Stefano FEDERICI, Par.O.La s.a.s.

An efficient algorithm for the automatic building of a lexicon from textual corpora

Abstract

The LE-2111 SPARKLE (Shallow Parsing and Knowledge extraction for Language Engineering) project is aimed at the automatic extraction of lexical and semantic information from textual corpora in order to improve the performances of NLP systems. In this paper we describe an algorithm for the extraction of subcategorization patterns for Italian verbs. The extraction procedure is carried out on the basis of an efficient and accurate analogy-based engine and pre- and post-filters based on simple linguistic constraints. Despite the simplicity of the analogy-based algorithm the amount of lost information is negligible, and precision and recall over a set of hand-crafted subcategorization patterns (namely those produced within the LE PAROLE project) is fairly high.

Keywords: linguistic knowledge extraction, lexicon building, finite state automata, chunking

1. The Sparkle Project

Shallow parsers have been developed to build partial syntactic analyses of sentences without the aim of building a complete syntactic structure in which all syntactic dependencies are solved (Briscoe 1997). A shallow parse of a sentence is a flat structure in which the main syntactic building blocks of the sentence are identified. Dependency links among these building blocks are only solved when the necessary cues are available, thus leaving a considerable amount of ambiguity in the analysed sentence.

In the first phase of the Sparkle Project (Sparkle Project 1995), corpora of millions of words in different languages (English, German and Italian) have been analysed by shallow parsers that do not rely on syntactic knowledge (Briscoe et al. 1996). In the second phase of the project, lexical syntactic knowledge has been acquired from the shallow-parsed texts in order to build a lexicon of subcategorization structures for English, German and Italian verbs. The third phase is aimed at enhancing the shallow parsers used in the first phase by using the information extracted in the previous phase in order to solve some of the dependencies left unsolved and to build less ambiguous syntactic structures. The final phase will test the improvements of the lexicalized parsers on a test bed of 200 sentences randomly extracted from newspapers. The usefulness of the extracted lexical knowledge will be also tested on some NLP applications, namely Machine Translation, Information Retrieval and Speech Understanding.

2. Extracting subcategorization patterns of Italian verbs

The procedure used to extract subcategorization patterns of Italian consists of four main steps (Calzolari et al. 1997, Federici et al. 1997):

1. Extraction: contexts of verbal headwords are extracted from shallow-parsed texts. Some simple linguistic heuristics are then used to prune out some contexts that potentially contain misleading information (see section 2.1).

2. Carving: the extracted contexts are shortened to the left and right of the headword (Calzolari et al. 1997). For example, an isolated adjective immediately preceeding a verbal headword is cut out.

3. Coring: cores (potential subcategorization patterns) are extracted from carved contexts by comparing the latter pairwise.

4. Typing: eventually, cores are typed in order to identify those cores that do not represent subcategorization patterns. Cores that potentially contain misleading information are then pruned out (see section 2.3).

Only the remaining cores that have not been typed in Step 4 are then output as a result of the acquisition procedure. In this paper we will concentrate on Step 3, the analogy-based algorithm for the acquisition of subcategorization patterns. We will start with a brief description of the shallow parser used to analyse the texts.

2.1. Extracting contexts from text

Contexts of verbal headwords are extracted from a text pre-analysed by a shallow parser, the Chunker system, that hinges only on information about i) the part of speech of each word, and ii) closed classes (e.g. predeterminers, auxiliaries) (Federici et al. 1996). The chunker is made up of eight Finite State Automata extrinsecally ordered to parse 10 possible different structures called chunks. An example of chunked sentence is the following: [N_C the red flag] [FV_C has been given] [P_C to the winning team], where a Nominal Chunk (N_C) is followed by a Finite Verb Chunk (FV_C) and a Prepositional Chunk (P_C). This simple analysis has the desired features: the words stick together and what emerges is that the verb *give* in its passive form can be preceded by a N_C, and can be followed by a P_C introduced by *to*. Despite its simplicity, the correctness of the chunking process —98% of correct chunks (Federici et al. 1996)— guarantees a sufficient degree of reliability for the extraction task.

