their input lexicons (VerbNet and Cyc), lexical entries, which correspond to lexemes, are associated with WordNet synsets. This allows them to put together lexical en-

tries that are associated with identical or related senses, although they resort to non-trivial techniques for dealing with various types of discrepancies and inconsistencies. On the other hand, Necşulescu et al. (2011) simply want to merge subcategorization lexicons, i.e., lexicons that list all possible subcategorization frames for a given verb lemma (as opposed to lexeme). With their strategy, they avoid the need for correctly mapping to the one another lexical entries that are defined based on syntactico-semantic criteria. However, the resulting lexicon is then only a subcategorization lexicon, and not a full-featured syntactic lexicon associating syntactic information with each lexeme.

In our case, our input resources for French verbs do not contain WordNet synset information. Nevertheless, we do want to take advantage of sense distinctions between entries, and to produce a merged lexicon at the lexeme level, that preserves these sense distinctions to the appropriate extent. Our methdology can be sketched as follows. First, we chose a model for representing syntactic information, and convert all input resources in this model, after a careful linguistic analysis. In this paper, this common model is Alexina, the lexical framework on which the Le*fff* is based. This is because Alexina lexicons, as mentioned above, are immediately usable in NLP tools. Moreover, and contrarily to our other input lexicons, the Le*fff* strongly relies on the notion of syntactic function, which is the basis for many parsing systems. In a second step, we try and create *groupings*, i.e., sets of lexical entries possibly extracted from more than one input resources and that will be merged in one entry in the output lexicon. Finally, we perform the actual merging. Such a methodology is useful for various reasons. Of course, it helps developing a resource that has a higher coverage and accuracy than all input resources, although some information might be lost during the conversion process. Second, it allows for an efficient manual work on the output resource, if such a work is considered; for example, pieces of information that originate in only one of the input resources are more dubious than others. Finally, as a consequence, it allows for detecting errors in the input lexicons, as will shall see below.

After a brief description of our four input syntactic lexicons in Section 2. illustrated with a running example, we describe in more details our merging methodology and algorithm (Section 3.). Then, we describe a set of experiments conducted in the last years that are based on this methodology (Section 4.). Finally, we draw several conclusions and indicate the next steps for this work.[1]

2. Input resources

We shall not provide a detailed description of our input resources. Such descriptions can be found in the various publications related to each resource (see citations below). Rather, we shall illustrate these resources on a running example, the lemma *vérifier* 'check', 'verify'. In the reminder of this paper, we refer to the entry with id n for the lemma v in the resource i as v_n^i. For example, the (only) entry in the Le*fff* for the lemma *vérifier* is $\text{vérifier}_1^{\text{Le}fff}$. For simplification purposes, we use v_n^i both for the lexical entry in its original form and after its conversion in Alexina.

2.1. The Le*fff* and the Alexina lexical formalism

The Le*fff* (Lexique des formes fléchies du français — *Lexicon of French inflected form*) is a large-coverage syntactic lexicon for French (Sagot, 2010).[2] The current version of the Le*fff* (which is not the last one, as explained below) contains 10,214 entries for 7,813 distinct lemmas. Contrarily to the three other lexicons we have used, which were developped manually, the Le*fff* was developed in a semi-automatic way: automatic tools were used together with manual work (Sagot et al., 2006; Sagot, 2010).

The Le*fff* relies on the Alexina framework for the acquisition and modeling of morphological and syntactic lexicons. To represent lexical information, an Alexina lexicon relies on a two-level architecture:

- the *intensional* lexicon associates (among others) an inflection table and a canonical sub-categorization frame with each entry and lists all possible redistributions from this frame;
- the *compilation* of the intensional lexicon into an *extensional lexicon* builds different entries for each inflected form of the lemma and every possible redistribution.

The version of the Le*fff* that was available before the experiments described below (version 3.0b) contains only one entry for the lemma *vérifier*. Here is a simplified version of this entry:

$\text{vérifier}_1^{\text{Le}fff}$ *Lemma;v;<Suj:cln|sn, Obj:(cla|qcompl|scompl|sinf|sn)>; %ppp_employé_comme_adj,%actif,%passif, %se_moyen_impersonnel,%passif_impersonnel*

It describes a transitive verb whose arguments have the *syntactic functions* *Suj* and *Obj* listed between angle brackets, and which allows for the functional redistributions *past participle used as an adjective*, *active* (the default distribution), *impersonal middle-voice "se" construction*, *impersonal passive*, and *passive*.

The different syntactic functions are defined in the Le*fff* by criteria close to that used in Dicovalence, i.e., they rely for a large part on cliticization and other pronominal features. The Le*fff* uses the following syntactic functions: *Suj* (subject), *Obj* (direct object), *Objà* (indirect object canonically introduced by preposition "à"), *Objde* (indirect object canonically introduced by preposition "de"), *Loc* (locative), *Dloc* (delocative), *Att* (attribute), *Obl* or *Obl2* (other oblique arguments).

Each syntactic function can be realized by three types of *realizations*: *clitic pronouns*, *direct phrases* (nominal phrase (*sn*), adjectival phrase (*sa*), infinitive phrase (*sinf*), completive (*scompl*), indirect interrogative (*qcompl*)) and *prepositional phrases* (direct phrases preceded by a preposition,

[1] If the paper is accepted, we will report on results we have obtained while trying to evaluate various syntactic lexicons by comparing the results of one of the best performing symbolic parsers for French when it uses one of these lexicons or another. These results are not included in this submision for space reasons.

[2] The Le*fff* is freely available under the LGPL-LR license. See http://gforge.inria.fr/projects/alexina/

such as *de-sn*, *à-sinf* or *pour-sa*). Finally, a function whose realization is not optional has its realizations list between angle brackets.[3]

The way morphological and syntactic information is encoded it the Le*fff* is such that the Le*fff* be directly used in NLP tools. For example, we are aware of several parsers using the Le*fff*, and based on various formalisms: LTAG, including LTAGs generated from meta-grammars developed in various meta-grammar formalisms (Thomasset and Éric de La Clergerie, 2005), LFG (Boullier and Sagot, 2005), and less well-known formalisms such as Interaction Grammars or Pre-Group Grammars.

2.2. Lexicon-Grammar tables

In the Lexicon-Grammar (Gross, 1975; Boons et al., 1976b; Boons et al., 1976a; Guillet and Leclère, 1992), the 14,000 entries for verb lexical entries are structured in the form of 61 *classes*, each class being described in a different *table*.[4] Each class (table) is defined by a *defining property*, which is valid for all lexical entries belonging to the class (i.e., the defining property described a sub-categorization that is valid for all entries in the class, although other sub-categorizations might be also valid for a given entry). Lexicon-Grammar tables include two entries for the lemma *vérifier*, which both belong to class 6. Let us illustrate the notion of defining property and the content of the corresponding Lexicon-Grammar table using these entries.

The defining property for class 6 is N_0VQuP, which means that all entries in this class are transitive and may have a finite or infinitive clause as the realization of the second argument in addition to the default noun phrase realization. Note that the notion of syntactic function is absent from the Lexicon-Grammar model. In table 6, 40 additional properties are "coded", i.e., each entry specifies whether it has each property or not, in the form of a matrix with one entry per row and one property per column. Among these 40 properties (the set of properties differs from one table to another), we can cite for example $N_1 =: QuPind$ (if the second argument is a finite clause, its verbal head is at the indicative mood) or $N_1 =: N_{\mathrm{hum}}$ (its second argument can be human).

The two Lexicon-Grammar entries the lemma *vérifier* are associated (among others) with the following properties (including the defining property for class 6):

$\textit{vérifier}^{\mathrm{LG}}_{6\_504}$	*Aux =: avoir*
	N0 V
	N1 =: Qu Pind
	N1 =: Qu P = Ppv
	N1 =: N-hum
	[passif par]
	Ex: Max a vérifié que la porte était fermée
	'Max checked that the door was closed'
$\textit{vérifier}^{\mathrm{LG}}_{6\_505}$	*(unknown, the lexical entry is not yet coded)*
	Ex: Les faits vérifient cette hypothèse
	'The facts validate this hypothesis'

The second entry is not yet coded: the only thing we know about this entry is that it satisfies the defining property. As for the first one, the properties we have indicated here show respectively that its periphrastic inflected froms are built using the auxiliary *avoir*, that the second argument (N_1) is not mandatory, that it can be realized (among others) as a finite clause whose verbal head is at the indicative mood, as a pre-verbal particle (a clitic pronoun) or as a non-human noun phrase, and finally that it can be passivized (the subject becoming a non-mandatory argument introduced by the preposition *par*).

2.3. Dicovalence

Dicovalence (van den Eynde and Mertens, 2006) is a verb valency lexicon for French that is a follow up to the PROTON lexicon.[5] It was developed in the Pronominal Approach framework (Blanche-Benveniste et al., 1984). In order to identify the valency of a predicate (i.e., its dependants and their properties), the Pronominal Approach uses the relation that exists between so-called *lexicalized* dependants (realized as syntagms) and pronouns that "intentionally cover" these possible lexicalizations. Pronouns (and "paranouns", cf. below), contrarily to syntagms, syntactic functions or thematic roles, have two important advantages: (1) they are purely linguistic units, and do not have any of the properties (e.g., semantic properties) that make grammaticality judgements about sentences with lexicalized dependants difficult to motivate; (2) there are only a limited amount of such units: their inventory is finite. Note that the pronouns used in Dicovalence are more numerous than what is usually called a pronoun. Indeed they also include what Dicovalence calls "paranouns", that differ from pronouns because they can be modified (as *rien* 'nothing' in *rien d'intéressant* 'nothing interesting') and because they can not be taken up by a syntagm (cf. **il ne trouve rien, les preuves* 'He finds nothing, the evidences', vs. *il les trouve, les preuves* 'He finds them, the evidences').

In Dicovalence, pronouns are grouped in *paradigms*, which correspond only approximately to syntactic functions (e.g., P0 corresponds to the subject, P1 to the direct object, and so on). But Dicovalence contains more paradigms than the usual inventories contain syntactic functions. For example, it licenses a quantity paradigm (PQ), a manner paradigm (PM) and others.

The version of Dicovalence used in the experiments described below[6] consists in a list of 8,214 entries for 3,729 unique verbal lemmas. These lemmas and entries are explicitely chosen because they are reasonably frequent.

Table 1 shows both (simplified) entries given for the lemma *vérifier* in Dicovalence. These two entries exactly correspond to the two entries found in the Lexicon-Grammar: The example in entry 85770 means 'I will check this piece of information before I publish it', and the example in entry 85780 'The experiment validated his hypothesis'.

[3] Other information are encoded in the Le*fff*, such as control, mood for finite clause argument realizations, and others.

[4] Lexicon-Grammar tables are freely available under the LGPL-LR license. See http://ladl.univ-mlv.fr/.

[5] Dicovalence is freely available under the LGPL-LR license. See http://bach.arts.kuleuven.be/dicovalence/

[6] It is the version labeled 061117, which is not the last version. Experiments about the last version of Dicovalence are planned.

vérifier$^{DV}_{85770}$	
VAL$	vérifier: P0 (P1)
VTYPE$	predicator simple
EG$	je vérifierais cette information avant de la publier
P0$	qui, je, nous, elle, il, ils, on, celui-ci, ceux-ci
P1$	0, que, la, le, les, en Q, ça, ceci, celui-ci, ceux-ci, le(qpind), ça(qpind), le(qpsubj), ça(qpsubj), le(sipind), ça(sipind), le(indq), ça(indq)
RP$	passif être, se passif

vérifier$^{DV}_{85780}$	
VAL$	vérifier: P0 P1
VTYPE$	predicator simple
EG$	l'expérience a vérifié son hypothèse
P0$	que, elle, il, ils, ça, celui-ci, ceux-ci
P1$	que, la, le, les, en Q, ça
RP$	passif être, se passif

Table 1: Entries for *vérifier* in Dicovalence.

2.4. The *Lexique des Verbes Français*

The LVF (*Lexique des Verbes Français*) is a dictionary of French verbs developed by Dubois and Dubois-Charlier (Dubois and Dubois-Charlier, 1997) that has the form of a thesaurus of syntactico-semantic classes, i.e., semantic classes defined using syntactic criteria. It is a very large coverage resource, that gathers syntactic and semantic information. The different classes, that contain 25,610 entries, are defined by both syntactic and semantic features and form a three-level hierarchy. At the lowest level of the hierarchy, sub-sub-classes are either homogeneous in terms of syntactic behaviour, or are divided once more in sets of entries with entries that all have the same syntactic behaviour. However, these syntactic behaviours are coded in a compact but abstruse way, that we shall illustrate on our running example.

In LVF, *vérifier* has three distinct entries. Two of them are in the sub-sub-class P3b of transitive verbs "of the type 'target one's thinking activity towards something'". It belongs to the sub-class P3 for verbs expressing the 'manifestation of a thinking activity towards somebody or something', which is a sub-class of the larger class P of psychologic verbs. The last entry belongs to class D of verbs like *donner* 'give', sub-class D3 containing verbs with a figurative meaning "giving something to somebody" or "obtaining something from somebody", sub-sub-class D3c of verbs meaning "granting validity to something or value to somebody". These entries contain, among other things, the following information:

vérifier$^{\mathrm{LVF}}_{1}$	*P3b*	*T1400 P3000*
vérifier$^{\mathrm{LVF}}_{2}$	*P3b*	*T1300 P3000*
vérifier$^{\mathrm{LVF}}_{3}$	*D3c*	*T3300*

The third column contains the syntactic codes. For example, code T1400 indicates a transitive construction with a human subject and a non-human (nominal) or clausal direct object. Code P3000 a pronominal construction with a non-human subject. T3300 stands for a transitive construction with non-human nominal subject and object, whereas T3100 stands for a transitive construction with a non-human nominal subject and a human object. On the one hand, these examples show, although not very clearly, a general fact: syntactic descriptions in LVF are less fine-grained than those found in other resources, except for semantic properties of the arguments, in particular prepositional ones. On the other hand, one can see that the inventory of lexical entries is more fine-grained than in other resources: the first two entries introduce a distinction that is not present in Dicovalence or in Lexicon-Grammar tables, which puts them together in only one entry (*vérifier*$^{DV}_{85770}$ and *vérifier*$^{\mathrm{LG}}_{6\_504}$). The third entry directly matches entries *vérifier*$^{DV}_{85780}$ and *vérifier*$^{\mathrm{LG}}_{6\_505}$).

3. Merging algorithm

As sketched in the Introduction, our merging algorithm is a three-step process (Sagot and Danlos, 2008):

- converting all input resources into the common model, which, as explained above, is Alexina; all converted resources must use the same inventory of syntactic functions, realizations and redistributions — in our case, that of the L*efff*; the main challenge at this stage is to be able to extract as much information as possible from the input resources and encode them in the form of an Alexina lexicon, despite all discrepancies between resources, as seen in the previous section;
- creating *clusters* of entries from various resources such that the entries in each should be merged into one entry; this step is very challenging, as its aim is to address the discrepancies in the granularity of lexical entries from one lexicon to another; for example, it is reasonable to consider that entries *vérifier*$^{DV}_{85770}$, *vérifier*$^{\mathrm{LG}}_{6\_504}$ and both *vérifier*$^{\mathrm{LVF}}_{1}$ and *vérifier*$^{\mathrm{LVF}}_{2}$ for a unique grouping
- merging of these clusters into output lexical entries.

3.1. Converting input lexicon in the L*efff* format

The way the lexical information is structured in the L*efff* is not very different from what can be found in **Dicovalence**. This makes the conversion process for Dicovalence reasonably straightforward. It is based on the following principles, which are obviously approximations:

- each Dicovalence *paradigm* is mapped into a L*efff syntactic function*;[7]
- each pronoun (or paranoun) in the Dicovalence paradigm is mapped into a L*efff* realization: for example, if the pronoun *te* belongs to paradigm *P1*, a realization *cla* (accusative clitic) is added to the syntactic function *Obj* (we lose here the fact that the direct object can be human);
- each Dicovalence reformulation is converted into a L*efff* redistribution.

[7] We insist on the fact that this is an approximation.

Converting **Lexicon-Grammar tables** into the Alexina format is much more complex a task. Although the extraction of an NLP-oriented lexicon from Lexicon-Grammar tables has raised interest for some time (Hathout and Namer, 1998; Gardent et al., 2005), the only attempt that was successful in producing and using in a parser an NLP-lexicon from *all* lexicon-grammar tables is the work by Tolone and colleagues (Tolone and Sagot, 2011). The final output of this conversion process is an Alexina lexicon that is consistent with the Le*fff* in terms of syntactic function, realization and redistribution inventories, and in terms of linguistic modeling. It is what we shall call the "full" Alexina version of Lexicon-Grammar tables.

Because this conversion process is complex, and before its results were available, we have also directly extracted a **"light" Alexina version of Lexicon-Grammar tables**, in the context of the work about pronominal constructions described in Section 4.1. (Sagot and Danlos, 2009). For each entry we have retained for this work, we only extracted its functional sub-categorization frame, i.e., the list of syntactic functions without their possible realizations, as well as some redistributions (active, passive and *se*-middle).

Building an **Alexina version of LVF** was simply achieved by parsing valency data (codes such as T3100) and generating on-the-fly the corresponding Alexina entries. The only pieces of information that required a few heuristics are syntactic functions, which are not all explicitely recoverable from LVF codes, but that can be inferred using LVF information of argument introducters (e.g., prepositions) and semantic types.

In the remained of this paper, E^i_n is the lexical entry n in the lexicon i after it is converted in the Alexina format. Thus, i can be Dicovalence, LVF or Le*fff*, as well as LG for the "full" Alexina variant of Lexicon-Grammar tables, and LG-light for the "light" version.

3.2. Grouping entries from various lexicons

For a given lemma, each resource might contain more than one entry. Therefore, it is necessary to determine the number of entries in the output merged lexicon, each of them being obtained by merging together one or several entries from each input lexicon, according to an algorithm detailed below. This means that the first step before merging is to build sets of entries for each lemma, each set corresponding to one output entry. We shall call such sets *groupings*.

Building such groupings is a challenging task. Indeed, what we need here is to identify cross-resource correspondances between entries, that are not necessarily one-to-one. Moreover, we can only rely on the information that is available in the input resources, i.e., mainly syntactic information, whereas distinctions between entries are often, at least in part, semantic. On the other hand, we have seen, while describing our input resources, and in particular the example of the entries for *vérifier*, that these resources have various granularities. In the case of *vérifier*, we can see that LVF entries are more fine-grained than Lexicon-Grammar and Dicovalence entries, which in turn are more fine-grained than the unique entry in the Le*fff*. It turns out that this ordering of the resources is the same for most verbal lemmas. Therefore, we have chosen to base our grouping algorithm on an *inclusion* relation, which formalizes this intuition.

We first define this *inclusion* relation at the entry level as follows. For a given lemma, an entry E_1 is *included in* or *more specific than* an entry E_2 if and only if the set of clauses headed by an occurrence of entry E_1 is included in the set of clauses headed by E_2. Such an inclusion is noted $E_1 \subset E_2$. For example, $\textit{vérifier}^{DV}_{85770} \subset \textit{vérifier}^{Lefff}_{1}$.

Then, we generalize this inclusion relation at the resource level. Contrarily to the inclusion relation at the entry level, which can be defined without any problem, assuming that we can define an inclusion relation at the resource level is obviously an approximation, but it is required for being able to create these mappings. A lexicon i is considered included in or more specific than another lexicon j if we make the assumption that entries from lexicon i are *all* more specific than entries from lexicon j, i.e., each entry E^i_n is more specific than an entry E^j_m, except if the lexicon j does not contain any entry corresponding to E^i_n (in that case, i's coverage is higher than j's); assuming that i is more specific than j, this means that building groupings can be achieved by computing inclusion relations of the form $E^i_n \subset E^j_m$. Generalizing the example given above, we can posit that DV $\subset$ Le*fff*.

We compute such inclusion relations using the following heuristics: starting from the resource-level hypothesis that $i \subset j$, an entry E^i_n is considered included in another entry E^j_m if it has exactly the same set of arguments with one of the base syntactic functions (subject, direct object, indirect object introduced by *à*, indirect object introduced by *de*) and if the set of syntactic functions of remaining arguments in E^j_m is included in that of E^i_n. One can see that we only rely on the inventories of syntactic functions. Moreover, we consider that only base synctactic functions can be used as safe clues, and that more oblique ones are likely to be found only in the most specific resource — but if one is found in the least specific resource, then it has to be in the specific one. In our case, this algorithm satisfyingly computes the relation $\textit{vérifier}^{DV}_{85770} \subset \textit{vérifier}^{Lefff}_{1}$, as they both entail, after conversion in the Alexina format, the syntactic functions *Suj* and *Obj* (the Alexina version of $\textit{vérifier}^{DV}_{85770}$ is shown below).

The groupings are then build as follows: we start from each entry that includes no other entry (often, an entry from the most specific lexicon) and we follow all inclusion relations until we reach entries that are include in no other entries. The set of all entries that are reached constitute a grouping. Of course, a grouping might end up containing entries from one resource only, if it is not included in any entry from another lexicon. If this is because this entry corresponds to a meaning or a valency that is not covered by other resources, this is the expected result. However, it might be the result of mismatches resulting from the original resources, either because of errors in an input lexicon or because there are differences in the way a same construction is analyzed (e.g., a resource might consider as an indirect object in *à* what another resource analyzes as a locative argument). These problems, as well as the fact that entries might be incomplete (see the case of $\textit{vérifier}^{\mathrm{LG}}_{6\_505}$), might also provoke erroneous groupings.

But this algorithm could still be improved. First, it can

never put two distinct entries from the same lexicon in a same grouping. Going back to our running example, this algorithm would cluster all entries for *vérifier* in three groupings, each of them containing one of the three LVF entries, although we might wish only two. Second, restricting the information used for creating groupings to the inventory of syntactic functions is not always precise enough. In our running example, a correct mapping between LVF and Dicovalence entries for *vérifier* would require using the information about the human vs. non-human features applied to the subject. These improvements will be implemented in the future.

3.3. Merging entries

Once the groupings are built, we merge the entries in each grouping in a relatively straightforward way:

- the set of syntactic functions is built as the union of the set of syntactic functions in the input entries;
- for each syntactic function, the set of realizations is also obtained by union; for each realization we indicate which sources include it (no indication is added if it is licensed by all entries in the grouping);
- a realization is considered mandatorily realized only if it is mandatory in all entries in the grouping,
- the set of possible redistributions is built as the union of all possible redistributions in all entries in the grouping.

Let us illustrate the output of the merging of a grouping containing the entries $vérifier_{85770}^{\text{Dicovalence}}$ and $vérifier_{1}^{\text{Lefff}}$. The Le*fff* entry has been shown above. The Dicovalence entry in its original format was also given. Once converted in the Alexina model, it has become:

$vérifier_{85770}^{\text{DV}}$ *Lemma;v;<Suj:cln|sn, Obj:(sn|cla|scompl|qcompl)>; %actif,%passif,%se_moyen*

Applying the merging algorithm leads to the following entry:

$vérifier_{1}^{\text{Lefff}} +_{85770}^{\text{DV}}$ *Lemma;v;<Suj:cln|sn, Obj:(cla|qcompl|scompl|sinf$^{\text{Lefff}}$|sn)>; %ppp_employé_comme_adj,%actif, %passif_impersonnel,%passif, %se_moyen,%se_moyen_impersonnel*

Note that the infinitive relalization of the direct object only comes from the Le*fff*, and is marked as such. This allows for a more efficient manual validation, if required, as a piece of information that is only licensed by one resource is more dubious than others. In this case, it is valid.

4. Merging experiments

4.1. Improving the coverage of the Le*fff* on pronominal entries with Dicovalence and Lexicon-Grammar tables

In order to improve the coverage of the Le*fff* over pronominal entries and pronominal constructions (i.e., realizations using the reflexive or the reciprocal *se*), we have leveraged the syntactic information Dicovalence, and to a lesser extend, Lexicon-Grammar tables, under the "light" version described above (Sagot and Danlos, 2009).

First, we have carefuly described such constructions, and explored the way they were encoded in Dicovalence and in Lexicon-Grammar tables, as well as the way they were to be formalized in the Le*fff*. Then, we have extracted from Dicovalence and converted in the Alexina formalism the 5,273 entries that are either pronominal or that include realizations in *se*. Moreover, we have extracted 550 such entries using the "light" conversion scheme. We have merged the Le*fff* as well as these two additional sets of entries, using the inclusion relations DV $\subset$ Le*fff* $\subset$ LG-light. The result of the merging, which has since then be included in the Le*fff*, consists in 5,464 lexical entries.

4.2. Merging the Le*fff*, Dicovalence and LVF entries for denominal and deadjectival verbs in *-iser* and *-ifier*

In French, verbs in *-iser* and *-ifier* are particularly interesting. First, most of them are denominal or deadjectival verbs, which means they are relevant for studying the relation between morphological derivation and valency. Second, a large amount of verbal neologisms are built using one of these two morphological derivation mechanisms, and studying verbs in *-iser* and *-ifier* is an important step towards the development of tools for turning a syntactic lexicon into a *dynamic* lexicon that evolves in parallel with textual corpora.

Our work (Sagot and Fort, 2009) was based on the Le*fff*, Dicovalence and on LVF, which has a very large coverage. We have restricted it to verbs in *-iser* or *-ifier* that are indeed denominal or deadjectival, by manually removing other verbs ending "accidentally" in *-iser* or *-ifier*, such as *croiser* 'cross'. We relied on the following inclusion relations: LVF $\subset$ DV $\subset$ Le*fff*. The merging process created 2,246 entries covering 1,701 distinct lemmas (1,862 entries for verbs in *-iser* covering 1,457 distinct lemmas, and 384 entries for verbs in *-ifier* covering 244 distinct lemmas.

Note that this work was complemented with a corpus-based extraction step for finding missing denominal and deadjectival entries in *-iser* and *-ifier*.

4.3. Merging the Le*fff* and Dicovalence for increasing the granularity and the accuracy of the Le*fff*

In order, again, to increase the granularity and the accuracy of the Le*fff*, we have conducted a work aiming at merging the whole verbal lexicon of the Le*fff* and Dicovalence, and then validate or correct and/or merge manually the resulting entries. We have applied the methodology described in this paper using the inclusion relation DV $\subset$ Le*fff*. The, we have validated the 100 most frequent lemmas as well as all *dubious* lemma, i.e., those lemma who got more entries in the merged lexicon than originally in both input lexicons. This validation step allowed us to remove erroneous realizations that were present in the Le*fff*, to indeed extend its coverage, accuracy, and fine-grainedness, but also to unvail errors in Dicovalence itself. This illustrates what we have explained above: not only merging syntactic lexicons lead to a improved output resource, but it also allows to improve

the input resources themselves.

This work extended the number of verbal entries in the Le*fff* from 10,214 to 12,610, whereas the number of distinct lemmas was extended from 7,813 to 7,990 lemmas. The new version of the Le*fff* resulting from this automatic merging and manual validation and correction step is already freely available in the last distribution of the Le*fff*, but is not yet considered validated enough to replace the previous verbal lexicon files, which are therefore still distributed as well. It corresponds to the NewLe*fff* in the parsing evaluation work described in (Tolone et al., 2012).

5. Conclusion and next steps

In this paper, we have shown how syntactic lexical resources can be merged, in order to take benefit from their respective strong points, and despite the differences in the way they represent syntactic lexical information. We have described four large-coverage syntactic (including valency) lexicons for French, among which the Le*fff*, and have shown how we have used our merging algorithm for extending and improving the Le*fff*. In two experiments, we have merged up to 3 resources but restricting ourselves to two classes of entries. In the last experiments, all entries of only two lexicons were merged. Moreover, we used one of our input lexicons, namely Lexicon-Grammar tables, only in a light way, as explained below.

The next step of our work will be twofold. First, we will implement improvements that will address the limits of our grouping algorithm, as explained above. Second, we will finally merge our four lexical resources, including Lexicon-Grammar tables fully converted in the Alexina format (as opposed to the "light" version). This should give birth to a new version of the Le*fff*, which will then become the syntactic resource with the largest coverage, and hopefully a very high accuracy, concerning French verbs.

6. References

Claire Blanche-Benveniste, José Delofeu, Jean Stefanini, and Karel van den Eynde. 1984. *Pronom et syntaxe. L'approche pronominale et son application au français*. SELAF, Paris.

Jean-Pierre Boons, Alain Guillet, and Christian Leclère. 1976a. *La structure des phrases simples en français : Constructions intransitives*. Droz, Genève, Suisse.

Jean-Pierre Boons, Alain Guillet, and Christian Leclère. 1976b. La structure des phrases simples en français, classes de constructions transitives. Technical report, LADL, CNRS, Paris 7.

Pierre Boullier and Benoît Sagot. 2005. Efficient and robust LFG parsing: SXLFG. In *Proceedings of IWPT 2005*, pages 1–10, Vancouver, Canada.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Stroudsburg, PA, USA.

Dick Crouch and Tracy Holloway King. 2005. Unifying Lexical Resources. In *Proceedings of the Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*, Saarbrücken, Germany.

Jean Dubois and Françoise Dubois-Charlier. 1997. *Les verbes français*. Larousse-Bordas, Paris, France.

Claire Gardent, Bruno Guillaume, Guy Perrier, and Ingrid Falk. 2005. Maurice Gross' Grammar Lexicon and Natural Language Processing. In *Proceedings of the 2nd Language and Technology Conference (LTC'05)*, Poznań, Pologne.

Maurice Gross. 1975. *Méthodes en syntaxe : Régimes des constructions complétives*. Hermann, Paris, France.

Alain Guillet and Christian Leclère. 1992. *La structure des phrases simples en français : Les constructions transitives locatives*. Droz, Genève.

Nabil Hathout and Fiammetta Namer. 1998. Automatic construction and validation of French large lexical resources: Reuse of verb theoretical linguistic descriptions. In *Proceedings of the 1st Language Resources and Evaluation Conference (LREC'98)*, Granada, Spain.

Miguel Ángel Molinero, Benoît Sagot, and Lionel Nicolas. 2009. A morphological and syntactic wide-coverage lexicon for spanish: The Le*ff*e. In *Proceedings of RANLP 2009*, Borovets, Bulgaria.

Silvia Necşulescu, Núria Bel, Muntsa Padró, Montserrat Marimon, and Eva Revilla. 2011. Towards the automatic merging of language resources. In *Proceedings of WoLeR 2011, the 1st International Workshop on Language Resources*, Ljubljana, Slovenia.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 271–278, Stroudsburg, PA, USA.

Benoît Sagot and Laurence Danlos. 2008. Méthodologie lexicographique de constitution d'un lexique syntaxique de référence pour le français. In *Actes du colloque Lexicographie et informatique: bilan et perspectives*, Nancy, France.

Benoît Sagot and Laurence Danlos. 2009. Constructions pronominales dans dicovalence et le lexique-grammaire — intégration dans le lefff. *Linguisticæ Investigationes*, 32(2).

Benoît Sagot and Karën Fort. 2009. Description et analyse des verbes désadjectivaux et dénominaux en *-ifier* et *-iser*. In *Proceedings of the 28th Lexis and Grammar Conference*, Bergen, Norway.

Benoît Sagot, Lionel Clément, Éric de La Clergerie, and Pierre Boullier. 2006. The Le*fff* 2 syntactic lexicon for French: architecture, acquisition, use. In *Proceedings of the 5th Language Resource and Evaluation Conference*, Lisbon, Portugal.

Benoît Sagot. 2010. The Le*fff*, a freely available, accurate and large-coverage lexicon for French. In *Proc. of the 7th Language Resource and Evaluation Conference*, Valetta, Malta.

François Thomasset and Éric de La Clergerie. 2005. Com-

ment obtenir plus des méta-grammaires. In *Proceedings of TALN'05*, Dourdan, France, June.

Elsa Tolone and Benoît Sagot. 2011. Using Lexicon-Grammar tables for French verbs in a large-coverage parser. In Zygmunt Vetulani, editor, *Human Language Technology, Forth Language and Technology Conference, LTC 2009, Poznań, Poland, November 2009, Revised Selected Papers*, Lecture Notes in Artificial Intelligence (LNAI). Springer Verlag.

Elsa Tolone, Éric Villemonte de La Clergerie, and Benoît Sagot. 2012. Evaluating and improving syntactic lexica by plugging them within a parser. In *Proceedings of the 8th Language Resources and Evaluation Conference (LREC'12)*, Istanbul, Turkey. To appear.

Karel van den Eynde and Piet Mertens. 2006. Valency dictionary - DICOVALENCY: user's guide. See http://bach.arts.kuleuven.be/dicovalence/.

Yannick Versley and Ines Rehbein. 2009. Scalable discriminative parsing for German. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 134–137, Stroudsburg, PA, USA.

Harmonization and Merging of two Italian Dependency Treebanks

Cristina Bosco*, Simonetta Montemagni◇, Maria Simi†

* Università di Torino, ◇ Istituto di Linguistica Computazionale "Antonio Zampolli" (ILC–CNR) - Pisa, † Università di Pisa
bosco@di.unito.it, simonetta.montemagni@ilc.cnr.it, simi@unipi.it

Abstract

The paper describes the methodology which is currently being defined for the construction of a "Merged Italian Dependency Treebank" (MIDT) starting from already existing resources. In particular, it reports the results of a case study carried out on two available dependency treebanks, i.e. TUT and ISST–TANL. The issues raised during the comparison of the annotation schemes underlying the two treebanks are discussed and investigated with a particular emphasis on the definition of a set of linguistic categories to be used as a "bridge" between the specific schemes. As an encoding format, the CoNLL de facto standard is used.

Keywords: Syntactic Annotation, Merging of Resources, Dependency Parsing

1. Introduction

Italian is featured by the availability of four dependency treebanks. Three of them were developed by national research institutions: the Turin University Treebank (TUT)[1] developed by the NLP group of the University of Turin (Bosco et al., 2000); the treebank called ISST–TANL, which was developed as a joint effort by the Istituto di Linguistica Computazionale (ILC–CNR) and the University of Pisa and originating from the Italian Syntactic–Semantic Treebank or ISST (Montemagni et al., 2003); the Venice Italian Treebank (VIT) developed by the University Ca' Foscari of Venice (Tonelli et al., 2008). A further Italian dependency treebank was developed in the framework of an international project, the Copenhagen Dependency Treebank (Buch-Kromann et al., 2009). Interesting to note, each of these resources, independently developed applying different dependency-based annotation schemes, has a quite limited size, ranging from around 94,000 tokens of TUT to about 60,000 tokens of the Italian CDT section.

In spite of their limited size, some of these resources have successfully been used for training and/or evaluation of dependency parsing systems. For instance, TUT was repeatedly used within the parsing task of the EVALITA evaluation campaign[2] in 2007, 2009 and 2011, for both training and testing dependency parsing systems. A previous version of ISST–TANL, namely ISST–CoNLL, was used for the CoNLL-2007 Shared Task on multilingual dependency parsing as far as Italian is concerned (Nivre et al., 2007; Montemagni and Simi, 2007). ISST–TANL was used in EVALITA 2009 and 2011 for two different tasks, syntactic parsing (Bosco et al., 2009) and domain adaptation (Dell'Orletta et al., 2012) respectively, and is currently being used in the SPLeT 2012 Shared Task on Dependency Parsing of Legal Texts[3].

Despite the encouraging results achieved with these treebanks in the above mentioned initiatives, we are aware that the relatively small size of these resources makes them usable in a restricted variety of tasks with an impact on the reliability of achieved results. By contrast, the availability of a larger resource, harmonizing and merging the original annotated resources, should result in crucial advancements for the Italian NLP.

Preliminary steps in this direction were performed for two of the above mentioned treebanks, namely TUT and ISST–TANL. The first step was represented by the exploitation of these resources in the framework of international evaluation campaigns (CoNLL and EVALITA) which required as a necessary prerequisite the conversion of the native annotation formats into the CoNLL representation standard. A further step was performed in the framework of EVALITA 2009 which included a dependency parsing track (Bosco et al., 2009) articulated into two subtasks differing at the level of used treebanks: TUT was used as the development set in the Main Subtask, and ISST–TANL represented the development set for the Pilot Subtask. The analysis of the results of the best scoring systems, in line with the state of the art dependency parsing technology for Italian, provided the opportunity to start investigating the influence of the design of both treebanks by testing these parsers on a common set of data annotated in both annotation schemes (Bosco et al., 2010). The last and still ongoing step is represented by the national project "Portal for the Access to the Linguistic Resources for Italian" (PARLI), involving several academic NLP groups. PARLI aims at monitoring and coordinating the activities of Italian NLP for fostering the development of new resources and tools that can operate together, and the harmonization of existing ones. The activities carried out within PARLI also comprise the annotation of a new corpus including the full text of the *Costituzione Italiana* [4] by the Pisa and Turin University groups within which the harmonization issue between the TUT and ISST–TANL annotations schemes started to be tackled.

In this paper we describe the methodology we are currently defining for the construction of a "Merged Italian Dependency Treebank" (MIDT) resulting from the harmonization and merging of existing Italian dependency treebanks. This methodology is being tested on the TUT and ISST–TANL treebanks. However, in the near future we would like to ex-

[1] http://www.di.unito.it/~tutreeb

[2] http://www.evalita.it/

[3] http://poesix1.ilc.cnr.it/splet_shared_task/

[4] http://parli.di.unito.it/activities_en.html

tend this methodology to also cover the other two available Italian dependency treebanks, i.e. VIT and Italian CDT. The paper is organised as follows: after illustrating (Section 2.) the main tenets of our approach to merging, Sections 3. and 4. provide a comparative analysis of the TUT and ISST–TANL annotation schemes, and of the performance of state–of–the–art dependency parsers trained on the two resources. Finally, Section 5. describes the construction of the merged resource and the parsing results achieved by using it as training data.

2. Our approach to merging

Since the early 1990s, different initiatives have been devoted to the definition of standards for the linguistic annotation of corpora with a specific view to re–using and merging existing annotated resources. A first attempt was represented by the outcome of the EAGLES (Expert Advisory Groups on Language Engineering Standards) initiative, in particular of the group of 'experts' set to work on the syntactic annotation of corpora who ended up with providing provisional standard guidelines (Leech et al., 1996). Whereas this first attempt operated at the level of both content (i.e. the linguistic categories) and encoding format, further initiatives tried to tackle these two aspects separately. This is the case, for instance, of LAF/GrAF (Ide and Romary, 2006; Ide and Suderman, 2007) and SynAF (Declerck, 2008), which represent on–going ISO TC37/SC4 standardization activities[5] dealing respectively with a generic meta–model for linguistic annotation and with a meta–model for syntactic annotation, including dependency structures. In both cases, the proposed framework for representing linguistic annotations is intended to be a pivot format capable of representing diverse annotation types of varying complexity which does not provide specifications for annotation content categories (i.e., the labels describing the associated linguistic phenomena), for which standardization appeared since the beginning to be a much trickier matter.

For what concerns the content categories, both architectures include a data category registry containing a (possibly hierarchical) list of data categories meant to represent a point of reference for particular tagsets used for the syntactic annotation of various languages, also in the context of various application scenarios. More recently, this issue is being handled by other standardization efforts such as ISOCat (Kemps-Snijders et al., 2009). ISOCat is intended to provide a set of data categories at various levels of granularity, each accompanied by a precise definition of its linguistic meaning. Labels applied in a user–defined annotation scheme should be mapped to these categories in order to ensure semantic consistency among annotations of the same phenomenon.

The work illustrated in this paper is concerned with the harmonization and merging of dependency–annotated corpora, with a particular emphasis on data categories. As an encoding format, we use the CoNLL representation format, which nowadays represents a *de facto* standard within the parsing community. As far as linguistic categories are concerned, we are not trying to create a single unified annotation scheme to be used by all Italian dependency treebanks: in line with the approaches sketched above, we believe that this represents an impractical and unrealistic task. To put it in other words, it is not a matter about one scheme being right and the other being wrong: we start from the assumption that all schemes are linguistically well–motivated and that there is no objective criterion for deciding which annotation scheme provides the most empirically adequate analysis of the texts. Rather, the challenge we are tackling in this paper, which to our knowledge still represents an open issue in the literature, is to find a way of translating between different annotation schemes and merging them, with the final aim of pooling costly treebank resources. This is being carried out by trying to define a set of linguistic categories to be used as a "bridge" between the specific schemes. This initial effort focused on the TUT and ISST–TANL resources, and in particular on the dependency annotation level, with the long term goal of involving in this process the other available dependency–based Italian treebanks. MIDT, i.e. "Merged Italian Dependency Treebank" represents the final result of the merging process being described in this paper. In order to achieve this goal, we proceeded through the following steps:

- analysis of similarities and differences of considered dependency annotation schemes;
- analysis of the performance of state of the art dependency parsers trained on both treebanks;
- mapping of the individual annotation schemes onto a set of shared (often underspecified) set of data categories;
- last but not least, parametrization of the annotation of the merged resources (still ongoing).

In what follows these different steps are described in detail.

3. The TUT and ISST–TANL treebanks

The TUT and ISST–TANL resources differ under different respects, at the level of both corpus composition and adopted representations.

For what concerns size and composition, TUT currently includes 3,452 Italian sentences (i.e. 102,150 tokens in TUT native, and 93,987 in CoNLL[6]) representative of five different text genres (newspapers, Italian Civil Law Code, JRC-Acquis Corpus[7], Wikipedia and the *Costituzione Italiana*). ISST–TANL includes instead 3,109 sentences (71,285 tokens in CoNLL format), which were extracted from the "balanced" ISST partition (Montemagni et al., 2003) exemplifying general language usage and consisting of articles from newspapers and periodicals, selected to cover a high variety of topics (politics, economy, culture, science, health, sport, leisure, etc.).

As far as the annotation scheme is concerned, TUT applies the major principles of the dependency grammar (Hudson,

[5] http://www.tc37sc4.org/

[6] In the following we will refer only to number of tokens in CoNLL format.

[7] http://langtech.jrc.it/JRC-Acquis.html

1984) using a rich set of grammatical relations, but it includes null elements to deal with non-projective structures, long distance dependencies, equi phenomena, pro drop and elliptical structures[8]. The ISST–TANL annotation scheme originates from FAME (Lenci et al., 2008), an annotation scheme which was developed starting from de facto standards and which was specifically conceived for complying with the basic requirements of parsing evaluation, and – later – for the annotation of unrestricted Italian texts.

3.1. Comparing the annotation schemes

The TUT and ISST–TANL annotation schemes are both dependency-based and therefore fall within the same broader family of annotation schemes. In spite of this fact there are significant differences which make the harmonization and merging of the two resources quite a challenging task. To put it in other words, if on the one hand there is a core of syntactic constructions for which the analysis given by different annotation schemes agree in all important respects, on the other hand there are also important differences concerning the inventory of dependency types and their linguistic interpretation, head selection criteria, the projectivity constraint as well as with respect to the analysis of specific syntactic constructions. In what follows we summarize the main dimensions of variation with a specific view to the merging issues.

Head selection criteria

Criteria for distinguishing the head and the dependent within dependency relations have been widely discussed in the linguistic literature, not only in the dependency grammar tradition, but also within other frameworks where the notion of syntactic head plays an important role. Unfortunately, different criteria have been proposed, some syntactic and some semantic, which do not lead to a single coherent notion of dependency (Kübler et al., 2009). Head selection thus represents an important and unavoidable dimension of variation between the TUT and ISST–TANL schemes, especially for what concerns constructions involving grammatical function words with respect to which there is no general consensus in the tradition of dependency grammar as to what should be regarded as the head and what should be regarded as the dependent. Let us focus on the following tricky cases: namely, the determiner–noun relation within nominal groups, the preposition–noun relation within prepositional phrases, the complementizer–verb relation in subordinate clauses as well as the auxiliary–main verb relation in complex verbal groups.

TUT always assigns heads on the basis of syntactic criteria, i.e. in all constructions involving one function word and one content word (e.g. determiner–noun, preposition–noun, complementizer–verb) the head role is always played by the function word. The only exception is represented by auxiliary–main verb constructions where the head role is played by the main verb. By contrast, in ISST–TANL head selection follows from a combination of syntactic and semantic criteria: i.e. whereas in the determiner–noun and auxiliary–verb constructions the head role is assigned to the semantic head (noun/verb), in preposition–noun and complementizer–verb constructions the head role is played by the element which is subcategorized for by the governing head, i.e. the preposition and the complementizer.

Note that this different strategy in at the level of head selection explains the asymmetric treatment of determiner–noun constructions with respect to preposition–noun ones in ISST–TANL and the fact that for TUT the same dependency type is used for both cases (see below).

Granularity and inventory of dependency types

TUT and ISST–TANL annotation schemes assume different inventories of dependency types characterized by different degrees of granularity in the representation of specific relations. The different degree of granularity of the annotation schemes is testified by the size of the adopted dependency tagsets, including 72 dependency types in the case of TUT and 29 in the case of ISST–TANL. Interestingly however, it is not always the case that the finer grained annotation scheme – i.e. TUT – is the one providing more granular distinctions: whereas this is typically the case, there are also cases in which more granular distinction are adopted in the ISST-TANL annotation scheme. In what follows, we provide examples of both cases.