Contexts extracted from chunked texts contain a headword chunk (the chunk that contains the verb we are interested in) followed by the sequence of the surrounding chunks (contextual chunks). For example the context extracted from the previous sentence would be represented as in Table 1.

HEAD:	[FV_C has been given]
CONT:	[N_C the red flag]
CONT:	[P_C to the winning team].

Table 1: The chunked sentence *the red flag has been given to the winning team*

Linguistically-based heuristics are then used to prune out the contexts that potentially contain some misleading chunks. To give but an example, N_Cs that contain time expression could be confused with the subject or the object of a sentence (e.g. *this evening* in the sentence *I will come this evening*). This procedure does not prune out a great amount of contexts, on

average less than 3.73% of extracted contexts is not used in the core extraction procedure (Calzolari et al. 1997, Federici et al. 1997).

Internal storing of extracted contexts is performed by eliminating all redundancies and by indexing the contexts by chunk-type using a tree-like index. This allows the system to speed up the extraction and coring steps.

2.2. Extracting cores

The comparison of contexts is aimed at finding what any pair shares, under the assumption that a subcategorization pattern of a verb is what two or more contexts of that verb have in common, the variable part of the contexts being ascribed to syntactic modification of various sorts. The common part of two contexts is what we call core.

After shortening the extracted contexts in the carving step, cores are extracted from the set of contexts by comparing pair. Comparison is performed by means of an analogy-based algorithm, a specialized version of a general algorithm for the automatic acquisition of linguistic knowledge (Federici & Pirrelli 1997). The most significant part of the algorithm is the Mapping Function (MF) which governs the way generalizations are derived from examples. MF is flexible enough to allow the system to find relevant similarities in very different contexts. For each context pair, starting from the headword chunk, MF extracts all chunks that share some desired features[1] (defined in an external file). There are two distinct sets of features (Table 2) for the headword and contextual chunks.[2] The settings shown in Table 2 are those used to obtain the results described in section 3.

	feature	headword chunk	contextual chunks
CC	(Chunk Category: e.g. N_C, P_C)	no	yes
PREP	(Preposition)	no	yes
CAUS	(has Causative verb)	yes	no
CLIT	(has Clitic pronoun)	yes	no
AUX	(has Auxilary verb)	yes	yes
INTRO	(has Introducer)	no	no
MODIF	(has adjectival or adverbial Modifier)	not relevant[3]	no
SUBCONJ	(has Subordinating Conjunction)	no	yes
LEMMA	(Lemma)	yes	no
POS	(Part of Speech)	yes	no
MORPH	(Morphosyntactic Features)	no	no
DIRECTION	(Direction (left/right) wrt the Headword Chunk)	not relevant	yes
POSITION	(Position wrt the Headword Chunk)	not relevant	yes
GFUNC	(Grammatical Function)	not relevant	yes

Table 2: The feature sets for headword and contextual chunks and their settings

Each context is likely to overlap with more than one other context: out of all resulting cores MF selects only one core, namely the one exhibiting the largest number of common features. For example, starting from contexts 1 and 2 in Table 3 (for the sake of clarity, the corresponding contextual chunks are aligned) we obtain the core in Table 4.

Context 1	Context 2
HEAD: [FV_C has been given] CONT: [ADV_C yesterday] CONT: [N_C the red flag] CONT: [P_C to the winning team]	HEAD: [FV_C has been given] CONT: [N_C the letter] CONT: [P_C to the man] CONT: [P_C in the room]

Table 3: Two chunked contexts

Core
HEAD: [FV_C has been given] CONT: [N_C] CONT: [P_C to]

Table 4: The core extracted from context1 and context2

MF can be guided through a definition of how the mapping is to be performed. Indeed, the algorithm can extract the cores either in a continuous or discontinuous way. By setting the parameter 'Continuous Match', the user can force the algorithm to extract a core if and only if the headword and contextual chunks that share the desired features are adjacent to one another.