Consider first TUT relational distinctions which are neutralized at the level of ISST–TANL annotation. A difference in terms of granularity refers e.g. to the annotation of appositive (or unrestrictive) modifiers, which in TUT are annotated by resorting to a specific relation (`APPOSITION`), and which in ISST–TANL are not distinguished from other kinds of modifiers (`mod`). Similarly, TUT partitions predicative complements into two classes, i.e. subject and object predicative complements (`PREDCOMPL+SUBJ` and `PREDCOMPL+OBJ` respectively) depending on whether the complement refers to the subject or the object of the head verb, whereas in ISST–TANL the same dependency type (`pred`) is used to annotate both cases.

Let us consider now the reverse case, i.e. in which ISST–TANL adopts finer–grained distinctions with respect to TUT: for instance, ISST–TANL envisages two different relation types for determiner–noun and preposition–noun constructions (`det` and `prep` respectively), whereas TUT represents both cases in terms of the same relation type (`ARG`). This latter example follows from another important dimension of variation between the two schemes, concerning head selection (see above).

Another interesting and more complex example can be found for what concerns the partitioning of the space of prepositional complements, be they modifiers or subcategorized arguments. TUT distinguishes between `MODIFIER`(s) on the one hand and subcategorised arguments on the other hand; the latter are further distinguished between indirect objects (`INDOBJ`) and all other types of indirect complements (`INDCOMPL`). ISST–TANL neutralizes such a distinction by resorting to a single dependency type, i.e. `comp` (mnemonic for complement), for all relations holding between a head and a prepositional complement, whether a modifier or a subcategorized argument. On the other hand, `comp`(lements) are further

[8] CoNLL format does not include null elements, but the projectivity constraint is maintained at the cost of a loss of information with respect to native TUT (in some cases).

subdivided into semantically oriented categories, such as temporal, locative or indirect complements (comp_temp, comp_loc and comp_ind).

Same dependency type, different annotation criteria

Even when the two schemes show common dependency types, they can diverge at the level of their interpretation, and thus of the underlying annotation criteria. This is the case, for instance, of the "object" relation which in the TUT annotation scheme refers to the direct argument (either in the nominal or clausal form) occurring at least and most once and expressing the subcategorized object, and which in ISST–TANL is meant to denote the relation holding between a verbal head and its non–clausal direct object (other dependency types are used for clausal complements).

Another interesting example is represented by relative clauses. TUT and ISST–TANL follow the same strategy in the representation of standard relative clauses, according to which the head of the relative clause is the verb and the relative pronoun is governed by it as a standard argument. The verbal head is then connected to the antecedent noun through a specific relation, RELCL in TUT and mod_rel in ISST–TANL. However, TUT also treates so–called reduced relative clauses, i.e. constructions where there is no overt relative pronoun and the verb appears in the participial form (either present or past participle), in the same way; namely, by using the same relation type to link the verb of the reduced relative clause to the governing noun. In ISST–TANL, constructions without overt relative pronouns are instead represented by resorting to a general modifier relation (mod).

Projectivity of dependency representations

Projectivity is an important constraint in dependency grammar, relating dependency structures to linear realizations. If on the one hand most NLP systems for dependency parsing assume projectivity, on the other hand this is not the case on the linguistic side where non–projective representations are resorted to for dealing with specific linguistic constructions (e.g. long-distance dependencies) mainly occurring in flexible word order languages (such as Italian). Whereas ISST–TANL corpus allows for non–projective representations, TUT assumes the projectivity constraint.

Treatment of specific constructions

Further important differences between TUT and ISST–TANL annotation schemes are concerned with the treatment of coordination and punctuation, phenomena which are particularly problematic to deal with in the dependency framework.

Besides the general issue widely discussed in the literature of whether coordination can be analyzed in terms of binary asymmetrical relations holding between a head and a dependent, there are different ways put forward to deal with it. In both TUT and ISST–TANL resources, coordinate constructions are considered as asymmetric structures with a main difference: while in ISST–TANL the conjunction and the subsequent conjuncts are all linked to the first conjunct, in TUT the conjuncts starting from the second one are linked to the immediately preceding conjunction.

Also the treatment of punctuation is quite problematic in the framework of a dependency annotation scheme, although this has not been specifically dealt with in the linguistic literature. Both TUT and ISST–TANL schemes cover punctuation with main differences holding at the level of both dependency types and head selection criteria. Whereas ISST–TANL has just one dependency type for all punctuation tokens, TUT distinguishes different dependency types depending on the involved punctuation token and syntactic construction. For example, in TUT an explicit notion of parenthetical is marked while in ISST–TANL it is not. Significant differences also lie at the level of the head assignment criteria: in TUT the head of the punctuation tokens in the parenthetical structure coincides with the governing head of the sub–tree covering the parenthetical structure (i.e. it is external to the parenthetical structure), whereas in ISST–TANL the paired punctuation marks of the parenthetical structure are both connected to the head of the delimited phrase (i.e. internally to the parenthetical). Other important differences holding between TUT and ISST–TANL schemes are concerned with sentence splitting, tokenization and morpho–syntactic annotation, all aspects which represent important prerequisites for the merging of dependency annotations. All these issues have been addressed and a solution has been proposed as part of the whole harmonization and merging process.[9] In this paper, however, we won't further discuss these aspects and we will focus on the merging of dependency annotations.

4. TUT and ISST–TANL as training corpora

In Bosco et al. (2010), a dependency–based analysis of the performance of state of the art parsers participating in EVALITA 2009 (two stochastic parsers and a rule–based one) with respect to a shared test set was reported, with the final aim of assessing the impact of annotation schemes on parsing results. In particular, for each relation in the TUT and ISST–TANL dependency annotation schemes, the performance of the three parsers was analyzed in terms of Precision (P), Recall (R) and related f-score. In order to identify problematic areas of parsing, both TUT and ISST–TANL dependency–relations were partitioned into three classes (i.e. low-, medium- and best-scored dependency relations) with respect to the associated f-score, which was taken to reflect their parsing difficulty (for more details see Bosco et al. (2010)). Achieved results showed that the improvement of parsing technology should proceed hand in hand with the development of more suitable representations for annotated syntactic data. In this paper we are dealing with the latter issue: we believe that the results of this comparative analysis should also be taken into account in the definition of the merging methodology.

Similar trends were observed in the performance of parsers against TUT and ISST–TANL. First, in both cases hard to parse relations include "semantically loaded" relations such as comp_temp, comp_loc and comp_ind for ISST–

[9] The interested reader is referred to the following URL for more details on the merging of TUT and ISST–TANL morpho–syntactic annotations: http://medialab.di.unipi.it/wiki/POS_and_morphology.

TANL and APPOSITION and INDOBJ for TUT. Moreover, relations involving punctuation appeared to be difficult to parse for statistical parsers in the case of TUT, whereas the rule–based parser had problems dealing with coordinate structures in ISST–TANL; it should be noted however that ISST–TANL con/conj relations show values very close to the low threshold value also in the case of the stochastic parsers. This contrastive analysis thus confirmed a widely acknowledged claim, i.e. that coordination and punctuation phenomena still represent particularly challenging areas for parsing (Cheung and Penn, 2009). The problems raised by the analysis of "semantically loaded" relations in the case of both treebanks suggest that the parsers do not appear to have sufficient evidence to deal reliably with them; in principle, the solutions to the problem range from increasing the size of the training corpus, to neutralising their distinction at this annotation level and postponing their treatment to further processing levels. Concerning the best scored relations, it came out that in both cases they mainly refer to "local" relations. Interesting to note, there is a significant overlapping between the two sets: e.g. the TUT ARG and the ISST–TANL det/prep together have the same coverage; the same holds for the TUT AUX+PASSIVE/ AUX+TENSE relations with respect to the ISST–TANL aux relation.

5. Merging TUT and ISST–TANL

In this section we summarise the work done towards merging the two annotated resources, by defining a bridge annotation scheme to be used as an interlingua for converting the individual treebanks and combining them into a wider resource. Whereas we are aware of previous efforts of combining different annotation types (e.g. ISOTimeML, PropBank, and FrameNet annotations as reported in Ide and Bunt (2010)) as well as dependency structures of different languages (e.g. English vs Japanese as discussed in Hayashi et al. (2010)), to our knowledge this represents the first merging effort carried out with respect to different dependency annotation schemes defined for the same language: we might look at them as dependency annotation "dialects". In what follows, we first illustrate the criteria which guided the definition of a bridge annotation scheme to be used for merging the two resources (Section 5.1.); second, in order to test the adequacy of the resulting annotation scheme as far as dependency parsing is concerned we report the parsing results achieved so far by exploiting the MIDT resources as training data (Section 5.2.).

5.1. Defining a bridge annotation scheme for MIDT

The results of the comparative analysis detailed in section 3.1. are summarized in columns 2, 3 and 4 of Table 1, where for each relation type in a given scheme the corresponding relation(s) are provided as far as the other scheme is concerned. The fourth column (headed "DIFF") provides additional information for what concerns the type of correspondence holding between ISST–TANL and TUT dependency categories: two different values are foreseen, which can also be combined together, corresponding to whether the correspondence involves different head selection criteria ("Hsel") and/or a different linguistic interpretation resulting in a different coverage ("covg"). It can be noted that the emerging situation is quite heterogeneous.

The only simple cases are represented by a) the root, relative clause and passive subject cases for which we observe a 1:1 mapping, and b) the relation(s) involving auxiliaries in complex tense constructions characterized by a 1:n mapping. As far as b) is concerned, in principle the TUT relation distinctions might be recoved by also taking into account the lexical and morpho–syntactic features associated with the involved auxiliary and main verbal tokens. In both a) and b) cases, however, the identification of a bridge category to be used for merging purposes does not appear to be problematic at all (see below).

A slightly more complex case is represented by the determiner–noun, preposition–noun and complementizer–verb relations whose treatment in the two schemes is different both at the level of involved relations and head selection criteria. For these cases, the merging process should also be able to deal with the "external" consequences at the level of the overall tree structure as far as the attachment of these constructions is concerned. For instance, depending on the scheme in a sentence like *I read the book* the object of reading would be either the article (TUT) or the noun (ISST–TANL). In these cases, besides defining a semantically coherent bridge category compatible with both TUT and ISST–TANL annotations, the conversion process is not circumscribed to the dependency being converted but should also deal with the restructuring of the sub–tree whose head governs the dependency head.

Most part of remaining dependency relations involve different, sometimes orthogonal, sets of criteria for their assignment and are therefore more difficult to deal with for merging purposes. Consider, as an example, the direct object relation, already discussed in Section 3.1.: in ISST–TANL the relation obj is restricted to non–clausal objects, whereas the TUT OBJ relation also includes clausal ones. This difference in terms of coverage follows from the fact that whereas TUT implements a pure dependency annotation where the dependency type does not vary depending on the complement type (e.g. clausal vs nominal objects), in ISST–TANL all clausal complements are treated under a specific relation type, named arg. This represents a much trickier case to deal with for merging purposes: here it is not a matter of choosing between two different representation strategies, but rather of converging on a possibly underspecified representation type which could be automatically reconstructed from both TUT and ISST–TANL resources. If on the one hand in TUT it is possible to recover the ISST–TANL notion of arg by exploiting the morpho–syntactic features of the tokens involved in the relation, on the other hand it is impossible to automatically recover the TUT notion of OBJ starting from ISST–TANL annotation only (in this case information about the subcategorization properties of individual verbs would be needed).

Another problematic conversion area is concerned with the representation of deverbal nouns (e.g. *destruction*) whose annotation in TUT is carried out in terms of the underlying predicate–argument structure (i.e. by marking relations such as subject, object, etc.) whereas in ISST–TANL is marked by resorting to generic surface (e.g. comp(lement))

relations. As in the subordination case, the only possible solution here is to converge on a representation type which can be automatically reconstructed from both TUT and ISST–TANL resources by combining morfo–syntactic and dependency information.

It should also be noted that there are semantically–oriented distinctions which are part of the ISST–TANL annotation scheme (e.g. temporal and locative modifiers, i.e. `mod_temp` vs `mod_loc`) but which do not find a counterpart in the CoNLL version of the TUT treebank. In this case the only possible solution consists in neutralizing such a distinction at the level of the MIDT representation.

The conversion process had also to deal with cases for which the difference was only at the level of annotation criteria rather than of the dependency types. Consider for instance the treatment of coordination phenomena. Both TUT and ISST–TANL foresee two different relations, one for linking the conjunction to one of the conjuncts (i.e. the ISST–TANL `con` and the TUT `COORD` relations) and the other one for connecting the conjoned elements (i.e. the ISST–TANL `conj` and the TUT `COORD2ND` relations). In spite of this parallelism at the tagset level, the strategy adopted for representing coordinate structures is different in the two resources: whereas ISST–TANL takes the first conjunct as the head of the whole coordinate structure and all subsequent conjoined elements and conjunctions are attached to it, in TUT both the conjuction and the conjunct are governed by the element immediately preceding it. In this case, the conversion towards MIDT consists in restructuring the internal structure of the coordinate structure.

So far, we focused on the conversion of "canonical" dependency relations and of coordination: the treatment of punctuation is still being defined. For each set of corresponding ISST–TANL and TUT categories, the last column of Table 1 contains the MIDT counterpart. The definition of the MIDT dependency tagset was first guided by practical considerations: namely, bridge categories should be automatically reconstructed by exploiting morpho–syntactic and dependency information contained in the original ISST–TANL and TUT resources. In MIDT, we also decided to neutralize semantically–oriented distinctions (such as the subject of passive constructions, or the indirect object) which turned out to be problematic (see Section 4.) to be reliably identified in parsing in spite of their being explicitly encoded in both annotation schemes. Last but not least, the linguistic soundness of resulting categories was also assessed, by comparing the MIDT tagset with de facto dependency annotation standards: among them it is worth mentioning here the annotation tagsets proposed by the syntactic annotation initiatives like TIGER, ISST, Sparkle and EAGLES as reported in Declerck (2008) or the most recent Stanford typed dependencies representation (de Marneffe and Manning, 2008).

It should be noted that, in some cases, MIDT provides two different options, corresponding to the TUT and ISST–TANL styles for dealing with the same construction: this is the case of determiner–noun, preposition–noun, complementizer–verb and auxiliary–main verb relations whose MIDT representation is parameterizable: for the time being only one possible option has been activated. The final MIDT tagset contains 21 dependency tags (as opposed to the 72 tags of TUT and the 29 of ISST–TANL), including the different options provided for the same type of construction. The question at this point is whether the MIDT annotation scheme is informative enough and at the same time fully predictable to reliably be used for different purposes: in the following section a first preliminary answer to this question is provided.

5.2. Using MIDT as training corpus

In this section we report the results achieved by using MIDT resources for training a dependency parsing system. We used DeSR (Dependency Shift Reduce), a transition–based statistical parser (Attardi, 2006) which builds dependency trees while scanning a sentence and applying at each step a proper parsing action selected through a classifier based on a set of representative features of the current parse state. Parsing is performed bottom-up in a classical Shift/Reduce style, except that the parsing rules are special and allow parsing to be performed deterministically in a single pass. It is possible to specify, through a configuration file, the set of features to use (e.g. POS tag, lemma, morphological features) and the classification algorithm (e.g. Multi-Layer Perceptron (Attardi and Dell'Orletta, 2009), Support Vector Machine, Maximum Entropy). In addition, the parser can be configured to run either in left–to–right or right–to–left word order. An effective use of DeSR is the Reverse Revision parser (Attardi et al., 2009), a stacked parser which first runs in one direction, and then extracts hints from its output to feed another parser running in the opposite direction. All these options allow creating a number of different parser variants, all based on the same basic parsing algorithm. Further improvement can then be achieved by the technique of parser combination (Attardi et al., 2009), using a greedy algorithm, which preserves the linear complexity of the individual parsers and often outperforms other more complex algorithms.

Let us start from the results achieved by this parser in the framework of the evaluation campaign Evalita 2011 with the original TUT and ISST–TANL datasets distributed in the framework of the "Dependency Parsing" (Bosco and Mazzei, 2012) and "Domain Adaptation" (Dell'Orletta et al., 2012) tracks respectively. Table 2 reports, in the first two rows, the values of Labeled Attachment Score (LAS) obtained with respect to the ISST–TANL and TUT datasets with the technique of parser combination: 82.09% vs 89.88%. This result is in line with what reported in Bosco et al. (2010), where a similar difference in performance was observed with respect to the TUT and ISST–TANL test sets: the composition of the training corpora and the adopted annotation schemes were identified as possible causes for such a difference in performance.

The results reported in rows 3–6 have been obtained by training DeSR with the MIDT version of the TUT and ISST–TANL individual resources, whereas rows 7 and 8 refer to the merged MIDT resource. In all these cases two different LAS scores are reported, i.e. the overall score and the one computed by excluding punctuation: this was done to guarantee the comparability of results since, as pointed out above, the conversion of punctuation is still under way

Table 1: ISST–TANL, TUT and MIDT linguistic ontologies

ID	ISST–TANL	TUT	DIFF	MIDT
1	ROOT	TOP		_ROOT
2	arg	no equivalent relation (see 5, 21)	covg	_ARG
3	aux	AUX(+PASSIVE +PROGRESSIVE +TENSE)		_AUX
4	clit	EMPTYCOMPL SUBJ/SUBJ+IMPERS		_CLIT
5	comp	INDCOMPL SUBJ/INDCOMPL COORD+COMPAR	covg	_COMP
6	comp_ind	INDOBJ SUBJ/INDOBJ		_COMP
7	comp_loc	no equivalent relation(see 5)	covg	_COMP
8	comp_temp	no equivalent relation (see 5)	covg	_COMP
9	con	COORD(+BASE +ADVERS +COMPAR +COND +CORRELAT +ESPLIC +RANGE +SYMMETRIC)	covg Hsel	_COORD
10	concat	CONTIN(+LOCUT +DENOM +PREP)		_CONCAT
11	conj	COORD2ND(+BASE +ADVERS +COMPAR +COND +CORRELAT +ESPLIC) COORDANTEC+CORRELAT	covg Hsel	_COOR2ND
12	det	ARG	Hsel	_DET, _ARG
13	dis	no equivalent relation (see 9)	covg	_COORD
14	disj	no equivalent relation (see 11)	covg	_COOR2ND
15	mod	APPOSITION RMOD RMOD+RELCL+REDUC INTERJECTION COORDANTEC+COMPAR	covg	_MOD
16	mod_loc	no equivalent relation (see 15)	covg	_MOD
17	mod_rel	RMOD+RELCL		_RELCL
18	mod_temp	no equivalent relation (see 15)	covg	_MOD
19	modal	no equivalent relation (see 3)	Hsel covg	_AUX
20	neg	no equivalent relation (see 15)	covg	_NEG
21	obj	OBJ SUBJ/OBJ EXTRAOBJ	covg	_OBJ
22	pred	PREDCOMPL(+SUBJ +OBJ) RMODPRED(+OBJ +SUBJ)		_PRED
23	pred_loc	no equivalent relation (see 22)	covg	_PRED
24	pred_temp	no equivalent relation (see 22)	covg	_PRED
25	prep	ARG	Hsel	_PREP, _ARG
26	punc	CLOSE(+PARENTHETICAL +QUOTES) END INITIATOR OPEN(+PARENTHETICAL +QUOTES) SEPARATOR		_PUNC
27	sub	ARG	Hsel	_SUB, _ARG
28	subj	SUBJ EXTRASUBJ	covg	_SUBJ
29	subj_pass	OBJ/SUBJ		_SUBJ

(i.e. for the time being the original treatment of punctuation is maintained). For the MIDT resources, the DeSR results achieved with the best single parser and with the combination of parsers are reported. It can be noticed that in both cases an improvement is observed with respect to the native TUT and ISST–TANL resources, +0.23% and + 2.90% respectively. The last two rows refer to the results achieved with the merged resource used as training, which at the time of writing is far from being completed due to the fact that the treatment of punctuation has not been unified yet. In spite of this fact (which in principle could generate noise in the model), the performance achieved by training the parser on the merged resource is still high, although lower than the result achieved with TUT_MIDT_train. The parsing model trained on the merged resource obtains the following results with respect to individual test sets: 83.43% for ISST–TANL_MIDT_test and 88.03% for TUT_MIDT_test, which represent slighly lower LAS scores than those obtained by using as training the corresponding resource. In spite of the fact that the harmonization and merging of the two resources is still under way, achieved parsing results show that the resulting MIDT resource can effectively be used for training dependency parsers.

6. Conclusion

The outcome of the effort sketched in this paper is three–fold. First, a methodology for harmonizing and merging annotation schemes belonging to the same family has been defined starting from a comparative analysis carried out a) with respect to different dimensions of variation ranging from head selection criteria, dependency tagset granularity to annotation guidelines or the treatment of specific constructions, and b) by analysing the performance of state–of–the–art dependency parsers using as training the original resources. Second, Italian will have a bigger treebank, which will be further extended if other available treebanks will be involved in the merging process. Third, but not least important, the set of "bridge" categories which have been defined for merging purposes can in principle be used to enrich the set of dependency–related data categories of the ISOcat Data Category Registry, thus enabling other merging initiatives operating within the same dependency–based family of annotation schemes to start from a richer and already experimented set of basic dependency–related categories. Current directions of work include: the completion of the conversion and merging process to obtain a fully harmonised resource; the parameterizability of conversion, in order to allow for different annotation choices.

Table 2: Parsing results with native vs MIDT resources

TRAINING	TEST	PARSER	LAS	LAS no punct
ISST–TANL_native_train	ISST–TANL_native_test	Parser comb.	82.09%	not available
TUT_native_train	TUT_native_test	Parser comb.	89.88%	not available
ISST–TANL_MIDT_train	ISST–TANL_MIDT_test	Best single	84.47%	86.15%
ISST–TANL_MIDT_train	ISST–TANL_MIDT_test	Parser comb.	84.99%	86.78%
TUT_MIDT_train	TUT_MIDT_test	Best single	89.23%	90.74%
TUT_MIDT_train	TUT_MIDT_test	Parser comb.	90.11%	91.58%
merged_MIDT_train	merged_MIDT_test	Best single	86.09%	88.60%
merged_MIDT_train	merged_MIDT_test	Parser comb	86.66%	89.04%

7. Acknowledgements

The work reported in this paper was partly carried out in the framework of the national PRIN project PARLI ("Portal for the Access to the Linguistic Resources for Italian") funded by the Ministry of Public Instruction and University.

8. References

G. Attardi and F. Dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In *Proceedings of NAACL HLT (2009)*.

G. Attardi, F. Dell'Orletta, M. Simi, and J. Turian. 2009. Accurate dependency parsing with a stacked multilayer perceptron. In *Proceedings of EVALITA, Evaluation of NLP and Speech Tools for Italian*, Reggio Emilia, Italy.

G. Attardi. 2006. Experiments with a multilanguage non–projective dependency parser. In *Proceedings of CoNLL'06*, New York City, New York.

C. Bosco and A. Mazzei. 2012. The evalita 2011 parsing task: the dependency track. In *Working Notes of Evalita'11*, Roma, Italy.

C. Bosco, V. Lombardo, L. Lesmo, and D. Vassallo. 2000. Building a treebank for italian: a data-driven annotation schema. In *Proceedings of LREC'00*, Athens, Greece.

C. Bosco, S. Montemagni, A. Mazzei, V. Lombardo, F. Dell'Orletta, and A. Lenci. 2009. Evalita'09 parsing task: comparing dependency parsers and treebanks. In *Proceedings of Evalita'09*, Reggio Emilia, Italy.

C. Bosco, S. Montemagni, A. Mazzei, V. Lombardo, F. Dell'Orletta, A. Lenci, L. Lesmo, G. Attardi, M. Simi, A. Lavelli, J. Hall, J. Nilsson, and J. Nivre. 2010. Comparing the influence of different treebank annotations on dependency parsing. In *Proceedings of LREC'10*, Valletta, Malta.

M. Buch-Kromann, I. Korzen, and H. Høeg Müller. 2009. Uncovering the 'lost' structure of translations with parallel treebanks. *Special Issue of Copenhagen Studies of Language*, 38:199–224.

J.C.K. Cheung and G. Penn. 2009. Topological field parsing of German. In *Proceedings of ACL-IJCNLP'09*.

M.C. de Marneffe and C.D. Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

T. Declerck. 2008. A framework for standardized syntactic annotation. In *Proceedings of LREC'08*, Marrakech, Morocco.

F. Dell'Orletta, S. Marchi, S. Montemagni, G. Venturi, T. Agnoloni, and E. Francesconi. 2012. Domain adaptation for dependency parsing at evalita 2011. In *Working Notes of Evalita'11*, Roma, Italy.

Y. Hayashi, T. Declerck, and C. Narawa. 2010. Laf/graf-grounded representation of dependency structures. In *Proceedings of LREC'10*, Valletta, Malta.

R. Hudson. 1984. *Word Grammar*. Basil Blackwell, Oxford and New York.

N. Ide and H. Bunt. 2010. Anatomy of annotation schemes: mapping to graf. In *Proceedings of the Linguistic Annotation Workshop*, Uppsala, Sweden.

N. Ide and L. Romary. 2006. Representing linguistic corpora and their annotations. In *Proceedings of LREC'06*, Genova, Italy.

N. Ide and K. Suderman. 2007. GrAF: A graph-based format for linguistic annotations. In *Proceedings of the Linguistic Annotation Workshop*, Prague, Czech Republic.

M. Kemps-Snijders, M. Windhouwer, P. Wittenburg, and S.E. Wright. 2009. Isocat: remodelling metadata for language resources. *IJMSO*, 4(4):261–276.

S. Kübler, R.T. McDonald, and J. Nivre. 2009. *Dependency Parsing*. Morgan & Claypool Publishers, Oxford and New York.

G. Leech, R. Barnett, and P. Kahrel. 1996. Eagles recommendations for the syntactic annotation of corpora. Technical report, EAG-TCWG-SASG1.8.

A. Lenci, S. Montemagni, V. Pirrelli, and C. Soria. 2008. A syntactic meta–scheme for corpus annotation and parsing evaluation. In *Proceedings of LREC'00*, Athens, Greece.

S. Montemagni and M. Simi. 2007. The Italian dependency annotated corpus developed for the CoNLL–2007 shared task. Technical report, ILC–CNR.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Mana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In A. Abeillé, editor, *Building and Using syntactically annotated corpora*. Kluwer.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of EMNLP-CoNLL'07*.

S. Tonelli, R. Delmonte, and A. Bristot. 2008. Enriching the venice italian treebank with dependency and grammatical relations. In *Proceedings of LREC'08*, Marrakech, Morocco.

L-LEME: an Automatic Lexical Merger based on the LMF Standard

Riccardo Del Gratta[◇], Francesca Frontini[◇], Monica Monachini[◇],

Valeria Quochi[◇], Francesco Rubino[◇], Matteo Abrate[*] and Angelica Lo Duca[*]

[*]IIT-CNR, [◇]ILC-CNR
Consiglio Nazionale delle Ricerche
Via Moruzzi, 1 - Pisa, Italy
[*]name.surname@iit.cnr.it, [◇]name.surname@ilc.cnr.it

Abstract

The present paper describes LMF LExical MErger (L-LEME), an architecture to combine two lexicons in order to obtain new resource(s). L-LEME relies on standards, thus exploiting the benefits of the ISO Lexical Markup Framework (LMF) to ensure interoperability. L-LEME is meant to be *dynamic* and heavily adaptable: it allows the users to configure it to meet their specific needs. The L-LEME architecture is composed of two main modules: the *Mapper* , which takes in input two lexicons $\mathcal{A}$ and $\mathcal{B}$ and a set of user-defined rules and instructions to guide the mapping process (Directives D) and gives in output all matching entries. The algorithm also calculates a cosine similarity score. The *Builder* takes in input the previous results, a set of Directives D_1 and produces a new LMF lexicon $\mathcal{C}$. The Directives allow the user to define its own building rules and different merging scenarios. L-LEME is applied to a specific concrete task within the PANACEA project, namely the merging of two Italian SubCategorization Frame (SCF) lexicons. The experiment is interesting in that $\mathcal{A}$ and $\mathcal{B}$ have different philosophies behind, being $\mathcal{A}$ built by human introspection and $\mathcal{B}$ automatically extracted. Ultimately, L-LEME has interesting repercussions in many language technology applications.

Keywords: LMF Standard, (semi-)automatic Lexicon Merging, Similarity Score

1. Introduction

For most Natural Language Processing (NLP) applications the availability of suitable Language Resources represents a major bottleneck for market coverage and quality improvement. While it is true that quantity alone does not necessarily imply sufficient quality, esp. for commercial bodies, it still makes the difference. Applications such as Machine Translation (MT), for example, require a relatively large quantity of data in order to achieve good performances. Yet, there are many language resources developed by different groups, researchers and projects that, put together, could constitute a wealth of knowledge for such applications. Along with research and development of new and better methods for the creation of language resources, it is then fundamental to continue exploring how to merge, repurpose and reuse existing resources. Merging two lexical resources, especially with (semi-)automatic methods, is a challenging task, since different lexicons often have different structures, different granularity of information, and in the worst case, they may contain even partially overlapping entries. Most resource merging experiments and tasks have been (and are being) carried out manually. A few exceptions of (semi-)automatic experiments for merging resources exist (cf. section 2.): they are built ad hoc, so that the proposed methodologies can not be readily extended to other cases.

The present paper describes L-LEME, an architecture which exploits the ISO LMF lexicon representation standard in order to generalize the merging process. The strongest points of L-LEME are that (i) it relies on standards, thus ensuring interoperability and making mapping/merging tasks easier, and (ii) it is *dynamic* and heavily adaptable: it foresees a human-driven phase that allows the users to configure it to meet their specific needs. L-LEME is applied to a specific, concrete task within the PANACEA project[1], namely the merging of two SubCategorization Frame (SCF) lexicons for Italian.

The paper is organized as follows: in section 2. we briefly report some related works. In section 4. we describe the system architecture, while in section 3.1. we describe the Lexical Markup Framework. Sections 6. and 7. are dedicated to the mapping and building processes respectively. Finally, in section 9. we present the results obtained in the PANACEA experiment.

2. Related Work

The problem of merging two lexicons has already been tackled and reported in the literature in the past.

Most of the approaches described focus on particular resources and different specific interpretation of *merging* and seem to be rather ad hoc for the resources addressed, although more recent works investigate possibly general and (semi-)automatic methods.

Chan and Wu (Chan and Wu, 1999) describe a basic method for automatically generating a set of mapping rules between lexicons that employ different incompatible Part Of Speech (POS) categories. The strategy described is to use the co-occurrence of tags in common lemmas as a basis for automatically learning POS mapping rules. Once mappings are established, a merged resource can be obtained.

(Monachini et al., 2006) focuses on merging the phonological layer of the Parole-Simple-Clips (PSC) lexicon and

[1]PANACEA is an FP7 EU funded project for the creation of a platform for the automatic production of language resources. See www.panacea-lr.eu

the LCSTAR pronunciation lexicon. The mapping is performed automatically on the original entries converted into an LMF compliant Interchange Format via rules based on manually set equivalence parameters. The merging is then achieved through unification of the matching entries.
(Crouch and King, 2005) describe a more complex merging attempt. Their goal is to build a Unified Lexicon (UL) for verb entries linking their syntactic SubCategorization Frames. The lexical resources involved in the merging are: WordNet[2] for word senses and synonym sets, Cyc[3] for ontological concepts and VerbNet[4] for lexical semantics and syntactic information on verb behaviour. Merging is achieved stepwise in a semi-automatic manner.
LeXFlow (Tesconi et al., 2006) is a web application framework where LMF lexicons semiautomatically interact by reciprocally enriching themselves. It is thus intended as an instrument for the development of dynamic multi-source lexicons. In a way similar to the one implemented in a document workflow (Marchetti et al., 2005), lexical entries move across agents and are dynamically updated. In principle, agents can be either human or software.
(Molinero et al., 2009) describe a method for building a large morphological and syntactic lexicon[5] by merging existing resources via conversion to a common format. Morphological merging relies on lemmas which are common addressing possible conflicts and exceptions by giving priority to a baseline lexicon. Syntactic merging exploits the fully specified (syntactic) information and is achieved by comparing and matching common syntactic frames. (Padró et al., 2011) elaborate on the methods proposed by (Molinero et al., 2009) and (Crouch and King, 2005) and aim at a fully automatic merging process. Their work focuses on two syntactic lexicons for Spanish and adopts a graph unification-based approach representing entries as feature structures. Nevertheless, human intervention is needed for converting the native encoding into the required format.
All the reported experiments seem to rely on the assumption that a protocol for lexicons merging implies three logical phases: a) *pre-integration*, b) *mapping*, c) *building*.
The *pre-integration* phase makes the two lexicons comparable. The *mapping* phase, instead, identifies a relation among the entries of the two lexicons. The *building* phase, finally, combines the related entries of the two lexicons in order to form a new lexicon.

3. The Lexical Markup Framework

The Lexical Markup Framework (Francopoulo et al., 2008) defines an abstract meta-model for the construction/description of computational lexicons. LMF provides a common, shared representation of lexical objects that allows the encoding of rich linguistic information, including morphological, syntactic, and semantic aspects. LMF is organized in several different packages: each package is enriched by resources that are part of the definition of LMF. These resources include specific data categories used to adorn both the core model and its extensions.
Data categories can be seen as the main building blocks for the representation of the entries; they are the actual linguistic descriptors that are used to instantiate each entry.
In this paper we focus on the LMF syntax extension where the `Syntactic Behaviour` represents the basic building block and encodes one of the possible behaviours of a `Lexical Entry`. A detailed description of the syntactic behaviour of a lexical entry is further defined by the `Subcategorization Frame` object, which is the "heart" of the syntax module.
SubCategorization Frame is used to represent one syntactic configuration, in terms of `Syntactic Arguments`, and does not depend on individual syntactic units; rather it may be shared by different units. In other words SubCategorization Frames are verb independent and can be linked by Syntactic Behaviours of different verbs sharing the same argument structure. In addition to the argument structure represented by the SCF, the Syntactic Behaviour (SB) also encodes other syntactic properties of the entry, notably the auxiliary.

3.1. Lexicon Model

We can model an LMF lexicon as follows:

$$\mathcal{L} = \mathcal{L}_{morpho} \oplus \mathcal{L}_{syn} \oplus \mathcal{L}_{sem} \oplus \dots \quad (1)$$

Where $\mathcal{L}_{xyz}$ are separated, yet interconnected according to LMF and implement the classes presented in figure 1.

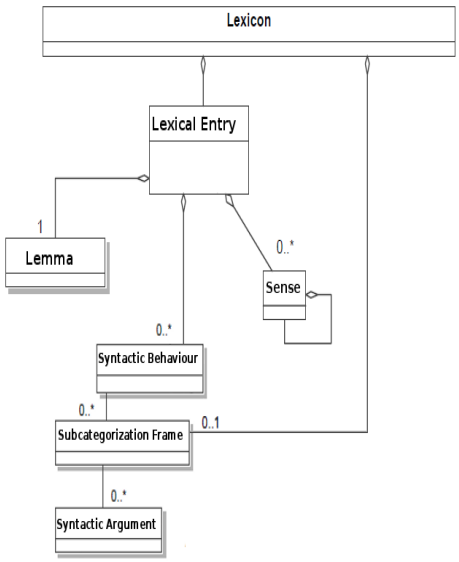

Figure 1: The excerpt LMF UML model used within this paper

The model (1) can be rewritten as follows:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{morpho} \oplus \mathcal{L}_{syn} \oplus \mathcal{L}_{sem} \oplus \dots \\ &= \mathcal{L}_{morpho} \cup Lnk(\mathcal{L}_{morpho}, \mathcal{L}_{syn}) \cup \mathcal{L}_{syn} \quad (2) \\ \cup\ & Lnk(\mathcal{L}_{morpho}, \mathcal{L}_{sem}) \cup \mathcal{L}_{sem} \cup \dots \end{aligned}$$

[2] http://wordnet.princeton.edu/
[3] http://cyc.com/cyc/opencyc/overview
[4] http://verbs.colorado.edu/ mpalmer/projects/verbnet.html
[5] The Leffe - Lexico de formas flexionadas del espanol.

Where $Lnk(x, y)$ are (LMF) classes that contain pointers or references to other classes.
In LMF-based subcategorization lexicons, the role of linker $Lnk(\mathcal{L}_{morpho}, \mathcal{L}_{syn})$ in model (2) is played by the `<SyntacticBehavior />` class that is hierarchically a *child* of `<LexicalEntry />` and contains an explicit *pointer* to the `<SubcategorizationFrame />` class. The LMF entry (1) reports an example.

```
<Lexicon>
 <LexicalEntry id="LE_chiedere_V">
  <feat att="POS" val="V"/>
  <Lemma>
   <feat att="writtenform" val="chiedere"/>
  </Lemma>
  <SB id="SB_SYNUchiedereV3" SCF="t-attoppin3">
   <feat att="aux" val="avere"/>
  </SB>
 </LexicalEntry>
 <SCF id="t-attoppin3">
  <SA id="synArg_1_Psubj_t-attoppin3">
   <feat att="position" val="0"/>
   <feat att="optionality" val="YES"/>
   <feat att="function" val="subject"/>
   <feat att="syntacticConstituent" val="NP"/>
  </SA>
  <SA id="synArg_4_Pobj_t-attoppin3">
   <feat att="position" val="1"/>
   <feat att="optionality" val="NO"/>
   <feat att="function" val="object"/>
   <feat att="syntacticConstituent" val="NP"/>
  </SA>
  <SA id="synArg_207_Pattoppin3_t-attoppin3">
   <feat att="position" val="2"/>
   <feat att="optionality" val="NO"/>
   <feat att="function" val="objpred"/>
   <feat att="syntacticConstituent" val="PP"/>
   <feat att="introducer" val="in"/>
  </SA>
 </SCF>
</Lexicon>
```

Entry 1: A LMF entry for *chiedere* 'to ask'

4. L-LEME Architecture

In this section we describe the LMF-based L-LEME prototype architecture, presented in figure 2. This prototype takes two LMF lexicons, $\mathcal{A}$ and $\mathcal{B}$, and a set of directives in input and outputs one or more LMF merged lexicon(s) according to different merging scenarios.

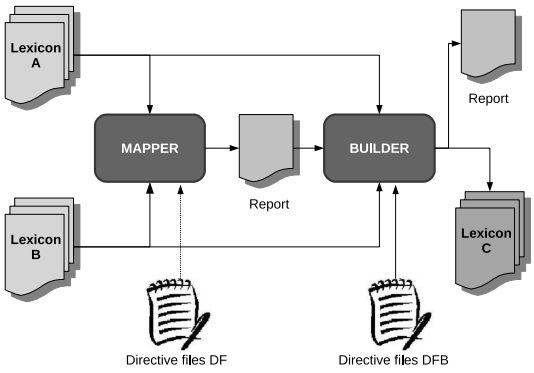

Figure 2: The L-LEME architecture.

The system is composed of two main modules: the *Mapper* and the *Builder*.

Mapper The *Mapper* component takes two lexicons $\mathcal{A}$ and $\mathcal{B}$ and a set of *Directive Files* D_f in input; the *Mapper* produces a report document R which lists all entries of lexicon $\mathcal{A}$ that have a potential match with entries of lexicon $\mathcal{B}$ according to the given directives. The report is meant both as output for the end-user and as input for the *Builder* component.

Builder The *Builder* component merges objects from the two lexicons. It takes the report document R, produced by the *Mapper*, both lexicon ($\mathcal{A}$ and $\mathcal{B}$), and a set of directive files D_{f_b} in input; it outputs one or more new LMF lexicons $\mathcal{C}$ and an updated report R.

4.1. *Mapper* and *Builder* Directives

Directive Files D_f and D_{f_b} are, together with the entering lexicons, the inputs of *Mapper* and *Builder* that use them for taking decisions in both the mapping and building phases, cf. dedicated parts in sections 6. and 7.
Directives make *Mapper* and *Builder* very flexible. They allow users to set the "constraints" under which entries from different lexicons can be "merged". The constraints can be more or less strict on the basis of the desired outcome.

5. The PANACEA Experiment

As mentioned in the introduction, the prototype presented here is meant to be part of a platform for the automatic production of language resources developed by the PANACEA Eu project. In this specific framework, the first test experiments of L-LEME consists in the merging of two lexicons of SubCategorization Frames (SCFs) for Italian: the first one, a subset of the PAROLE lexicon, (Ruimy et al., 1998), is an already existing, manually built lexicon (in the following we will refer to it as the PAROLE-SCF-IT), while the second is a lexicon of SCFs automatically induced from a corpus by a component integrated in the platform (that we will call the PANACEA-SCF-IT). The two lexicons therefore are quite different in terms of content and of background philosophy. For example, PANACEA-SCF-IT contains frequency information which is missing from PAROLE-SCF-IT, while the latter contains information about the position of arguments, which is missing from the former.
The *Mapper* and *Builder* components of the L-LEME architecture have been designed to meet the specific requirements that the merging of two SCF lexicons involves in terms of all the LMF classes that play fundamental roles in such lexicons, i.e. `<LexicalEntry />`, `<SyntacticBehavior />`, `<SubcategorizationFrame />` and `<SyntacticArgument />`. The use of LMF, in fact, ensures the "syntactic interoperability" between lexicons, since the basic structure is the same, cf. section 3.1. This aspect allows the *Mapper* and *Builder* to work on the same (lexicon) structure and define the same (software) structure, cf. section 6.1. However, LMF by itself, does not guarantee the "semantic interoperability", i.e. that the *values* used to adorn/characterize LMF classes are the

same in both lexicons. In the prototype described in this paper the "semantic interoperability" has been managed by the *Mapper* directly from the directive files D_f, cf. section 6.2.

6. The Mapper

In this section we explain how two LMF-based lexicons can be mapped, i.e. how a specific sublexicon $\mathcal{S} \subseteq \mathcal{A} \cap \mathcal{B}$ can be obtained.

6.1. General Mapping Strategy for SubCategorization Frames

In this paper we study the mapping of two SubCategorization Frames (SCFs) lexicons. SCFs belong to the syntactic layer of the lexicon and we can simplify the notation of model (1) since there is no reference to semantic:

$$\mathcal{L} = \mathcal{L}_m \oplus \mathcal{L}_s = \mathcal{L}_m \cup Lnk(\mathcal{L}_m, \mathcal{L}_s) \cup \mathcal{L}_s \qquad (3)$$

$\mathcal{L}$ in model (3) consists of three distinct submodels[6]:

(i) $\mathcal{L}_m$ is the morphological layer of the lexicon and contains (for the scope of the paper) lexical entries and lemmas;

(ii) $\mathcal{L}_s$ is the syntactic layer of the lexicon and contains the SubCategorization Frames along with their Syntactic Arguments structure;

(iii) $Lnk(\mathcal{L}_m, \mathcal{L}_s)$ contains the link between lexical entries and SubCategorization Frames.

We have defined 4 data structures to implement the lexicon in model (3):

$$\mathbb{L} = \begin{cases} \mathbb{L}_m & \text{The morphological layer of the lexicon;} \\ \mathbb{L}_{m2sb} & \text{The link between the morphological part and the Syntactic Behaviour. This is part of the } Lnk(\mathcal{L}_m, \mathcal{L}_s); \\ \mathbb{L}_{sb2scf} & \text{The link between the SB and the SubCategorization Frame. This is again part of the } Lnk(\mathcal{L}_m, \mathcal{L}_s); \\ \mathbb{L}_{sa} & \text{The link between the SCFs and their Syntactic Argument structures;} \end{cases} \qquad (4)$$

6.2. Mapper Directives

For the mapping phase we have defined 4 distinct directives (D_f) that can be used at different levels:

(i) to set those features that *must* be checked to establish equivalence between two Syntactic Arguments as in directive (1);

```
<list2chk>
  <list att="function">
  <list att="realization">
  <list att="introducer">
</list2chk>
```

Directive 1: Features to be checked.

[6]Readers can refer to LMF entry (1) for an overview of these objects. As a recap:
$\mathcal{L}_m \mapsto$ `<LexicalEntry />`;
$\mathcal{L}_m \mapsto$ `<SubcategorizationFrame />`;
$Lnk(\mathcal{L}_m, \mathcal{L}_s) \mapsto$ `<SyntacticBehaviour SCF=".." />`

(ii) to set the features attribute(s) and value(s) pairs that *must* be skipped when two two Syntactic Arguments are compared as in directive (2);

```
<!-- attribute-value pair(s) to skip -->
<list2ignor>
   <ignored feat="function=subject"/>
.....
</list2ignor>
```

Directive 2: Features and values to be skipped.

(iii) to set correspondences between features (attributes and values) of lexicon $\mathcal{A}$ and features of lexicon $\mathcal{B}$ as in directive (3);

```
<corresp>
  <attributes>
    <att a="function" b="function" />
    <att a="syntacticConstituent" b="realization" />
    <att a="introducer" b="introducer" />
  </attributes>
  <values>
    <val a="object" b="direct_object" />
    <val a="NP" b="noun_phrase" />
    <val a="PP" b="prepositional_phrase" />
    <val a="aprepcomp" b="complement" />
    <val a="clauscomp" b="complement" />
  </values>
</corresp>
```

Directive 3: Correspondences between attributes and values.

(iv) a special set of directives, $D_{f_{th}}$, has been introduced to better specify mapping rules. For example directive (4) tells the *Mapper* to set equivalence between Syntactic Arguments that share, at least, $threshold\_min$ features;

```
<!-- equivalence for SA mapper component -->
<threshold min="3" max="3"/>
<!-- extract SA with similarity -->
<similarity min="0.32" max="1"/>
```

Directive 4: $D_{f_{th}}$: thresholds and similarity scores.