If 'Continuous Match' is disabled, MF is allowed to compare two contexts without having to take into account the relative position of contextual chunks. This can be useful when a particular verb has only a few occurrences in the corpus, thus allowing us to get around the problem of sparse data. In any case, by setting on the DIRECTION and POSITION features and the 'Continuous Match' flag a more restrictive comparison can be performed. These latter settings reduce significantly the amount of extracted cores.

2.3. Typing cores

When the extraction procedure has come to an end, the typing step is carried out. The resulting cores are supposed to contain only complements that depend on the headword and can thus be the potential subcategorization patterns of the verb we are interested in.

Core typing is carried out by comparing to one another the extracted cores. Cores are typed (see Table 5) to identify those that potentially contain misleading information.

Noisy	(Cores that contain contextual information that could not depend on the headword)
Subsumed	(Cores containing subcategorization information contained in other selected cores)
Low-frequency	(Cores that match less than a certain amount of contexts)
Bare	(Cores that have no contextual information)

Table 5: Core Types

Let us briefly illustrate the mechanism used for core typing. Noisy cores, that is cores containing contextual information that could not depend on the headword, are identified by checking if the complements that are contained in a potential subcategorization pattern can be also found in isolation in another core. This criterion is a very weak form of optionality check. A subsumption criterion that is reminiscent of the set inclusion test is then applied to identify those cores that contain subcategorization information contained in other (bigger) cores. The identification of cores that have a low-frequency of occurrence (low frequency cores) and that do not have contextual information (bare cores) is rather straightforward.

3. Analysis of preliminary results

In the current phase of the Sparkle project, Italian subcategorization patterns have been extracted for a set of 30 verbs that have been selected for the representativeness of their subcategorization patterns. The list of verbs and the obtained results are shown in Tables 6-12. Results of the acquisition procedure are obtained by applying the described algorithm to a corpus of about one million words. The extraction algorithm is efficient and economical because the extracted subcategorization information meets very restrictive conditions. In Table 7 the number and the percentage of extracted and selected cores (potential subcategorization patterns) resulting at the end of the extraction procedure is given with respect to the total number of contexts in the acquisition corpus. In the average, these percentages are 30% and 7% and they never overcome the thresholds of 50% and 29% respectively. This clearly shows that the procedure is not tailored on some well-behaving verbs.

verb	contexts	extracted cores		selected cores	
		number	percentage wrt context	number	percentage wrt contexts
aggiungere 'add'	261	62	24%	7	3%
aggiustare 'fix'	7	3	43%	2	29%
aiutare 'help'	99	25	25%	6	6%
aspettare 'expect'	106	39	37%	7	7%
cambiare 'change'	182	46	25%	13	7%
caricare 'charge'	23	7	30%	1	4%
causare 'cause'	35	14	40%	3	9%
chiamare 'call'	90	34	38%	5	6%
chiedere 'ask'	316	85	27%	16	5%
cominciare 'begin'	163	33	20%	9	6%
concordare 'agree'	58	29	50%	7	12%
considerare 'consider'	235	87	37%	11	5%
costruire 'build'	55	18	33%	1	2%
credere 'believe'	156	49	31%	7	4%
dare 'give'	1088	48	4%	41	4%
decidere 'decide'	393	89	23%	15	4%
fornire 'provide'	154	56	36%	9	6%
muovere 'move'	119	34	29%	9	8%
oscillare 'swing'	28	11	39%	1	4%
permettere 'allow'	211	51	24%	8	4%
piacere 'like'	36	11	31%	5	14%
portare 'bring'	432	107	25%	25	6%
produrre 'produce'	110	30	27%	3	3%
scegliere 'choose'	112	30	27%	10	9%
sembrare 'seem'	422	108	26	15	4%
sentire 'feel'	68	25	37%	3	4%
stabilire 'establish'	160	48	30%	15	9%
tagliare 'cut'	90	29	32%	11	12%
terminare 'end'	35	11	31%	3	9%
trovare 'find'	413	129	31%	15	4%
AVERAGE	188.6	44.9	30%	9,4	7%