6.3. Reading and using mapping Directives

The directives D_f are used by the *Mapper* in order to perform the following operations:

- First of all the list of relevant features (to be checked or ignored) is created according to directives (1) and (2);
- The feature attributes in lexicons $\mathcal{A}$ and $\mathcal{B}$ are unified to a common value, for example "syntacticConstituent" in $\mathcal{A}$ is translated into "realization" as in $\mathcal{B}$, following the equivalences in directive (3);
- The feature values in $\mathcal{A}$ and $\mathcal{B}$ are unified to a common value, for example "object" in $\mathcal{A}$ is translated into "direct_object" as in $\mathcal{B}$, (directive 3). The same directive tells the *Mapper* how to manage a potential $m : n$ correspondence between attributes and/or values. For example "complement" in $\mathcal{B}$ is mapped to a list of values "aprepcomp, clauscomp ..."

- Feature `<function />` with value `<subject />` is skipped in lexicons $\mathcal{A}$ and $\mathcal{B}$.

The first 3 rules apply a sort of "semantic interoperability"; in fact directives (2) and (3) define a mapping among the *values* used to adorn/characterize LMF classes in both lexicons. In other words, the *Mapper* is able to establish that to each *value*, v_A, in lexicon $\mathcal{A}$ corresponds one or more *values*, v_B, in $\mathcal{B}$ and viceversa:

$$\begin{aligned} \forall v_A \in \mathcal{A} : v_A \quad &\mapsto M_{\mathcal{AB}} \equiv \quad [v_B^{(1)}, v_B^{(2)}, \dots v_B^{(k)}] \\ \forall v_B \in \mathcal{B} : v_B \quad &\mapsto M_{\mathcal{BA}} \equiv \quad [v_A^{(1)}, v_A^{(2)}, \dots v_A^{(n)}] \end{aligned} \tag{5}$$

where $M_{\mathcal{AB}}$ is the list of possible values with which the specific v_A is encoded in lexicon $\mathcal{B}$.

6.4. Mapping Algorithm

In this section we provide a detailed outline of the mapping algorithm. The algorithm consists of the following steps:

Reading : Lexicons $\mathcal{A}$ and $\mathcal{B}$ are read and an XML parser is used to create a structure for each lexicon: $\mathcal{L} \mapsto \mathbb{L}$, as in (4);

Extracting the common verbs : Verbs in $\mathbb{A}_m$ and $\mathbb{B}_m$ are extracted and the intersection is used to create a list $\mathrm{L_v}$ of common verbs;

Extracting SCFs : For each verb $\mathrm{V_i}$ in $\mathrm{L_v}$ the Syntactic Behaviours $\mathbb{A}_{m2sb}$ and $\mathbb{A}_{m2sb}$ are extracted and the related SCFs $\mathbb{A}_{sb2scf}$ and $\mathbb{B}_{sb2scf}$ are retrieved;

Comparing syntactic arguments : For each pair of SCFs, all Syntactic Arguments in $\mathbb{A}_{sb2scf_i}$ and $\mathbb{B}_{sb2scf_j}$ are retrieved and a function $diff(\mathbb{A}_{sa}, \mathbb{B}_{sa})$ is used to decide if they are equivalent. This function relies on the the mapping rule defined in equation 5: one given Syntactic Argument (SA) $Sa_{\mathcal{A}} \in \mathcal{A}$ is equivalent to a SA $Sa_{\mathcal{B}} \in \mathcal{B}$ if and only if for all *values* of $Sa_{\mathcal{A}}$, $Sa_{\mathcal{B}}$ is in the list of possible mapped values of $Sa_{\mathcal{A}}$:

$$\begin{aligned} &Sa_{\mathcal{A}} \equiv Sa_{\mathcal{B}} \qquad \text{iif} \\ &\forall v_{\mathcal{A}} \in Sa_{\mathcal{A}} \quad \exists v_{\mathcal{B}} \in Sa_{\mathcal{B}} \quad | v_{\mathcal{B}} \in M_{\mathcal{AB}} \end{aligned} \tag{6}$$

Evaluating *SCF*s similarity : Given the results of $diff(scf(A), scf(B))$, each SCF pair receives a similarity score that is given by the number of positive argument matches normalized by the total number of distinct arguments, cf. section 6.5.

6.5. Cosine similarity

The *Mapper* calculates a score based on a cosine similarity between two SCFs with same/or different number of syntactic arguments. For example, in Italian, the verb *chiedere* ('to ask') has the following valency structures:

(7) (i) *Chiedere **qualcosa a qualcuno***
'to ask **something** to **someone**'

(ii) *'Chiedere **qualcosa a qualcuno in** prestito*
'to get **something** from **someone on** loan'

The two LMF structures are reported in entries (1), for lexicon $\mathcal{A}$, and (2) for $\mathcal{B}$:

```
<Lexicon>
 <LexicalEntry id="PANACEA-SCF-IT_LE_chiedere_V">
  <feat att="POS" val="V"/>
  <Lemma>
   <feat att="writtenform" val="chiedere"/>
  </Lemma>
  <SB id="SB_PANchiedere" SCFs="scf_comp-in_dobj">
   <feat att="aux" val="avere"/>
  </SB>
 </LexicalEntry>
 <SCF id="SCF_comp-in_dobj">
  <SA id="synArg_dobj_scf_comp-in_dobj">
   <feat att="function" val="direct_object"/>
   <feat att="realization" val="noun_phrase"/>
  </SA>
  <SA id="synArg_comp-in_SCF_comp-in_dobj">
   <feat att="function" val="complement"/>
   <feat att="realization" val="prepositional_phrase"/>
   <feat att="introducer" val="in"/>
  </SA>
 </SCF>
</Lexicon>
```

Entry 2: An LMF entry for *chiedere* 'to ask' (lexicon $\mathcal{B}$)

For each SCF, the mapping algorithm creates a vector of syntactic arguments, along with their relevant features (notice how the subject is ignored):

$$\begin{aligned} V_{Sa(A)} \quad = \quad (Sa_{a1} \rightarrow &\begin{bmatrix} func, obj \\ real, np \end{bmatrix}, \\ Sa_{a2} \rightarrow &\begin{bmatrix} func, indobj \\ real, pp \\ int, a \end{bmatrix}) \end{aligned} \tag{8}$$

$$\begin{aligned} V_{Sa(B)} \quad = \quad (Sa_{b1} \rightarrow &\begin{bmatrix} func, obj \\ real, np \end{bmatrix}, \\ Sa_{b2} \rightarrow &\begin{bmatrix} func, complement \\ real, pp \\ int, in \end{bmatrix}) \end{aligned} \tag{9}$$

Vectors $V_{Sa(A)}$ and $V_{Sa(B)}$ are then represented into the combined vector $V_{Sa(A \circ B)}$:

$$\begin{aligned} V_{Sa(A \circ B)} \quad = \quad (Sa_{a1} \rightarrow &\begin{bmatrix} func, obj \\ real, np \end{bmatrix}, \\ Sa_{a2} \rightarrow &\begin{bmatrix} func, indobj \\ real, pp \\ int, a \end{bmatrix}, \\ Sa_{b2} \rightarrow &\begin{bmatrix} func, complement \\ real, pp \\ int, in \end{bmatrix}) \end{aligned} \tag{10}$$

Each vector is then expressed as a binary sequence, where each position corresponds to a Syntactic Argument in the combined vector (10), and its value states whether the SA is present or not.

$$\begin{aligned} V'_{Sa(A)} \quad &= \quad (1, 1, 0) \\ V'_{Sa(B)} \quad &= \quad (1, 0, 1) \end{aligned}$$

The cosine similarity measure between the two vectors is then calculated as per usual. In the example at hand, the similarity between the SubCategorization Frames is[7] 0.33.

6.6. *Blind* Mapping

The mapping algorithm, described above, calculates the mappability of SubCategorization Frames on the basis of their internal structure, i.e. independently from the verb and without taking into account the Syntactic Behaviours of the verbs they are related to. Therefore the mapping algorithm generates a *blind* mapping between SubCategorization Frames: two SCFs are "structurally" similar with a given "cosine similarity" simply because the structure of the related Syntactic Arguments is similar in terms of attributes and values.

7. The Builder

In this section we explain how two LMF-based lexicons can be merged, i.e. how specific (sub)lexicon(s) $\mathcal{C}$ can be obtained combining lexicons $\mathcal{A}$ and $\mathcal{B}$.

7.1. General Building Strategy

The first task of the *Builder* is to verify the *blind* mapping results from the *Mapper* component. This means that, before enriching entries in one lexicon with mapped information from the other, the *Builder* should also take into consideration the connection between the SubCategorization Frame and the Syntactic Behaviour, thus checking if the structural equivalence of two given SubCategorization Frames also corresponds to the equivalence at the syntactic level (syntactic behaviours) of the verb they are related to. In other words, one same verb from the two lexicons can point to SubCategorization Frames that are set "as equivalent" by the *Mapper* but that can have different Syntactic Behaviours in the two lexicons, e.g. by taking different auxiliaries. For example the Italian constructions for *gravare su* ..., 'to weigh on ...', and *rimbalzare su* ..., 'to bounce on ...' has the same SCF, (*i-ppsu*), an intransitive one-argument structure with prepositional phrase introduced by *su* 'on' but they can two different SBs on the basis of the auxiliary they take. In Italian the two verbs can have both the auxiliary *avere*, 'to have' and *essere*, 'to be'. See examples (11):

(11) (i) *Questa cosa ha **gravato sui** risultati/questa cosa è **gravata su** di me*
'this thing has **weighed on** results/this thing is **weighed** down **on** me'

(ii) *la palla ha **rimbalzato sul** muro/la palla è **rimbalzata su** me*
'the ball has **bounced on** the wall/the ball **bounced on** me'

$$\begin{bmatrix} SB_{\mathcal{A}} \\ aux : have \\ freq : 1234 \\ freq\_aux : 23 \end{bmatrix} \rightarrow \begin{bmatrix} SCF_{\mathcal{A}} \\ scf : i-ppsu \end{bmatrix} \begin{bmatrix} ARGS_{\mathcal{A}} \\ realiz. : PP \\ int. : su \end{bmatrix}$$

$$\begin{bmatrix} SB_{\mathcal{B}} \\ aux : be \\ freq : 2345 \\ freq\_aux : 32 \end{bmatrix} \rightarrow \begin{bmatrix} SCF_{\mathcal{B}} \\ scf : i-ppsu \end{bmatrix} \begin{bmatrix} ARGS_{\mathcal{B}} \\ realiz. : PP \\ int. : su \end{bmatrix}$$

As explained in section 6.6. the *Mapper* reports that $SCF_{\mathcal{A}}$ is equivalent to $SCF_{\mathcal{B}}$ with cosine similarity (cs) equal to 1, $cs = 1$. When evaluating the two mapped SubCategorization Frame structures "i-ppsu" from lexicons $\mathcal{A}$ and $\mathcal{B}$, the *Builder* will not consider the mapping between e.g. i-ppsu(+have) (from, say, $\mathcal{A}$) and i-ppsu(+be) (from $\mathcal{B}$) as valid since they do not share the same auxiliary and consequently will not merge the information of the two entries.

7.2. Builder Directives

For the building phase we have defined 2 distinct directives (D_{f_b}) that can be used at different levels:

(i) to define different merging scenarios, directive (5).

```
<!-- merging scenarios -->
<mergings>
   <merging type="union" />
      <operation type="union" />
   </merging>
   <merging type="union_intersection" />
      <operation type="union intersection" />
   </merging>
</mergings>
```

Directive 5: Merging scenarios.

(ii) to choose the format of *values* for the output lexicon $\mathcal{C}$, as in directive (6).

```
<!-- choose A or B as selected format -->
<output_format>
   <use_lexicon source="A" />
</output_format>
```

Directive 6: The values of $\mathcal{C}$ can be selected. In this example the output will use the format of $\mathcal{A}$

(iii) a special directive $D_{f_{beq}}$ has been introduced to include/exclude the check of the auxiliary when two Syntactic Behaviour are compared, as in directive (7)

(iv) to set two features as equivalent if they have the same attribute and value, again directive $D_{f_{beq}}$;

```
<!-- equivalence for SB builder component -->
<equivalence>
 <equiv att="check_auxiliaries" val="true" />
 <equiv att="features_equal_by" val="same_att_and_val"/>
</equivalence>
```

Directive 7: $D_{f_{beq}}$, parameters for *Builder*

7.3. Reading and using building Directives

The version of the *Builder* that has been implemented in our experiment merges the Syntactic Behaviours of the two lexicons. It works on both the report R and the two lexicons to recreate the data structures $\mathbb{L}_{m2sb}$ and $\mathbb{L}_{sb2scf}$ (cf. section 6., equation 4) containing information from the lexicons that "potentially" can be merged. The *Builder* uses the directive D_{f_b} to to perform the following operations:

[7] Notice that in our test case experiment, because of the specific characteristics of the automatically acquired lexicon ($\mathcal{B}$) that never represents subjects in the SubCategorization Frames because they are not acquired by the inductive module, the similarity will never be very high. Notice also that the directives allowed us to ignore the *function=subject* in cosine similarity (cf: 6.3.); if the subject were taken into account (supposed as always present in lexicon $\mathcal{B}$, the cosine similarity would increase to 0.50

- perform the check on auxiliaries as in directive (7) before merging information (i.e. features) for Syntactic Behaviours;
- merge information according to directive (7);
- provide output lexicons according to directive (5). We have defined three different merging scenarios

  (i) **intersection**: $\mathcal{C} = \mathcal{A} \cap \mathcal{B}$, where the symbol $\cap$ indicates the lexical entries of $\mathcal{A}$ and $\mathcal{B}$ which have matched according to the mapping algorithm. This means that the intersection merges only common lexical entries between lexicons $\mathcal{A}$ and $\mathcal{B}$. The *intersection* enriches the common lexical entries with the information for Syntactic Behaviours coming from both $\mathcal{A}$ and $\mathcal{B}$. All the other lexical entries are discarded.

  (ii) **union**: $\mathcal{C} = \{(A \cap B) \cup \overline{A} \cup \overline{B}\}$. With respect to the intersection, the *union* (represented by the symbol $\cup$) unites the three (standard) parts of each union (intersection with the two complements);

  (iii) $\mathcal{L}$ **enriched**: in this case $\mathcal{C} = \mathcal{L}^{+}$. Where $\mathcal{L} \in (\mathcal{A}, \mathcal{B})$ and the $^{+}$ means the entries of lexicon $\mathcal{A}|\mathcal{B}$ are enriched with information of *matched* entries from $\mathcal{B}|\mathcal{A}$;

Table 1 shows that if the equivalence rule[8] in directive (7) is set to `same_att` then attributes with same name but different value are outputted once with the value that is in the source lexicon chosen in (6); if `same_att_and_val` is chosen, they are repeated so that each value is maintained.

eq. rule	**Lexicon $\mathcal{A}$**	**Lexicon $\mathcal{B}$**	**Merged Features**
SA	$\begin{bmatrix} freq : 1234 \\ freq_{aux} : 23 \end{bmatrix}$	$\begin{bmatrix} freq : 4321 \\ freq_{aux} : 32 \end{bmatrix}$	$\begin{bmatrix} freq : 1234 \\ freq_{aux} : 23 \end{bmatrix}$
SAV	$\begin{bmatrix} freq : 1234 \\ freq_{aux} : 23 \end{bmatrix}$	$\begin{bmatrix} freq : 4321 \\ freq_{aux} : 32 \end{bmatrix}$	$\begin{bmatrix} freq : 1234 \\ freq_{aux} : 23 \\ freq : 4321 \\ freq_{aux} : 32 \end{bmatrix}$

Table 1: Merged features depend on equivalence rule

7.4. Building Algorithm

In this section we provide a detailed outline of the building algorithm. The algorithm consists of the following steps:

Reading Lexicons : Lexicons $\mathcal{A}$ and $\mathcal{B}$ are read and an XML parser is used to create a structure as in model (4) for each lexicon: $\mathcal{L} \mapsto \mathbb{L}$. This step is the same executed by the mapping algorithm (cf. section 6.4.);

Reading Report : The report R is read to extract from lexicons $\mathcal{A}$ and $\mathcal{B}$ entries that have been mapped;

Extracting Structures : The structures $\mathbb{L}_{m2sb}$ (Syntactic Behaviours) and $\mathbb{L}_{sb2scf}$ (link between Syntactic Behaviours and SubCategorization Frames) are extracted from each lexicon and (eventually) updated with information from R, cf. section 8.3.;

Extracting the correct SBs : Directive 5 is used by the *Builder* to filter out the matching results provided by the *Mapper*. The *Mapper* reports SCFs from both lexicons that have been mapped without taking into account whether their related Syntactic Behaviour could be merged. Directive (5) tells the *Builder* whether the auxiliary must be used to resolve this situation so that only Syntactic Behaviours with the same auxiliary are finally set as "merged";

Outputting lexicon(s) Directive 7 is used to output the lexicon(s) according to the merging scenarios;

Updating Report R The report R is updated with details on the building phase, cf. section 8.2.

8. Report

The report R is a list of common lexical entries (verbs) between lexicons $\mathcal{A}$ and $\mathcal{B}$. It is firstly created by the *Mapper*, then displayed to the users and finally updated by the *Builder*. Therefore, R is meant for both machine and human. The former is an XML document which reflects the input parameters (the two lexicons) and the two components involved in the process; the latter is an HTML rendering which is editable so that the users become an essential part of the process, cf. section 8.3. The XML report R is composed by two different sections:

Directives Report (R_d) The R_d fragment of the report is meant for the end-users and simply summarizes the directives that have been used to provide the LMF lexicon(s) in output. The directives are grouped per component: D_f and $D_{f_{th}}$ for the *Mapper*, D_{f_b} and $D_{f_{beq}}$ for the *Builder*, cf. (1);

```
<directives>
 <mapper>
  <checked source="lexicon_a" list="[function, syntacticConstituent,
           introducer]"/>
  <checked source="lexicon_b" list="[function, realization,
           introducer]"/>
  <ignored list="{function=subject}"/>
  <threshold min="3" max="3"/>
  <similarity min="0.5" max="1"/>
 </mapper>
 <builder>
  <equiv att="check_auxiliaries" val="true" />
  <equiv att="features_equal_by" val="same_att_and_val" />
  <outputFormat source=lexicon_a" />
 </builder>
</directives>
```

Report 1: R_d after mapping and building phases.

Matches Report (R_m) The R_m recaps the identifiers of the matching SubCategorization Frames along with their similarity. This section of the report contains information from the mapping and building components (cf. sections 8.1. and 8.1.) and a serialization of the data structures $\mathbb{L}_{sb2scf}$ and $\mathbb{L}_{sa}$ for both lexicons.

[8] In table 1 sa stands for `same_att`, while SAV stands for `same_att_and_val`

8.1. Mapping Report

The *Mapper* components produces a report which contains the SCF pairs that have matched with a given similarity. The SA structure of matching SCFs is also reported. The mapping report is rendered as an HTML page and presented to the end users, see figure 3.

Figure 3: HTML page from report R automatically built by the *Mapper*.

8.2. Building Report

The *Builder* components updates the report R with information related to the merged Syntactic Behaviours. These SBs are enriched with their features extracted from lexicons from $\mathcal{A}$ and $\mathcal{B}$ according to directive (5). The complete structures of SB, $\mathbb{L}_{sb2scf}$, are then serialized.

8.3. Updating the report R

The *Builder* component is able to automatically merge Syntactic Behaviours from different lexicons which are linked to SubCategorization Frames that have a similarity $cs = 1$. In this case, in fact, the human intervention is no longer needed. However, the *Mapper* can be run to extract SCFs is a given range of similarity, directive (4). The HTML report presented in figure 3 is editable by the users so that they, as experts, can "force" the similarity of two mapped SCFs to be 1.

This aspect of the L-LEME architecture plays a fundamental role in the building phase: the *Builder* components filters the report R and extracts the entries with the similarity, $cs = 1$. From these entries the *Builder* recreates the $\mathbb{L}_{sb2scf}$ and $\mathbb{L}_{sa}$ data structures before outputting the LMF lexicon(s) with merged Syntactic Behaviours.

9. Results for the PANACEA Experiment

L-LEME has been run over the two lexicons: PAROLE-SCF-IT and PANACEA-SCF-IT which contain 3214 and 31 verbs respectively. In this experiment, L-LEME has been run twice using different values for $D_{f_{th}}$ in the two different runs:

run #1	run #2
$D_{f_{th}} = \begin{cases} th\_min = 3 \\ th\_max = 3 \\ sim\_min = 0.33 \\ sim\_max = 1 \end{cases}$	$D_{f_{th}} = \begin{cases} th\_min = 3 \\ th\_max = 3 \\ sim\_min = 1 \\ sim\_max = 1 \end{cases}$

Table 2: Different values for $D_{f_{th}}$

`run #1` generates all possible matches, from similarity confidence (cs) between 0.33 (minimum threshold) to 1.0. Table 3 shows the results of this run by grouping all matching pairs (834 matches) with different thresholds of similarity in 7 groups.

PANACEA Experiment `run #1`			
Matches	Similarity	Number	Percentage
Matches: 834	sim=1	120	14.4%
	sim=0.82	3	∼ 0.3%
	sim=0.71	383	45.9%
	sim=0.58	15	1.8%
	sim=0.5	273	32.7%
	sim=0.41	36	∼ 0.5%
	sim=0.33	4	4.3%
Total		834	100%

Table 3: Figures for `run #1`

From figures presented in table 3 we note that $\sim$ 65% of matching SubCategorization Frames has a confidence $cs \geq 0.58$. In addition the 73.5% of these matched SCFs reach a confidence of 0.71. This means that the lexicon PAROLE-SCF-IT, which has been built by human introspection, has a good projection in the daily use of the language. Figure 4 shows the maximum similarity (confidence) vs. some SubCategorization Frames for the following Italian verbs: *accusare* ('to accuse'), *amare* ('to love'), *andare* ('to go'), *aprire* ('to open'), *arrivare* ('to arrive') and *cercare* ('to find'). The maximum similarity is defined as the maximum value of the similarity that a given SubCategorization Frame can assume for a given verb.