Table 7: Number and percentage of selected and extracted cores for the 30 verbs

The average values given in table 7 do not imply that these percentages are fairly constant or that, for bigger corpora, the amount of extracted and selected cores indefinitely grows. On the contrary, in Table 8 the percentage[4] of selected cores (potential subcategorization patterns) resulting at the end of the extraction procedure for a high frequency verb (*dare* 'give', on the left), a medium frequency verb (*cominciare* 'start', in the center) and a low frequency verb (*piacere* 'like', on the right) is given: the gentle decrease of this value for increasing corpus sizes clearly indicates that the extraction procedure does not extract more and more patterns as the corpus grows.

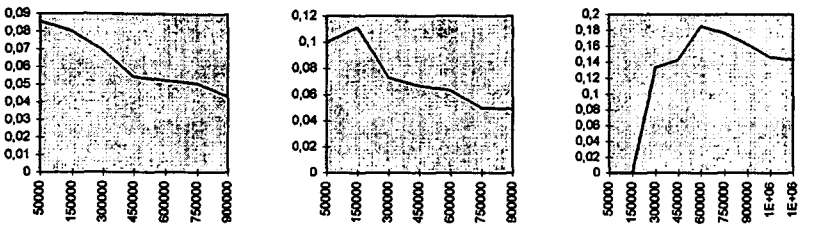

Table 8: Percentage of selected cores for *dare* 'give', *cominciare* 'start' and *piacere* 'like' (X-axis: number of words in the acquisition corpus; Y-axis: percentage of selected cores)

Furthermore, despite the number of filtering steps (e.g. carving, coring, delimitation heuristics), the amount of lost information is small. Indeed, the analysis of pruned out cores has shown that they almost ever contain useless information. In Table 9 the number of frames that have been correctly selected, with respect to the total amount of frames listed in the Parole lexicon and the frames available in the corpus, is shown.

verb	lexicon frames	identified	missed	not in the Corpus	recall wrt Parole	recall wrt Corpus
aggiungere 'add'	6	5	0	0	83%	83%
aggiustare 'fix'	3	2	1	1	67%	100%
aiutare 'help'	5	5	0	0	100%	100%
aspettare 'expect'	6	3	2	0	50%	50%
cambiare 'change'	10	5	5	4	50%	83%
caricare 'charge'	5	0	5	1	0%	0%
causare 'cause'	1	1	0	0	100%	100%
chiamare 'call'	8	3	5	2	38%	50%
chiedere 'ask'	8	6	1	0	75%	75%
cominciare 'begin'	7	6	1	0	86%	86%
concordare 'agree'	7	4	4	2	57%	80%
considerare 'consider'	11	5	6	2	45%	56%
costruire 'build'	1	0	1	0	0%	0%
credere 'believe'	9	5	4	2	56%	71%
dare 'give'	12	8	4	3	67%	89%
decidere 'decide'	8	6	2	1	75%	86%
fornire 'provide'	4	2	2	1	50%	67%
muovere 'move'	14	4	10	9	29%	80%
oscillare 'swing'	2	1	1	0	50%	50%
permettere 'allow'	5	3	2	0	60%	60%
piacere 'like'	3	0	3	1	0%	0%
portare 'bring'	19	14	5	0	74%	74%
produrre 'produce'	6	1	5	4	17%	50%
scegliere 'choose'	6	3	3	0	50%	50%

verb	lexicon frames	identified	missed	not in the Corpus	recall wrt Parole	recall wrt Corpus
sembrare 'seem'	17	6	11	8	35%	67%
sentire 'feel'	14	4	10	5	29%	44%
stabilire 'establish'	6	4	2	2	67%	100%
tagliare 'cut'	13	8	5	5	62%	100%
terminare 'end'	5	3	2	1	60%	75%
trovare 'find'	12	8	4	2	67%	80%
AVERAGE	7.77	4.17	3.5	1.87	53%	70%