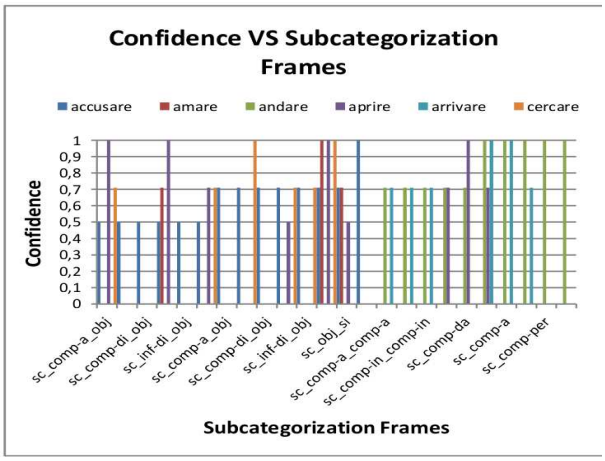

Figure 4: Confidence vs. SubCategorization Frames for some Italian verbs.

`run #2` only selects pairs having confidence 1.0, that is only perfect matches that do not require human intervention and it is used for the building phase. The similarity 1.0 guarantees, indeed, that the mapped SCFs are equal so that the features of related Syntactic Behaviours can be automatically merged.
We report the distinct numbers of SCFs and SBs in both lexicons as well as in the report R, (table 4). A detailed view of data extracted from the report R in `run #2` is presented in tables (5) and (6).

PANACEA Experiment: data from $\mathcal{A}$ and $\mathcal{B}$		
Source	**Distinct SCFs**	**Distinct SBs**
PAROLE-SCF-IT	124	244
PANACEA-SCF-IT	79	496
REPORT	60	201

Table 4: SCFs and SBs from $\mathcal{A}$ and $\mathcal{B}$

PANACEA Experiment: Report in run #2			
SubCategorization Frames			
Verb-dependent Matched SCFs	Distinct Matched SCFs		
	Total Pair Matches	Source	Matches
120	47	Lexicon $\mathcal{A}$	47
		Lexicon $\mathcal{B}$	13
Syntactic Behaviours			
Total SBs	Distinct Merged SBs		
	Total Merged Matches	Source	Matches
201	107	Lexicon $\mathcal{A}$	107
		Lexicon $\mathcal{B}$	75

Table 5: Figures for Report R in `run #2`

Report				
SCFs	Lexicon $\mathcal{A}$		Lexicon $\mathcal{B}$	
	Extracted	Matched	Extracted	Matched
60	47	47	13	13
SBs	Lexicon $\mathcal{A}$		Lexicon $\mathcal{B}$	
	Extracted	Merged	Extracted	Merged
201	120	107	81	75

Table 6: Report R, extracted vs. matched/merged entities

From table 5 we note that 47 distinct SubCategorization Frames correspond to 120 verb-dependent SCFs and thus to 120 Syntactic Behaviours.[9] In reality, when checking the matching at SB level though, only 107 of these SB pairs have matching features (namely auxiliary) and can be thus included in the final lexicon.

From tables 4, 5 and 6 we deduce some interesting clues:

(i) on average the $\sim 29.5\%$ of (aggregate) SubCategorization Frames are automatically mapped;

(ii) in the lexicon PAROLE-SCF-IT, the mapped SCFs cover $\sim 38\%$ of the total, while in PANACEA-SCF-IT, the percentage decreases to $\sim 17\%$. This means that more than one SCF from PAROLE-SCF-IT map on the same SCF of PANACEA-SCF-IT, in fact, the values of the features in PANACEA-SCF-IT are under-specified: directive (3) shows how the `complement` in PANACEA-SCF-IT is mapped into `aprepcomp, aclauscomp...` in lexicon PAROLE-SCF-IT;

(iii) on average the $\sim 14.5\%$ of (aggregate) Syntactic Behaviours are automatically merged;

(iv) in lexicon PAROLE-SCF-IT, the merged SBs cover $\sim 42\%$ (expected $\sim 49.2\%$), while in PANACEA-SCF-IT the percentage decreases to $\sim 15\%$ (expected $\sim 16.3\%$)

In point (iv) above, the percentages decrease because of the auxiliary. In fact, an analysis of the report R has provided evidence that 13 Syntactic Behaviours from lexicon PAROLE-SCF-IT have a different auxiliary in lexicon PANACEA-SCF-IT so that they are not merged.

10. Conclusions

In this paper we presented L-LEME, an architecture which exploits the LMF representation standard for lexicons in order to generalize the merging process. We created a prototype that has been applied to an actual case study, which consists in merging an extracted SCF lexicon from the PANACEA project with a subset of the PAROLE lexicon. The results are quite encouraging, since it was possible to obtain $\sim 30\%$ of automatic mapping among SubCategorization Frames by implementing a set of directives D_f, which represents a coarse-grained mapping between key features in the two lexicons.
As a future work, we intend to explore the possibility of obtaining a better mapping by applying different weights to different features. We also intend to improve the usability by transforming the tool into a web service; users will be allowed to upload their input lexicons, to edit the directives in a user friendly environment and to check and modify the intermediate mapping report before building their result lexicon.

Acknowledgments

This work is supported by the EU FP7 ICT Project PANACEA (Grant Agreement no. 248064).

11. References

Daniel Ka-Leung Chan and Dekai Wu. 1999. Automatically merging lexicons that have incompatible part-of-speech categories.

[9] See section 3.1.

R. S. Crouch and T. H. King. 2005. Unifying lexical resources. In *Proceedings of Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*, pages 32–37, Saarbruecken, Germany.

Gil Francopoulo, Nuría Bel, Monte George, Nicoletta Calzolari, Monica Monachini, Mandy Pet, and Claudia Soria. 2008. Multilingual resources for nlp in the lexical markup framework (lmf). *Language Resources and Evaluation*.

Andrea Marchetti, Maurizio Tesconi, and Salvatore Minutoli. 2005. Xflow: An xml-based document-centric workflow. In *WISE*, pages 290–303.

Miguel A. Molinero, Benot Sagot, and Lionel Nicolas. 2009. Building a morphological and syntactic lexicon by merging various linguistic resources. In *Proceedings of 17th Nordic Conference on Computational Linguistics (NODALIDA-09)*, Odensee, Danemark.

Monica Monachini, Nicoletta Calzolari, Khalid Choukri, Jochen Friedrich, Giulio Maltese, Michele Mammini, Jan Odijk, and Marisa Ulivieri. 2006. Unified lexicon and unified morphosyntactic specifications for written and spoken italian. In *Proceedings of the LREC 2006)*, Genova, Italy.

Muntsa Padró, Núria Bel, and Silvia Necsulescu. 2011. Towards the automatic merging of lexical resources: Automatic mapping. In *RANLP*, pages 296–301.

Nilda Ruimy, Ornella Corazzari, Elisabetta Gola, Antonietta Spanu, Nicoletta Calzolari, and Antonio Zampolli. 1998. The european le-parole project: The italian syntactic lexicon. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'98)*, Granada, Spain, may. European Language Resources Association (ELRA).

Maurizio Tesconi, Andrea Marchetti, Francesca Bertagna, Monica Monachini, Claudia Soria, and Nicoletta Calzolari. 2006. Lexflow: A system for cross-fertilization of computational lexicons. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *ACL*. The Association for Computer Linguistics.

Merging heterogeneous language resources and tools in a digital library

Anelia Belogay, Diman Karagiozov,
Svetla Koeva, Cristina Vertan,
Adam Przepiórkowski, Maciej Ogrodniczuk, Dan Cristea, Eugen Ignat, Polivios Raxis
E-mail: anelia@tetracom.com, diman@tetracom.com, svetla@dcl.bas.bg, cristina.vertan@uni-hamburg.de, adamp@ipipan.waw.pl, maciej.ogrodniczuk@gmail.com, dcristea@info.uaic.ro, eugen.ignat@info.uaic.ro, raxis@atlantisresearch.gr

Abstract

Merging of Language Resources is not only a matter of mapping between different annotation schemata but also of linguistic tools coping with heterogeneous annotation formats in order to produce one single output. In this paper we present a web content management system ATLAS which succeeded to integrate and harmonize resources and tools for six languages, including four less-resourced[1] ones. As a proof of the concept we implemented a personal digital library (i-Librarian). We performed two user evaluation rounds in order to assess the increase of user productivity when the language technologies are harnessed in the processes of content management.

Keywords: multilingual content management system, UIMA, language processing chains

1. Introduction

During the last twenty years a large number of monolingual and bilingual language resources and tools were produced. Especially in the 90'ties, LRs were produced for a dedicated application, without any concern regarding reuse, respectively standardization.

With the development of statistical and machine learning approaches, the existence of reusable training data (i.e. language resources) became crucial, especially for multidisciplinary and multi-domain systems. While standards for encoding corpora, lexicons and treebanks are widely used, there is still a big issue with the harmonisation of output produced by processing tools. Complex systems usually require a pipeline of language processing engines, and there is a need of harmonization of tagsets, input and output formats. Recent European and national initiatives contributed a lot to the development of monolingual complex linguistic processing pipelines. However, there is still a big gap at the multilingual level because each language has its own linguistic particularities, tools use different linguistic formalisms, and engines have different degree of complexity.

In this paper we describe the approach used in order to embed multilingual language technology support into a Web-based Content Management System[1]. The system is developed for six languages (Bulgarian, English, German, Greek, Polish and Romanian) and includes technology for automatic categorization, summarization and translation, extraction of concepts (ideas) as well as cross-lingual retrieval. All these rely on language specific pipelines harmonised through UIMA framework. As the whole system is fairly complex and allows the creation of different Web services based on the above mentioned technology, we present here one particular Web service, i-Librarian.

2. i-Librarian Features

i-Librarian (http://www.i-librarian.eu) was developed as a free online library to assist users in organising and managing multilingual content. i-Librarian is powered by ATLAS – an open-source software platform for multilingual Web content management, containing i-Publisher and Linguistic framework. i-Publisher is a Web-based instrument for creating, running and managing dynamic content-driven Web sites (i-publisher.atlasproject.eu). The Linguistic framework enriches the content in these Web sites with automatic annotation of important words, phrases and names, suggestions for categories and keywords, summary generation and computer-aided translations in English, Bulgarian, German, Greek, Polish and Romanian.

As a service powered by ATLAS, i-Librarian is enhanced by the innovative features provided by its linguistic framework. On uploading new documents to i-Librarian, the library system automatically provides the user with an extraction of the most relevant information (phrases, named entities, and keywords). i-Librarian then uses this information to generate suggestions for classification in the library catalogue as well as a list of similar documents. Finally, the system compiles a summary and translates it in all supported languages. Among supported document formats are Microsoft Office documents, PDF, OpenOffice, books in various electronic formats, HTML pages and XML documents. Users have exclusive rights to manage content in the library at their discretion.

3. Linguistic framework

The ATLAS framework employs technologically and linguistically diverse natural language processing (NLP) tools in a platform, based on UIMA[2]. The UIMA pluggable component architecture and software framework are designed to analyse content and to structure it. The ATLAS core annotation schema, as a

[1]The work reported here was performed within the ATLAS project funded by the European Commission under the CIP ICT Policy Support Programme (http://www.atlasproject.eu).

[2] Unstructured Information Management Application. See http://uima.apache.org/.

uniform representation model, normalizes and harmonizes the heterogeneous nature of the NLP tools. The language processing tools are integrated in a language processing chain (LPC), so the output of a given NLP tools is used as an input for the next tool in the chain. For example, the sentence boundaries and the token tags are used by the POS tagger; respectively, the tagger's annotations are used by the lemmatizer etc.

The main challenge was to harmonize the output of different processing tools as well as their degree of annotation. To our knowledge it is for the first time when such harmonisation is realised within a real system and with involvement of less resourced languages. Two steps were performed:

- the minimal set of required tools was identified in order to ensure accuracy near to state-of the art for all languages;
- minimal linguistic annotation required for each language was identified and the correspondent was model defined; this model contains obligatory features (like PoS, case, gender, number) but also optional ones (which may be used for one language or another: e.g. a special feature to make connection between verb particles and verbs in German); all these features are part of the UIMA-based processing chains, so annotation tags don't need to be harmonised across languages.

In order to ensure the interoperability and coherence of the NLP tools, they have been tuned for higher accuracy and performance.

The processing of text in the system is split into three subtasks, executed sequentially. The text is extracted from the input source (text or binary documents) in the "pre-processing" phase. The text is annotated by several NLP tools, chained in a sequence in the "processing" phase. Finally, the annotations are stored in a data store, such as relational database or file system.

The architecture, shown on Figure 1, is based on asynchronous message processing patterns and allows the processing framework to be scaled horizontally.

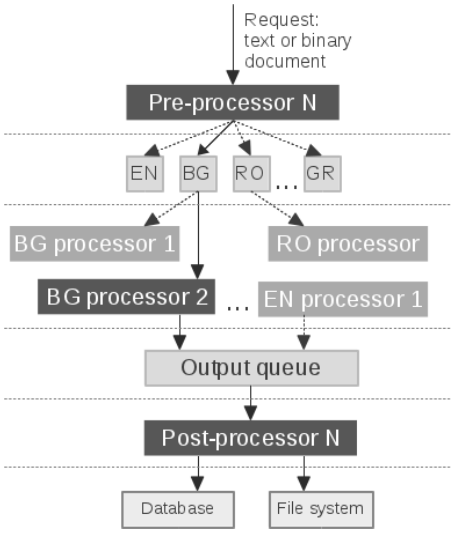

Figure 1. Top-level architecture of the Linguistic framework

3.1 Language Processing Chain components

The baseline LPC for each of the project language includes a sentence and paragraph splitter, tokeniser, part of speech tagger, lemmatizer, optional word sense disambiguation component, noun phrase chunker and named entity extractor. Figure 2 depicts the LPC basic components and their execution sequence.

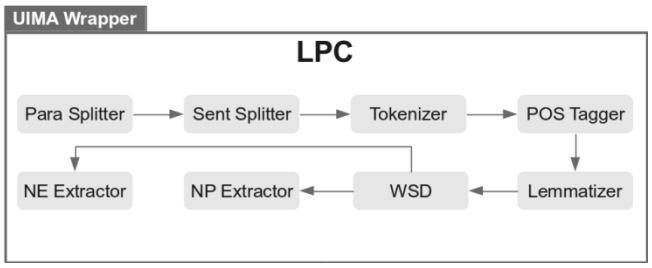

Figure 2. Components of a language processing chain

In addition to the general language-independent rules for paragraph and sentence splitting, the sentence splitters and tokenizers utilize language specific knowledge for word and sentence grammatical structure, for each of the target languages.

The POS taggers exploit different classification algorithms depending on the language to be processed: the OpenNLP [3] tagger for English is based on Maxent and Perceptron classifiers; the SVM Bulgarian tagger applies classification based learning (Gimenez J., Marquez L., 2004); the Greek and Polish taggers are transformation based. Moreover, some of the algorithms are combined or extended to improve the robustness. The German tagger is a combination of the TnT tagger [4] and a constraint dependency disambiguation, while the Romanian tagger exploits a statistical and a symbolic method based on the context information (the sets of tags retrieved by the maximum entropy statistical component).

Most of the lemmatizers (i.e. for Bulgarian, English, and Greek) are rule based engines integrated with large morphological lexicons, while for Polish the lemmatisation is combined with shallow parsing (Degórski, 2011).

The Noun Phrase Extractors (NPE) and Named Entity Recognizer (NER) have been developed using language-independent tools and later customised for each of the target languages. The Spejd tool (Buczyński and Przepiórkowski, 2009) - a shallow parser exploiting cascade regular grammars - has been promoted for Polish and Greek by implementing language-specific parsing rules. The ParseEst (Genov and Koeva, 2011), a parser based on the finite-state framework, is used for the identification of different types of syntactic entities for Bulgarian and English - phrase structures, named entities, multi-word expressions, etc. The German NP chunker is obtained by applying NP constituent rules on dependency structures. The Polish NER tool (Savary, Waszczuk and Przepiórkowski, 2010) - a statistical CRF-based named

[3] http://incubator.apache.org/opennlp

[4] http://www.coli.uni-saarland.de/~thorsten/tnt

entity recognizer has been extended to cover seven major types of named entities, namely dates, money, percentage and time expressions apart from originally recognized hierarchical names of organizations, locations and persons.
The annotations produced by each LPC along with additional statistical methods are subsequently used for detection of key words and phrases, generation of extractive summary, multi-label text categorisation and machine translation.

3.2 Text categorization

A language-independent text categorisation tool which works for heterogeneous domains was implemented by the project. This engine is exemplified by documents in Bulgarian and English. The engine employs different categorization algorithms, such as Naïve Bayesian, relative entropy, Class-Feature Centroid (CFC) (Guan, Zhou and Guo, 2009), Support Vector Machines. The text, annotated by the NLP tools feeds the categorisation engine. New algorithms can be integrated easily because the categorisation engine is based on OSGI platform. The results of each of the categorization algorithms are consolidated by a custom label-voting system.

3.3 Summarization

The diverse content in i-Librarian requires a flexible generation of summary depending on the text length. The baseline method, used for both short and long texts, relies on shallow level language independent heuristics that rank sentences by summing up scores of cue words such as first occurrences, inclusion of capital letters, repetitions, etc.
We adopted two state-of-the-art summarisation approaches – for short and long texts. The summarisation approach (Cristea, Postolache, and Pistol, 2005) used for short texts exploits cohesion and coherence properties of the text on discourse structures that resemble rhetorical trees. The short texts summarisation chain is a combination of the output of the language processing chains and an anaphora resolver, clause splitter, discourse parser, and summarizer. Currently, the summarisation chain is implemented for English and Romanian.
The method used to produce summaries of long texts assembles a template summary from different pieces of information, specific to different genres. For instance, a crime novel summary is made of a number of sentences, uttering pieces of disparate knowledge, extracted using pattern matching templates, such as the place of the action, the main characters, etc.

3.4 Machine translation

The ATLAS machine translation engine implements the hybrid paradigm, combining an example-based (EBMT) component and a Moses-based statistical approach (SMT). Firstly, the results of the categorisation engine are used to select the most appropriate translation model and MT sub-component. Each input to the translation engine is then processed by the example-based MT sub-component. If the input, as a whole, or important chunks of it are found in the translation database then the translation equivalents are used and if necessary combined (Gavrila, 2011). In all other cases the input is sent further to the SMT component which uses a PoS and domain factored model following (Niehues and Waibel, 2010).
Similar to the architecture of the categorization engine, the translation system in ATLAS is able to accommodate and use different 3rd party translations engines, such as Google, Bing Translator, Lucy, Yahoo translators.

4. System evaluation

The scope of our current work does not cover the examination of accuracy of each individual language processing chain but focuses on evaluation of the quality and performance of the system, as well as the satisfaction of the users with a system that harnesses multilingual processing tools in the processes of the content management.
The technical evaluation uses indicators that assess technical elements such as:

- Overall quality and performance attributes (MTBF, uptime, response time);
- Performance of specific functional elements (content management, machine translation, cross-lingual content retrieval, summarisation, text categorisation).

The user evaluation assesses the level of satisfaction with the system. We measure non-functional elements such as:

- User friendliness and satisfaction, clarity in responses and ease of use;
- Adequacy and completeness of the provided data and functionality;
- Impact on certain user activities and the degree of fulfilment of common tasks.

We have planned for three rounds of user evaluation; all users are encouraged to try online the system, freely, or by following the provided base-line scenarios and accompanying exercises. The main instrument for collecting user feedback is an online interactive electronic questionnaire (http://ue.ATLASproject.eu/). All that is needed is to select a user group and follow the onscreen suggestions.
The second round of user evaluation is scheduled for March 2012 involving 180 users, while the first round took place in Q1 2011, with the participation of 33 users. The overall user impression was positive and the Mean value of each indicator (in a 5-point Likert scale) was measured on AVERAGE or ABOVE AVERAGE. Figure 3 and 4 depicts the user opinion on their increased productivity and satisfaction with the available functionality.

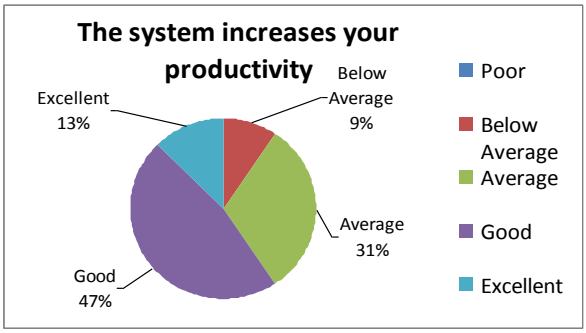

Figure 3: Distribution of user opinion on productivity increase.

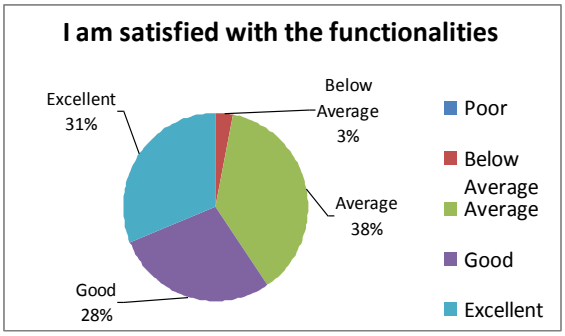

Figure 4: Distribution of user opinion on satisfaction with the system functionalities.

5. Conclusion and Further work

The abundance of knowledge allows us to widen the application of NLP tools, developed in research environment. The combination of Web content management and state of the art language technologies helps the reader to cross the language barrier, to spot the most relevant information in large data collections and to keep all this information in order. The tailor made voting system maximizes the use of the different categorization algorithms. Two distinctive approaches summarise short and long texts and their translation are provided by state-of-the-art hybrid machine translation system.

ATLAS linguistic framework has been released in February 2012 as open-source software. The language processing chains for Bulgarian, Greek, Romanian, Polish and German will be fully implemented by the end of 2012.

6. References

Buczyński, A., Przepiórkowski, A. (2009). Spejd: A Shallow Processing and Morphological Disambiguation Tool. In Zygmunt Vetulani, Hans Uszkoreit (eds.), *Human Language Technology: Challenges of the Information Society, Lecture Notes in Artificial Intelligence 5603*, Berlin, Germany. Springer-Verlag, pp. 131--141.

Cristea, D., Postolache, O., Pistol, I. (2005). Summarisation through Discourse Structure. *Computational Linguistics and Intelligent Text Processing, 6th International Conference CICLing 200*. Mexico City, Mexico: Springer LNSC, vol. 3406; ISBN 3-540-24523-5, pp 632--644.

Degórski, Ł. (2011). Towards the Lemmatisation of Polish Nominal Syntactic Groups Using a Shallow Grammar. In Pascal Bouvry, Mieczysław A. Kłopotek, Franck Leprevost, Małgorzata Marciniak, Agnieszka Mykowiecka, and Henryk Rybiński, editors, *Security and Intelligent Information Systems: International Joint Conference, SIIS 2011*, Warsaw, Poland, June 13-14, 2011, Revised Selected Papers, volume 7053 of Lecture Notes in Computer Science. Springer-Verlag, 2011.

Gavrilam M. (2011). Constrained recombination in an example-based machine translation system. In M. L. Vincent Vondeghinste (Ed.), *15th Annual Conference of the European Association for Machine Translation*, Leuven, Belgium, pp. 193--200.

Genov, A., Koeva, S. (2011). Bulgarian Language Processing Chain. GSCL, *Integration of multilingual resources and tools in Web applications Workshop*. Hamburg, Gemany.

Giménez J., Márquez, L. (2004). Svmtool: A general POS tagger generator based on support vector machines. *IV International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, pp. 43-- 46.

Guan, H., Zhou, J., Guom, M. (2009). A class-feature-centroid classifier for text categorization. *18th International World Wide Web Conference*, Madrid, Spain, pp. 201--210.

Niehuesm J., Waibel, A. (2010). Domain Adaptation in Statistical Machine Translation using Factored Translation Models. *EAMT 2010*. Saint-Raphael.

Ogrodniczuk, M., Karagiozov, D. (2011). ATLAS — The Multilingual Language Processing Platform. *Procesamiento del Lenguaje Natural, 47,* pp. 241--248.

Savary, A., Waszczuk, J., Przepiórkowski, A.. (2010). Towards the Annotation of Named Entities in the National Corpus of Polish. *Seventh International Conference on Language Resources and Evaluation, LREC 2010*. Valletta, Malta.

Automatized Merging of Italian Lexical Resources

Thierry Declerck[1,2], Stefania Racioppa[1], Karlheinz Mörth[2]
[1] DFKI GmbH, Language Technology Lab
Stuhlsatzenhausweg, 3
D-66123 Saarbrücken, Germany
[2] Institute for Corpus Linguistics and Text Technology (ICLTT), Austrian Academy of Science
Sonnenfelsgasse 19/8, 1010 Vienna, Austria
E-mail: declerck@dfki.de, stefania.racioppa@dfki.de, karlheinz.moerth@oeaw.ac.at

Abstract

In the context of a recently started European project, TrendMiner, there is a need for a large lexical coverage of various languages, among those the Italian language. The lexicon should include morphological, syntactic and semantic information, but also features for representing the level of opinion or sentiment that can be expressed by the lexical entries. Since there is no yet ready to use such lexicon, we investigated the possibility to access and merge various Italian lexical resources. A departure point was the freely available Morph-it! lexicon, which is containing inflected forms with their lemma and morphological features. We transformed the textual format of Morph-it! onto a database schema, in order to support integration process with other resources. We then considered Italian lexicon entries available in various versions of Wiktionary for adding further information, like origin, uses and senses of the entries. We explore the need to have a standardized representation of lexical resources in order to better integrate the various lexical information from the distinct sources, and we also describe a first conversion of the lexical information onto a computational lexicon.

Keywords: Lexical Resources, Standards, Computational Lexicon

1. Introduction

In the context of a recently launched European R&D project, TrendMiner[1], there is a need for a large lexical coverage of Italian language. The lexical resources should include information about morphology, syntax and semantics, but also about opinions or sentiments that can be carried by the entries. The lexicon should also be easily extendable to new types of expressions, like those occurring in micro-blogs, twitter etc. We are therefore experimenting with integration issues of existing lexical resources, starting for now with good quality lexical data available from both language specialists and collaborative efforts. In a next step we will investigate how to integrate in a lexical framework "lower quality" or "noisy" lexical data, as these are typically used in short messaging frameworks or other forms of social media.

2. The First Set of Resources

A starting point for our work was a set of Italian resources made available to the NooJ community. Those relatively limited resources[2] gave us in first line the representation format for the NooJ resources, both for lexical entries and inflexion paradigms, against which we could start the porting of a larger available Italian lexicon, Morph-it![3], which contains more than 35.000 lemmas.

The textual format of the Morph-it! lexicon consists in a list of triples displaying a full-form, its corresponding lemma and the associated morpho-syntactic information, as can be seen in the examples in Table 1.

```
casco casco NOUN-M:s
caschi casco NOUN-M:p
…
casellari casellario ADJ:pos+m+p
casellari casellario NOUN-M:p
casellaria casellario ADJ:pos+f+s
casellarie casellario ADJ:pos+f+p
casellario casellario ADJ:pos+m+s
casellario casellario NOUN-M:s
casellarissima casellario ADJ:sup+f+s
casellarissime casellario ADJ:sup+f+p
casellarissimi casellario ADJ:sup+m+p
casellarissimo casellario ADJ:sup+m+s
```

Table 1: Examples of lexical entries in Morph-it!

We wrote a script for transforming the Morphit-it! textual representation into a hash table, with the lemmas used as the keys. This representation is more compact, since lemmas are not repeated as often as they have distinct full-form realizations, and our intermediate format is also giving a basic linguistic interpretation for the listed language data, allowing also marking explicitly ambiguities. An example of this intermediate format is given in Table 2.

```
"casco"  => {
    "NOUN"  => {
        "Infl_1"  => {
             "caschi" => "m+p",
        },
        "Infl_2"  => {
             "casco" => "m+s",
...
"casellario"  => {
    "NOUN"  => {
        "Infl_1"  => {
             "casellari" => "m+p",
        },
```

[1] http://www.trendminer-project.eu
[2] http://www.nooj4nlp.net/pages/italian.html. In the meantime, the author of the Italian resources for NooJ uploaded a much larger resource, which is for the time being available only in the compiled format, and therefore not usable for our experiment.
[3] http://dev.sslmit.unibo.it/linguistics/morph-it.php. See also (Zanchetta & Baroni, 2005)

```
        "Infl_2"  => {
             "casellario" => "m+s",
        },
    },
    "ADJ"  => {
        "Infl_1"  => {
             "casellari" => "pos+m+p",
        },
        "Infl_2"  => {
             "casellaria" => "pos+f+s",
    …
```

Table 2: Intermediate representation resulting from a transformation of Morph-it! Basic linguistic information is marked-up, contrary to the original format.

A second step consisted in computing the string differences between the lemma and the set of associated full-forms. This information is important in the case we want to use the lexicon in the context of a finite state machine (FST) platform, like this is the case in NooJ. The computed string differences are encoded in the form of morphological operations, that are performed by the FST engine in order to generate full-forms, as can be seen in Table 3. There, the value of the "fst" element in the first case tells that the engine processing the lemma "casco" has to go back one character (starting from the end of the lemma), delete the character that has been consumed, and add the letters "h" and "i" to the remaining of the string, and to mark the new word form with the inflectional values "m" and "p".

The "<E>" symbol in the second case specifies that no string operation is defined, and that the lemma and the full-form are thus identical, the latter being morphologically marked as "m" and "s".

```
"casco"  => {
        "NOUN"  => {
            "Infl_1"  => {
                 "fst" => "<B1>hi/+m+p",
                 "caschi" => "m+p",
            },
            "Infl_2"  => {
                 "fst" => "<E>/+m+s",
                 "casco" => "m+s",
            },
        ...
```

Table 3: Adding to the intermediate representation procedural information for the generation of full-forms.

At this level, we included thus some "operational" information to the lexicon, but this in a modular way. To use the LMF[4] terminology: we can consider this module describing operational information as being an extension of the core lexicon.

In NooJ, all those operational information can be encoded in inflectional paradigms, so that all the lemmas generating the same type of full-forms can share a unique paradigm, like for example the nominal lemmas "casco" and "carico" (and many other lemmas) are sharing the inflectional paradigm "NOUN_132", while the paradigm is specifying the concrete string operation (see Table 4)

[4] http://www.lexicalmarkupframework.org

```
carico,NOUN+FLX=NOUN_132
casco,NOUN+FLX=NOUN_132
------------
NOUN_132 = <B1>hi/+m+p + <E>/+m+s ;
```

Table 4: Sharing of a inflectional paradigm (NOUN_132) over various entries

The actual NooJ version of Morph-it contains all the main classes, and more specifically 6072 verbs, 17443 nouns and 9385 adjectives. The compiled inflected dictionary has 657062/12155 states and recognizes 442629 forms.

3. The Second Set of Resources: Entries in Wiktionary

As one could see from its description above, semantic information is not encoded in Morph-it! In order to palliate this lack of information, we searched for other freely available lexical sources, and we drove our attention to Witkionary. We didn't take Witkionary as our first source, assuming that the morpho-syntactic information encoded in Morph-it! is of a higher quality.

And in general, a drawback of the Wiktionary project is that the content of its lexical databases is formatted in a lightweight mark-up system commonly used in Wiki applications. This mark-up system is neither standardized nor very structure-oriented. To acerbate the situation, it is often applied in a considerably inconsistent manner, which makes extracting structured lexical information a really challenging task. But we consider Wiktionary still as a good source, also improving and in constant extension: We also discovered that the Italian Wiktionary[5] is one of the largest Wiktionary resources at all. Therefore we went into the task of porting the XML dump of this resource into our internal format. We extracted 29639 purely Italian entries; all encoded as lemma, and did not consider the full-form entries. An example of an entry we extract from the XML dump:

```
<page>
    <title>casco</title>
    <id>162499</id>
    <revision>
<id>1112205</id>

<timestamp>2011-10-29T05:27:52Z</timestamp>
      <contributor>
        <username>Ulisse</username>
        <id>18921</id>
      </contributor>
      <text
xml:space="preserve">{{in|it|noun}}
{{pn | w}} ''m sing '' {{linkp|caschi}}
# {{term|abbigliamento|it}} [[copricapo]]
difensivo atto a proteggere la [[testa]] da
urti
# particolare tipo di [[assicurazione]] che
copre anche i danni causati dal [[conducente]]
di un [[autoveicolo]] nei confronti del
medesimo
```

[5] http://it.wiktionary.org/wiki/Pagina_principale. The Italian Wiktionary (like other Wiktionaries) contains entries for many languages, but with all the associated information written in Italian: Therefore the use of the name "Italian Wiktionary".

```
# tipo di pettinatura femminile a forma di
{{pn}}
{{-hyph-}}
; cÃ  | sco
{{-etim-}}
dallo spagnolo [[casco]] di etimo incerto
{{-rel-}}
```

Table 5: An example of an entry in the Italian Wiktionary. Entries also have information about etymology, semantics, translation, etc., all of which can not be displayed here

We also transform this data representation onto a hash table, in order to allow comparisons with the data we already got from NooJ and Morph-it! Our main attention in this case is given to the acquisition of semantic information. An example of the transformation from the XML dump onto the machine readable hash table is given in Table 6.

```
"28033"
    => "casco" :: pos = noun {
              pl => caschi
              morph => m sing
             semantic[1] => [[copricapo]]
difensivo atto a proteggere la [[testa]] da
urti
             semantic[2] => particolare tipo
di [[assicurazione]] che copre anche i danni
causati dal    [[conducente]] di un
[[autoveicolo]] nei confronti del medesimo
             semantic[3] => tipo di
pettinatura femminile a forma di {{pn}}
              term[1] => abbigliamento
       synonym[1] => [[elmo]],
[[copricapo]], [[asciugacapelli]] ….
```

Table 6: Transformation of the XML dump of Wiktionary. Marking explicitly certain properties and re-organizing the distribution of information.

From this hash it is then easy to attach the semantic information to the already ported lexical entries from Morph-it! and to encode it also in the NooJ format[6], just extending slightly our script.

We mentioned above that encoding in Wiktionary is not always consistent. An example is given by the entries “blu”, which is associated to the semantic term “colore”, and “bianco” which is associated to the semantic term “colori”. There is a need for harmonization of the naming of the semantic categories. And further it would be better to use an Interlingua for naming related semantic categories.

[6] A reviewer of our submission very correctly noticed: things are not so easy, when one has to integrate semantic information in a lexicon that has already such information available. Decisions have to be taken, and it is not obvious how to deal with this aspect in an automated fashion. We will very soon attack this problem, also along the lines of very recently announced lexical resources, for English and German, which are integrating semantic information from various sources, like FrameNet and Wiktionary: UBY 1.0 - a large-scale lexical-semantic resource for natural language processing. See http://www.ukp.tu-darmstadt.de/data/lexical-resources/uby/ or (Gurevych et al., 2012)

Fortunately the Wikimedia foundation has foreseen such a system, so that all the language specific Wiktionaries can point to a unique set of descriptors (in English) for semantic categories[7], while keeping the origin of the pointing with the use of standardized language codes. Nevertheless not a lot of contributors do this.

We further decided then to test the extraction of Italian entries from the English Wiktionary. Since the representation format of the English lexicon is different from the Italian one, we had to adapt our extraction and transformation script. We can extract the high number of 463480 Italian entries, and we are in the process of reducing this number to the entries being in fact lemmas.

An example in our intermediate hash format of an Italian entry we extract from the XML dump of the English Wiktionary is shown in Table 7

```
"13837"
    => “spumante” :: pos = Adjective {
             morph = {{it-adj|spumant|e|i}}
             transl = EN =foaming
    }
    =>  “spumante” :: pos = Noun {
             morph =
{{it-noun|spumant|m|e|i}}
             transl = EN = sparkling wine
    }
    =>  “spumante” :: pos = Verb
             morph = {{present participle
of|[[spumare#Italian|spumare]]|lang=it}}
    }
    => Related Topics: * [[frizzante]]
    => Category: [[Category:en:Wines]]
```

Table 7: An example of an Italian entry in the English Wiktionary, in our intermediate harmonized format

The reader can get an idea of the disparity of information encodings using in different editions of the Wiktionary dictionaries, when looking at the entry in the Italian lexicon (Table 8).

```
"2141"
    => "spumante" :: pos = agg {
              morph => m
    }
    => "spumante" :: pos = noun {
              morph => m
              pl => spumanti
    }
```

Table 8: The entry “spumante” in the Italian Wiktionary to be compared to the entry in the English version in Table 7

Our actual work consists in mapping the tagset from the Italian Wiktionary to the tagset of the English Wiktionary, as the basis for merging both lexicons. At the same time we will add a link to the ISO Data Categories (http://www.isocat.org/) for ensuring the re-usability of the tagset.

On the basis of the semantic categorization proposed by Wiktionary and the mapping of these category descriptors to the categories suggested in the language specific

[7] http://en.wiktionary.org/wiki/Category:All_topics

Wiktionaries, we also started to extract a multilingual Wiktionary-Net, which could be combined with WordNet (http://wordnet.princeton.edu/)[8]. And last but not least we are establishing a machine readable translation dictionary (IT <-> EN).

4. Standardization

In this submission we stressed our need to get relatively quickly a large Italian lexicon running on the platform used in the project. And although can report on successful and promising work, we are aware that some solutions are still ad-hoc, since the approach we described was motivated first by pragmatic needs. We identified clearly the need to propose, beyond the actual implementation in the context of a specific platform, more standardized representations. We mentioned already LMF and we are in the process of porting the basic lexical information of our merged lexicon onto the LMF model. Additionally we will map the used tagset onto the ISO Data Categories, and include this information into the LMF representation.

An additional plan consist in making the extracted and integrated lexical information in the context of the Linked Open Data initiatives active in the field of language resources. Some works in this direction have been presented at the recent Workshop "Linked Data in Linguistics"[9]. In this context a main effort consists in publishing linguistic data using W3C standards like RDF and SKOS[10]. An example of such work is given in (McCrae et. al, 2012).

But we first started with the porting of our lexical information onto TEI (P5), since some work as already been done in this respect at ICLTT, also in order to make our work easily available to the Digital Humanities community, which is making an heaving use of text annotation properties introduced in TEI. For now, the German Wiktionary has been converted into TEI (P5)[11], also making use of standardized feature structures (a joint work by ISO and TEI standardization bodies), especially for the representation of morpho-syntactic features, following the recommendation of ISO-MAF, which is not yet an established standard. The user can access the data both via a GUI and via a XML download[12]. We plan to achieve the same results for the merged Italian lexicon as the next step of our work, after we merged the entries from both the English and the Italian Wiktionary resources.

[8] As mentioned in footnote 6, we will have detailed look at the recent developments described in the work of (Gurevych et al., 2012)

[9] http://ldl2012.lod2.eu/

[10] See both http://www.w3.org/RDF/ and http://www.w3.org/2004/02/skos/

[11] Result of this work can be seen at: http://corpus3.aac.ac.at/showcase/index.php/wiktionaryconvertor

[12] See http://www.tei-c.org/Guidelines/P5/

5. Conclusion

We presented an approach for integrating various Italian resources, in the context of concrete needs. Beyond this we identified ways for publishing results of our work in standardized representations that can be used by the NLP community at large. We will establish concrete cooperation with initiatives like UBY (in the context of ISO standards) or LDL (in the context of W3C standards).

6. Acknowledgements

This work has been partly supported by R&D project TrendMiner, which is co-funded by the European Commission under the contract nr. 287863.

7. References

Gurevych, I., Eckle-Kohler, J., Hartmann, S., Matuschek, M., Meyer, C.M., Wirth, C. (2012). A Large-Scale Unified Lexical-Semantic Resource. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics.* Avignon.

Krizhanovsky, A. (2010). The comparison of Wiktionary thesauri transformed into the machine-readable format. (http://arxiv.org/abs/1006.5040)

Krizhanovsky, A., Lin F (2009). Related terms search based on WordNet / Wiktionary and its application in ontology matching. In: Proceedings of the 11th Russian conference on Digital Libraries (RCDL 2009).

McCrae, J., Montiel-Ponsoda, E., Cimiano, P. (2012). IntegratingWordNet andWiktionary with Lemon. In Proceedings of the Workshop **"***Linked Data in Linguistics: Representing and Connecting Language Data and Language Metadata***"** (LDL).

Meyer, C.M., Gurevych, I. (2010): Worth its Weight in Gold or yet another resource – a comparative study of Wiktionary, OpenThesaurus and Germanet. In: *Proceedings of the 11th International conference on Intelligent Text Processing and Computational Linguistics.* Iasi (Romania) 2010: pp. 38-49

Moerth, K., Declerck, T., Lendvai, P., Váradi, T. (2011)**:** Accessing Multilingual Data on the Web for the Semantic Annotation of Cultural Heritage Texts. In: *Proceedings of the 2nd International Workshop on the Multilingual Semantic Web* (Bonn 2011): 80-85.

Navarro, E., Sajous, F., Gaume, B., Prévot, L., Hsieh, S.-K., Kuo, T.-Y., Magistry, P., Huang, C.-R. (2009). Wiktionary and NLP: Improving synonymy networks. In: *Proceedings of the 2009 Workshop on Peoples's Web Meets NLP, ACL-IJCNLP*. Singapore: pp. 19-27.

Zanchetta, E., Baroni, M. (2005). Morph-it! A free corpus-based morphological resource for the Italian language. In *Proceedings of Corpus Linguistics 2005*, University of Birmingham, Birmingham, UK.

Zesch T., Mueller C., Gurevych I. (2008a). Extracting lexical semantic knowledge from Wikipedia and Wiktionary. In: *Proceedings of the Conference on Language Resources and Evaluation*. LREC 2008.

Zesch T., Mueller C., Gurevych I. (2008b). Using Wiktionary for computing semantic relatedness. In: *Proceedings of 23rd AAAI conference on Artificial Intelligenc*

Towards An Universal Automatic Corpus Format Interpreter Solution

Radu Simionescu[1], Dan Cristea[1,2]
[1] Faculty of Computer Science, "Alexandru Ioan Cuza" University of Iași
[2] Institute for Computer Science, Romanian Academy, the Iași branch
E-mail: radu.simionescu@info.uaic.ro, dcristea@info.uaic.ro

Abstract

The process of building a processing chain is always cumbersome because, in most cases, the NLP tools making up a chain do not match with respect to the input/output format. Convertors are required to transform the output format of a tool to the input format of the next one in the chain, in order to assure correct communication between modules. The work presented in this paper proposes a solution for automatic format interpretation of annotated corpora. A mechanism of this kind would finally make possible the automatic generation of processing architectures. ALPE is a system designed to compute processing workflows, given a sample of an input format and a description of an intended output format.

Keywords: linguistically annotated files, annotation schemes, natural language processing chains, ALPE

1. Introduction

The field of Natural Language Processing (NLP) has seen significant developments in terms of resource accessibility over the later years. Given the constant rich flow of both theoretical and practical innovations brought to this research field, many corpora were built, giving birth to many different annotation formats. Even though there are many comprehensive annotation standards, the global cloud of natural language corpora contains many unknown/nonstandard annotation formats.

The general trend in CL is that both theoretical and practical developments are deeply anchored on linguistic data. The research in CL and NLP will always go on with an arrow head of new, unseen yet, annotation conventions, because researchers will invent them and use as soon as the needs appear. The research does not have time to wait the apparition of standards. Very often, the object under investigation is spotted on the text on a bed of annotations that describe already known phenomena, therefore using annotation conventions already accepted by the scientific community. As such, an annotated linguistic corpus often includes three categories of linguistic markings:

- old, standardised: these describe linguistic phenomena which, having acquired a large acceptance from the community, have already been in the focus of a standardisation process;
- old, non-standardised: these describe known linguistic phenomena, but for which still subsist a diversity of notations;
- new, non-standardised: these describe the new phenomena under investigation, for which annotations have been freshly proposed by the authors of the corpus.

For the reasons explained, innovative research in NLP will always use non-standard notations and there is a continuous process of inventing new annotation formats for linguistic phenomena previously unseen.

By looking for the first time in a file which includes annotation markings, a computational linguist is however capable to distinguish the meaning of those markings. This paper addresses the issue of deciphering the semantics of a new, previously unseen, annotation convention of a text file, by mimicking a human expert behaviour.

2. Previous work

Projects such as CLARIN[1], FLaReNet[2] and METANET[3], among others, intend to offer both developers and users of language resources and tools a management solution for the growing set of resources available. The primary objectives of these projects are to provide reusability in new contexts for existing resources and to guarantee interconnectivity of newly developed resources and tools. An easy widening of the original setting of usage means a multiplication of the visibility of a tool and, finally, of the productivity of the research activity. In terms of managing linguistic processing tools, previous efforts lead to the development of linguistic processing meta-systems, most significant ones being GATE [4] (Cunningham et al., 2002) and UIMA [5] (Ferrucci and Lally, 2004). Both systems allow access to a set of independently developed NLP tools, integrated into an environment offering means to create and use processing chains adding linguistic metadata to an input corpus. They allow integration of new tools and new processing chains, but these functionalities require programming experience.

ALPE (Automated Linguistic Processing Environment) (Pistol and Cristea, 2009) (Pistol, 2011) is a processing environment which, making use of annotation schemas arranged hierarchically in a hyper-graph, is able to automatically compute workflows made up of disparate processing resources. The model ALPE is based upon, called the **Formats and Modules Hierarchy** (FMH) is designed to help users to build complex processing

[1] www.clarin.eu/
[2] www.flarenet.eu/
[3] www.meta-net.eu/
[4] www.gate.ac.uk/
[5] www.research.ibm.com/UIMA/

architectures, by involving minimum of expert skills.
The FMH is a hyper-graph whose nodes represent descriptions of annotation formats. Its edges represent relations of the following categories: subsumption (Cristea and Butnariu, 2004), conversion and reduction. The subsumption relation is defined as follows: a format A subsumes a format B (A⊆sB) if all elements (annotations) which can be specified by A can also be specified using the B format. In addition, there can be elements which can be described using B, but not A.
ALPE is intended to offer solutions of building easily new processing architectures, by combining existent tools, each of them performing an elementary operation. If we bound tools to the edges of the hyper-graph, itself build by exploiting only the formats, then new architectures can result at the end of a navigation process within the graph. However, the ALPE philosophy has little applicability in an environment characterised by a great diversity of formats of the files to be processed or of the input/output conventions of the processing modules.