Table 9: Sparkle subcat frames vs Parole Lexicon and the Corpus

As we can see, even from a comparatively small corpus of 1 million words, the system extracted more than a half of the expected frames (recall wrt Parole). For what concerns the case of a context not matching any of the selected subcategorization patterns, this is mainly due to the fact that this context occurs only once in the corpus and then it is the only context matching its subcategorization pattern (Calzolari et al. 1997, Federici et al. 1997). Indeed, as shown above, the Mapping Function needs two contexts to extract a potential subcategorization pattern (Federici & Pirrelli 1997). Furthermore, because of the size of the corpus (and its specialization in the financial domain), not all frames are available. So, when compared to the frames really available in the corpus, 70% of them are selected. This does not imply that the remaining frames are not extracted at all. Indeed, most of the missed frames are discarded as low frequent ones. We expect that these frames will be promoted to selected frames when we increase the corpus size. In this case the recall (recall wrt Corpus) nicely grows to 84%. Finally, not all selected frames are correct. In Table 10 we counted the number of correctly selected frames, that is those frames that do not contain chunks that do not depend on the head verb.

verb	selected cores	correct selections	precision of correct selections
aggiungere 'add'	7	7	100%
aggiustare 'fix'	2	2	100%
aiutare 'help'	6	6	100%
aspettare 'expect'	7	5	71%
cambiare 'change'	13	8	62%
caricare 'charge'	1	1	100%
causare 'cause'	3	1	33%
chiamare 'call'	5	5	100%
chiedere 'ask'	16	15	94%
cominciare 'begin'	9	7	78%
concordare 'agree'	7	6	86%
considerare 'consider'	11	7	64%
costruire 'build'	1	1	100%
credere 'believe'	7	6	86%
dare 'give'	41	10	24%
decidere 'decide'	15	10	67%
fornire 'provide'	9	2	22%
muovere 'move'	9	9	100%
oscillare 'swing'	1	1	100%
permettere 'allow'	8	8	100%
piacere 'like'	5	5	100%
portare 'bring'	25	17	68%

verb	selected cores	correct selections	precision of correct selections
produrre 'produce'	3	2	67%
scegliere 'choose'	10	8	80%
sembrare 'seem'	15	11	73%
sentire 'feel'	3	3	100%
stabilire 'establish'	15	9	60%
tagliare 'cut'	11	7	64%
terminare 'end'	3	3	100%
trovare 'find'	15	11	73%
AVERAGE	9.4	6.4	79%

Table 10: Precision of selected frames

More detailed results concerning those frames of the Parole Lexicon that have not been extracted (missed frames) and the frames that have been selected, but do not belong to the Parole Lexicon (over-extracted frames), are given in Table 11.

verb	correct selections	missed frames			over-extracted frames				
		not selected	low freq	total	redundant	partial	noisy	chunker errors	total
aggiungere 'add'	5	0	0	0	3	0	0	0	3
aggiustare 'fix'	2	0	0	0	0	0	0	0	0
aiutare 'help'	5	0	0	0	1	0	0	0	1
aspettare 'expect'	3	2	0	2	3	0	0	0	3
cambiare 'change'	5	1	0	1	5	0	4	1	10
caricare 'charge'	0	4	0	4	0	1	0	0	1
causare 'cause'	1	0	0	0	0	0	1	1	2
chiamare 'call'	3	1	2	3	0	0	0	0	0
chiedere 'ask'	6	1	0	1	8	0	0	1	9
cominciare 'begin'	6	0	1	1	0	0	0	2	2
concordare 'agree'	4	0	2	2	1	0	1	0	2
considerare 'consider'	5	2	2	4	4	0	1	0	5
costruire 'build'	0	0	1	1	1	0	0	0	1
credere 'believe'	5	0	2	2	2	0	0	1	3
dare 'give'	8	1	3	4	3	0	30	1	34
decidere 'decide'	6	0	1	1	6	0	4	1	11
fornire 'provide'	2	0	1	1	0	0	7	0	7
muovere 'move'	4	1	0	1	5	0	0	0	5
oscillare 'swing'	1	0	1	1	0	0	0	0	0
permettere 'allow'	3	0	2	2	5	0	0	0	5
piacere 'like'	0	2	0	2	2	3	0	0	5
portare 'bring'	14	2	3	5	1	4	8	0	13
produrre 'produce'	1	0	1	1	1	0	1	0	2
scegliere 'choose'	3	3	0	3	5	0	2	0	7
sembrare 'seem'	6	3	8	11	5	0	4	0	9
sentire 'feel'	4	4	1	5	0	0	0	0	0
stabilire 'establish'	4	0	0	0	5	0	1	0	6

verb	correct selections	missed frames			over-extracted frames				
		not selected	low freq	total	redundant	partial	noisy	chunker errors	total
tagliare 'cut'	8	0	0	0	0	0	2	2	4
terminare 'end'	3	1	0	1	0	0	0	0	0
trovare 'find'	8	0	2	2	3	0	7	0	10

Table 11: Missed and over-extracted frame analysis

Most of the frames that have not been extracted are frames that occur rarely in the corpus (low freq). As to over-extracted ones, frames that contain adjuncts that have not been considered as relevant to the head verb in the Parole lexicon (redundant frames), but that are not wrong in a strict sense as they still depend on the head verb, are near 44%.

In the final table (Table 12) we compare the amount of frames that are listed in the Parole lexicon (Lexicon) with the frames that are available in the corpus (Corpus), the frames that can be extracted by our algorithm as they appears more than once in the corpus (Corpus > 1) and the frames that are currently selected by our algorithm (Sparkle). The number of selected frames is then augmented with the number of frames that have been extracted but that have been classified as low frequent (low freq) due to their small number of occurrences in the corpus (Expected Sparkle). This last figure is the number of frames that we expect to be extracted as the size of the corpus increases.

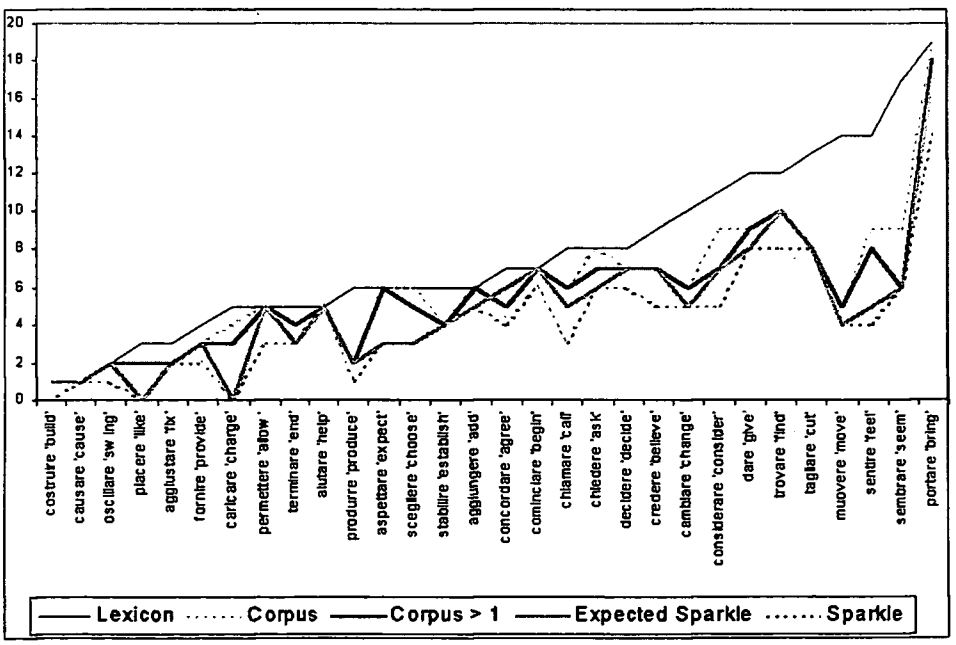

Table 12: Number of frames for the 30 verbs (X-axis: Sparkle verbs; Y-axis: number of frames)