3. Our model

The work described in this paper adopts a simplified version of the FMH model, in which the format descriptions are separated from the modules hierarchy. In the simplified FMH model a hierarchy H must have access to an ontology O_h of annotations which consists of abstract descriptions of annotations used in NLP. Such an ontology covers various aspects, including the dependencies between annotation types (e.g., an annotation for a part of speech requires a notation of tokens). The nodes of a reduced FMH are simple sets of such annotation classes.
Such a model can be used to set up a network of input sets of annotation types which are required in various NLP processes (associated with edges of the hierarchy) and their resulting set of annotation types. This approach requires the annotation classes to be defined only once, in a unified manner, for all the nodes of a hierarchy H – that is why any H must have an ontology O_h which describes the types of annotations that it can handle.
The **Corpus-Format Interpreter Component** (CFIC) is used to help a semi-automatic „*interpretation*" of a given corpus format, even previously unseen (as long as it adopts a commonly used structure, like XML, tabular data, raw text, or others). CFIC is responsible for the following tasks:

1) to „guess" the significance of annotations existed in a given corpus C and, as a result, to compute the ontology of annotations that it contains, O_c;
2) to establish the position of the set of annotations present in C relative to a hierarchy H. In general, this implies adding a new node to H, a node which contains all the classes of O_c, and establishing its relations with the other nodes of the existent hierarchy. Prior to such a computation, it is required that pairs of equivalent annotation classes are found by matching O_c against O_H;
3) once the place of the set of formats the user's corpus C includes was established with respect to H, the user may enter the annotation requirements of the corpus he/she wants to be obtained out of C (as a result of applying a processing workflow). At this moment an ALPE-like system can automatically compute out of the hierarchy the possible workflows needed;
4) to create the annotation instances in an internal format which can be fed to the processing modules associated with the edges of a hierarchy, when a workflow processing is requested.

Another component which is part of this model is the **Corpus Writer Component** (CWC) which is responsible for outputting annotations to disk, in a large variety of formats, including custom formats based on output samples.
Both CFIC and the CWC will be used only once for a computed workflow, at the start node for reading the input and at the destination node for outputting. The communication between modules is done using the internal format.
Separating the formats from the modules has some consequences. First of all, the conversion relations are no longer necessary in this simplified model. This is a good thing – in a real world situation there are few NLP components which match the format from one module's output to the next module's input. For most of the cases such conversion relations should be established, which means that modules that actually perform the needed conversions should be implemented ad-hoc. Of course, getting rid of the conversion relations does not mean that conversions are not necessary anymore. They are, but the conversion is taken over by CFIC, which uses an internal format as pivot.
Having the freedom to define an internal format makes also possible to choose the most convenient approach. For example, an offset based internal format solves the problem of merging annotations: merging annotations becomes only a matter of concatenating two or more sets of annotation instances together; problems like intersecting inline XML annotations are gone. Even more, using UIMA's internal format, this approach would allow for any UIMA integrated NLP tool to be easily made compatible with an ALPE hierarchy system.

4. Defining an ontology of annotations

This section presents a possible manner of describing an ontology of annotation conventions used in NLP. To keep things simple, we will describe these conventions in XML, instead of using advanced RDF-OWL features. Please consider the actual XML format less important – the key aspect here is the data model.
For the purpose of this paper an ontology of NLP annotations may contain only three types of classes:

1) **offset** – these types of annotations are used mostly for positioning tokens at character level offsets in a

source text; any instance of an annotation element of type *offset* will contain a start value offset and an end value offset, both relative to the source text.

```
<annotType name="tok" type="offset"/>
```

This is an example of definition of an annotation class, named "tok", of type *offset*. Any instance of *tok* must contain two offset values.

2) **attachAttrib** – these annotations are used for attaching an attribute to another annotation

```
<annotType name="pos" type="attachAttrib"
annotClass="tok" valType="string"/>
<annotType name="lemma" type="attachAttrib"
annotClass="tok" valType="string"/>
```

"annotClass" specifies the class of annotations to which an attribute of type string can be attached.

Any instance of an annotation of the type *attachAttrib* type will identify the instance of the class that it attaches to and the value of the attribute that it attaches.

"valType" defines the type of the attribute which is to be attached and can be one of the following:

- string
- string{'list','of','possible','values'}
- ref(annotationClassName) – references to instances of the annotationClassName class (to refer to other annotations, like in the case of syntactic dependencies).

```
<annotType name="synHead" type="attachAttrib"
annotClass="tok" valType="ref(tok)"/>
<annotType name="synRel" type="attachAttrib"
annotClass="tok" valType="string{'list' 'of'
'possible' 'values'}"/>
```

3) **list** – signals direct parentage of annotation instances

```
<annotType name="NP" type="list" annotClass="tok
PP"/>
<annotType name="PP" type="list" annotClass="tok
NP"/>
```

"annotClass" specifies the type of the annotation instances which can be contained in the list. An instance of a list class will specify the annotation instances which it contains.

Since a list annotation is an annotation class, it is permitted to attach attributes to a list, for annotating named entities, for instance:

```
<annotType name="NE" type="list" annotClass
="tok"/>
<annotType name="neType" type="attachAttrib"
annotClass ="NE"
valType="string{'city', 'country', 'person', 'c
ompany', 'product'}"/>
```

Let's consider the following annotations ontology:

```
<annotType name="tok" type="offset"/>
<annotType name="lemma" type="attachAttrib"
annotClass ="tok" valType="string"/>
<annotType name="pos" type="attachAttrib"
annotClass="tok" valType="string"/>
<annotType name="synHead" type="attachAttrib"
annotClass="tok" valType="ref(tok)"/>
<annotType name="synRel" type="attachAttrib"
annotClass="tok" valType="string"/>
<annotType name="NP" type="list" annotClass
="tok"/>
<annotType name="NE" type="list" annotClass
="tok"/>
<annotType name="neType" type="attachAttrib"
annotClass ="NE"
valType="string{'city','country','person','comp
```

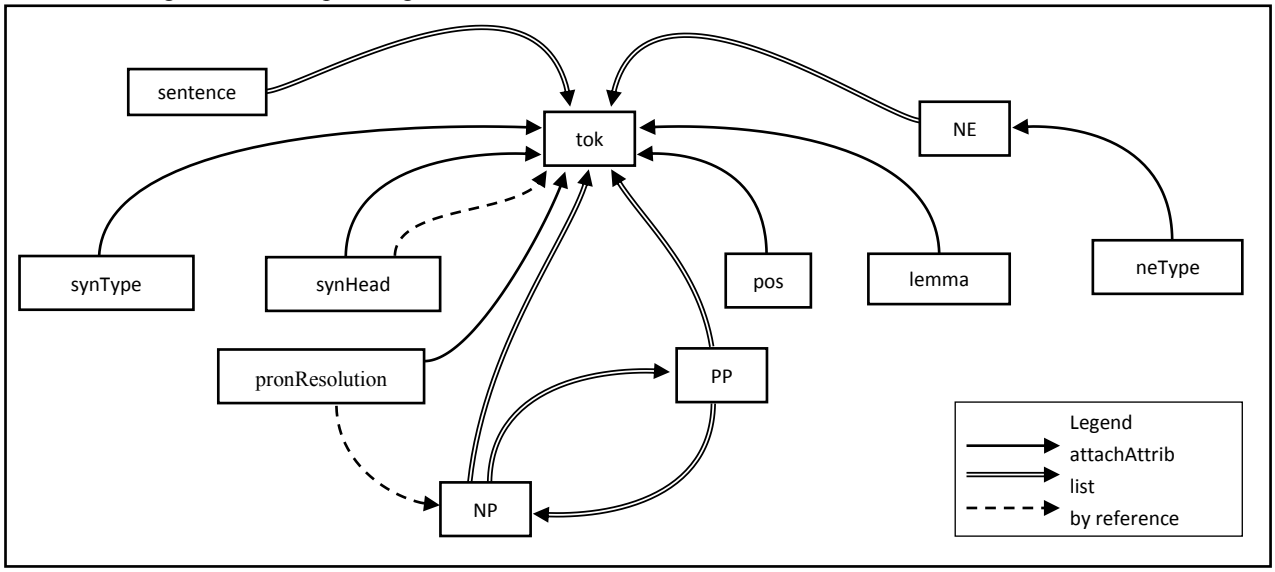

Figure 1

```
any'}"/>
<annotType name="sentence" type="list" annotClass
="tok"/>
<annotType name="pronResolution"
type="attachAttrib" annotClass="tok"
valType="ref(NP)"/>
```

(The *pronResolution* annotation class is used for specifying the noun phrase which a pronoun refers to)

One could easily define all the annotation classes to be of type *offset* but this would make the dependencies between them difficult to manage. The idea, therefore, is to glue to the character level the minimum possible of annotation classes and then attach other annotation classes on their strict inferior. In this example, the *tok* annotation is the only one which directly references the source text, using offsets. All the other annotations are based on the *tok* annotation, by grouping tokens together and by attaching attributes.

Figure 1 shows the dependency graph for the ontology given as example above.

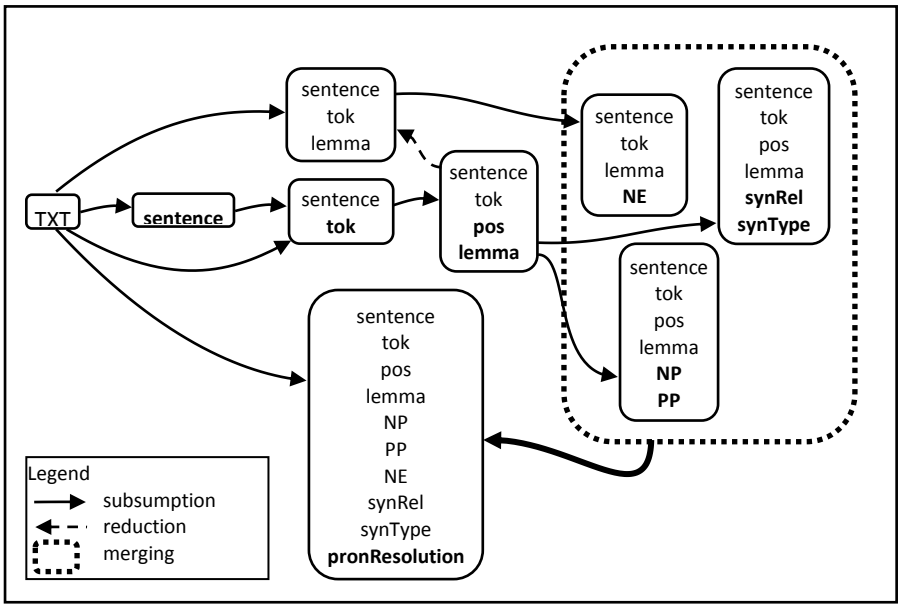

Figure 2

Figure 2 shows an example of a reduced FM hierarchy which uses the ontology from Figure 1. As explained earlier, a node in the simplified FMH model is just a set of annotation classes which are identified by their names and are defined only once in the ontology associated with the hierarchy.

5. The Corpus Format Interpreter Component

When **interpreting a given corpus** *C*, the first step of the CFIC is to build an ontology O_C of annotation classes that characterise *C*. These classes of annotations will be named arbitrarily. First, the type of annotation format is identified (XML, tabular data, continuous text, etc.), then the offset annotations are identified. Next, the CFIC analyses the various annotation markers found in *C* and creates definitions of annotations based on their structure, attribute values, etc. This process depends very much on the type of annotation format (XML, tabular etc).

Like stated before, the CFIC is also responsible for establishing the relations of the annotations belonging to C with an already existent FMH hierarchy *H* (which has access to an annotation ontology O_H). The CFIC merges O_C and O_H together into O_{C+H} by finding equivalent annotation definitions. The idea here is to match a smaller graph (as given by O_c) onto a larger graph (O_h). This matching process is still work in progress. Any clues that contribute to this match are used, including matching of annotation markers' attribute values, libraries of names of elements and attributes (the name space), etc.

The position of the annotation schema of *C* is determined relative to *H*, by classification (Cristea, Butnariu, 2004; Pistol, 2011). This process might require the extension of *H*. Let *H'* be a hierarchy which contains all the nodes and structure from *H* but which uses O_{C+H} instead of O_C. *H'* also contains an additional new node *N*, inserted into *H'* and containing all the annotation classes contained in O_H.

To find the position of *N* (and implicitly of the information in *C*) relative to *H'*, it is required to find all the maximal nodes *X* which subsume *N* ($\forall X$, there is no $X' \neq X \neq N$ in *H'* such that $X \subseteq_s X' \subseteq_s N$) and to construct simplification relations between *N* and these nodes.

If all the annotations from O_C have equivalents in O_H and if *H* contains a node which is defined by a set of all the annotation classes in O_C then the extension of *H* is redundant. In this case, the information in *C* has a very clear spot in *H*.

The paragraphs below present a **simulation** of this process on an example. Let's consider the situation were the user wants to know what types of annotations are contained in a corpus *C* (see a sample below) and to determine what workflows could the system run, considering the hierarchy *H* from Figure 2 and O_H from Figure 1.

```
…
<p id="15">
<s id="15.6">
     <clause id="15.6.I">
<w id="15.6.1" lemma="it" ana="#Pp3ns">It</w>
<w id="15.6.2" lemma="be" ana="#Vmis3s">was</w>
<w id="15.6.3" lemma="a" ana="#Di">a</w>
<w id="15.6.4" lemma="bright"
ana="#Af">bright</w>
<w id="15.6.5" lemma="cold" ana="#Afp">cold</w>
<w id="15.6.6" lemma="day" ana="#Ncns">day</w>
<w id="15.6.7" lemma="in" ana="#Sp">in</w>
<w id="15.6.8" lemma="April"
```

```
ana="#Ncns">April</w>
</clause>
     <clause id="15.6.II">
<w id="15.6.9">,</c>
<w id="15.6.10" lemma="and" ana="#Cc-n">and</w>
<w id="15.6.11" lemma="the" ana="#Dd">the</w>
<w id="15.6.12" lemma="clock"
ana="#Ncnp">clocks</w>
<w id="15.6.13" lemma="be" ana="#Vais-p">were</w>
<w id="15.6.14" lemma="strike"
ana="#Vmpp">striking</w>
<w id="15.6.15" lemma="thirteen"
ana="#Mc">thirteen</w>
<w id="15.6.16">.</c>
</clause>
</s>
…
<s>…</s>
…</p>…
```

Given *C* for interpretation, the CFIC would build its annotation ontology O_C:

```
<annotType name="tok" type="offset"/>
<annotType   name="lemma"   type="attachAttrib"
annotClass="tok" valType="string"/>
<annotType    name="ana"    type="attachAttrib"
annotClass="tok" valType="string"/>
<annotType name="clause" type="list" annotClass
="tok"/>
<annotType   name="s"   type="list"   annotClass
="clause"/>
<annotType name="p" type="list" annotClass ="s"/>
```

Figure 3 shows the dependency graph of O_C.

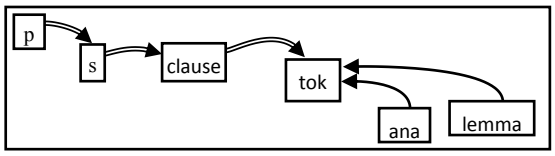

Figure 3

Next, the annotation equivalences between O_C and O_H are established and O_{C+H} is built. The equivalent annotation classes pairs are tok->tok, ana->pos, lemma->lemma and s->sentence. This paper does not cover details of the process responsible for detecting equivalences. But it does explain briefly how could the equivalences from this particular example be established by an automated process:

- The equivalence of the *tok* annotation classes is based on the fact that *tok* is the only annotation of the *offset* type and on the fact that most hierarchies only have one annotation class of this type. Also, the average of the length of the instances is close to a certain value (used by a heuristic), and there is identity of names (another heuristic).
- Clues that support the equivalences *ana->pos* and *lemma->lemma* are: the values of the instances extracted from *C*, the number of distinct values found in the instances of *C*, the fact that each token has an *ana* and a *lemma*, same *lemma* attribute in both classes.
- The *s->sentence* equivalence is sustained by the average number of tokens found in a sentence. Even though *s* annotations are directly dependent on the clause annotation, the system also considers the indirect dependency.

Figure 4 shows O_{C+H} with the nodes merged from O_C emphasized.

Finally, Figure 5 shows the position of the annotations in *C* relative to *H* by establishing a reduction relation.

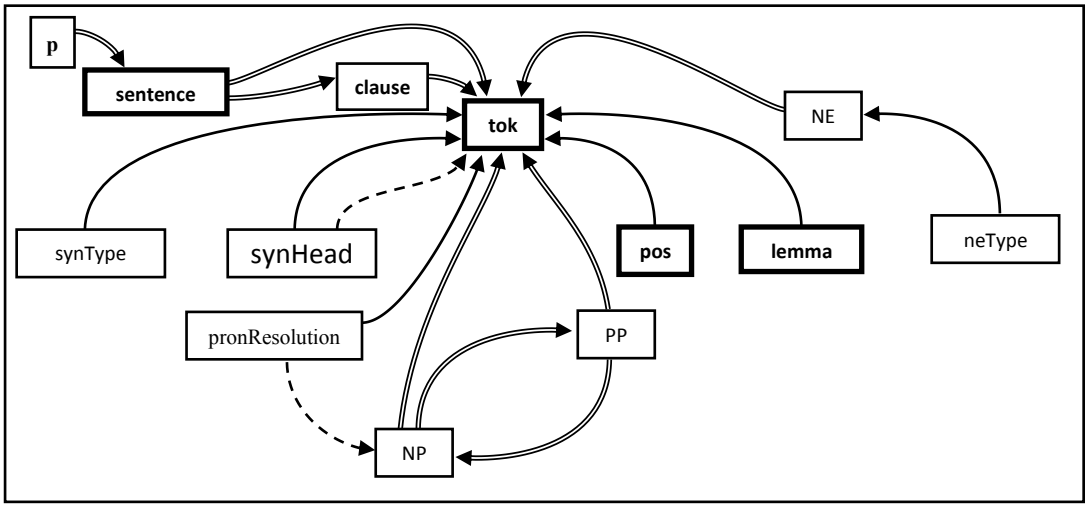

Figure 4

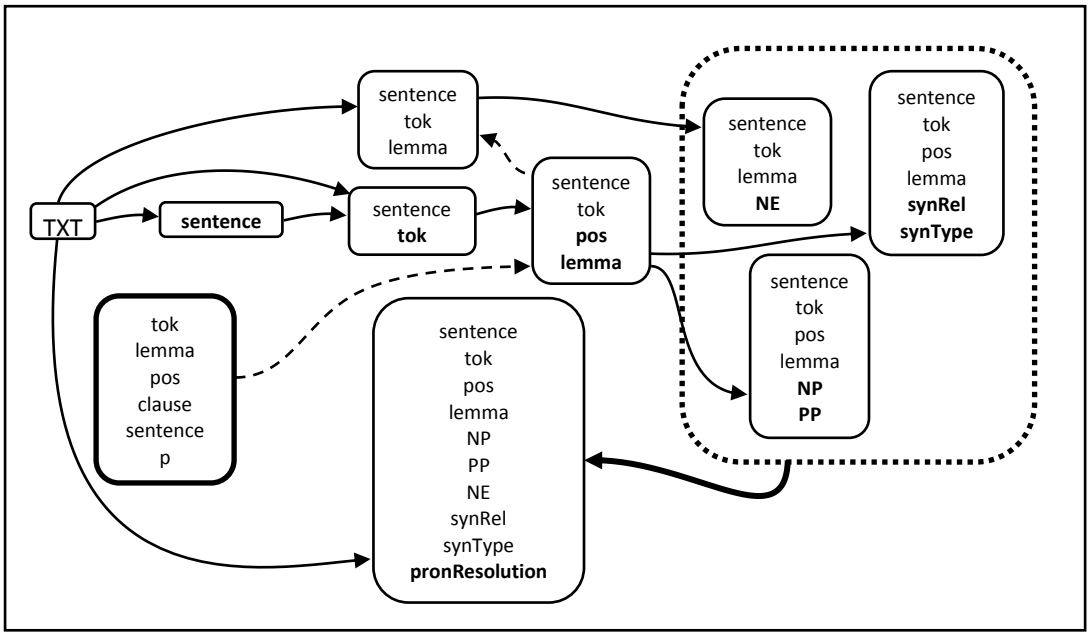

Figure 5

6. Conclusion

Even though the goal of automatically interpreting an annotation format seems yet being distant, there are important clues suggesting that it can be achieved. A trained human can easily interpret most of the previously unseen annotation formats. Like in many artificial intelligence fields, this goal can be achieved by observing the human's mind behaviour when reading a new file, and formalizing this behaviour could result in a system which its performance.

This paper outlines a pathway towards automatic interpretation of previously unseen annotation formats. Finding a position in an FHM hierarchy of any corpus, would make possible for it to be processed at once, with no more energy being wasted on converting the corpus in a particular format or computing the process chain. A system which can offer such functionality will represent the next level of performance and accessibility to resources. And this would have a great impact on the entire field of NLP.

7. Acknowledgements

The research conducted in this article was partially supported by the ICT-PSP projects MetaNet4U and Atlas.

8. References

Cristea, Dan, and C Butnariu. *Hierarchical XML.* Lisbon: In Proceedings of the LREC 2004 Workshop on XML-Based Richly Annotated Corpora, 2004.

Cunningham H., Maynard D., Bontcheva K., Tablan V. GATE: A framework and graphical development environment for robust NLP tools and applications. In Proceedings of the 40th Anniversary Meeting of the ACL (ACL'02). Philadelphia, US, 2002.

Ferrucci D. and Lally A. UIMA: an architectural approach to unstructured information processing in the corporate research environment, Natural Language Engineering 10, No. 3-4, 327-348, 2004.

Pistol, I.C. *The Automated Processing of Natural Language.* Phd Thesis, Iași: "Alexandru Ioan Cuza" University of Iași, 2011.

Pistol, I.C., and Dan Cristea. *Managing Metadata Variability.* Milan: Proceedings of the 6th International Workshop on Natural Language Processing and Cognitive Science - NLPCS 2009, p111-116, 2009.