The comparison of the number of frames that our algorithm could potentially extract (Corpus > 1) and the number of selected and low frequent frames (Expected Sparkle) gives a good idea of the potential of our algorithm even when the kind of constraints we imposed on the extraction procedure (e.g. continuity, carving, pruning of some contexts) is used. The biggest differences between the available and the selected frames are for the verbs *caricare* 'charge', *aspettare* 'expect', *sentire* 'feel' and *piacere* 'feel'. For *aspettare* and *sentire* the failure is due to a small amount of available contexts for each frame: they range from 2 to 3. For *caricare* and *piacere*, the strict constraints imposed on the position of the complements (we asked for a perfect correspondence for the DIRECTION and POSITION features) have blocked the identification of long distance matches.

4. Conclusions

We have used a text corpus of about one million words to obtain the above shown results. For such a comparatively small corpus the illustrated procedure could be considered as discarding too much relevant information. The idea behind the acquisition procedure is that, by means of an analogy-based algorithm, the subcategorization patterns of a verb can be acquired from simple cases without loss of important patterns, even if the amount of available contexts is relatively small (Calzolari et al. 1997, Federici et al. 1997): cases of only 20-30 or fewer contexts per verb are not unusual.[5]

Nevertheless, the algorithm proved to be accurate and results are promising. We have compared the potential subcategorization patterns selected by the analogy-based algorithm with a set of hand-crafted subcategorization patterns produced within the LE PAROLE project (Ruimy et al. 1998). Recall and precision are fairly good and are expected to increase when a bigger corpus is used.

5. Notes

[1] The set of values given in the Table are the ones used in the extraction experiment carried out on a set of 30 verbs selected by the Sparkle Project Consortium.

[2] Hereafter the formalized features are not explicitely shown within the chunks.

[3] In order to obtain a better performance of the core extraction procedure, modifiers appearing in the headword chunk are moved in contextual chunks with relative position 0.

[4] Percentages are calculated wrt the total amount of extracted contexts (the number of occurrences of the verb).

[5] Procedures for extraction of linguistic knowledge from textual corpora are mainly based on statistics. However, the estimation of parameters in statistical methods for knowledge extraction is generally not credited to be reliable enough when confronted with so sparse data.

6. References

Briscoe T. (1997). Robust Parsing, in Varile G.B., Zampolli A. (eds.), *Survey of the State of the Art in Human Language Technology.* Giardini Editori, Pisa.

SPARKLE project (1995). Technical Annex.

Briscoe T., J. Carroll, G. Carroll, S. Federici, G. Grefenstette, S. Montemagni, V. Pirrelli, M. Rooth, I. Prodanof, M. Vannocchi. (1996). Phrasal Parser Software, in Deliverable 3.1. LE-2111 SPARKLE project.

Calzolari N., S. Federici, S. Montemagni, V. Pirrelli (1997). Contribution to Deliverable 5.1 "Syntactic and Semantic Type and Selection". LE-2111 SPARKLE project.

Federici S., S. Montemagni, V. Pirrelli (1996). Shallow Parsing and Text Chunking: a View on Underspecification in Syntax, in Proceeding of the *Workshop on Robust Parsing*. ESSLLI, Prague.

Federici S., S. Montemagni, V. Pirrelli (1997). The Automatic Building of a Lexicon from Textual Corpora. Sparkle Working paper. Submitted to Coling 98.

Federici S., V. Pirrelli (1997). Analogy, Computation and Linguistic Theory, in Jones D., Somers H. (eds), *New Methods in Language Processing*. UCL Press, London.

Ruimy N., M. Battista, O. Corazzari, E. Gola, A. Spanu (1998). Italian Lexicon Documentation, in WP3.11. LE-PAROLE, Pisa.