Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling

Xavier Carreras and **Lluís Màrquez**
TALP Research Centre
Technical University of Catalonia (UPC)
{carreras,lluism}@lsi.upc.edu

Abstract

In this paper we describe the CoNLL-2005 shared task on Semantic Role Labeling. We introduce the specification and goals of the task, describe the data sets and evaluation methods, and present a general overview of the 19 systems that have contributed to the task, providing a comparative description and results.

1 Introduction

In the few last years there has been an increasing interest in shallow semantic parsing of natural language, which is becoming an important component in all kind of NLP applications. As a particular case, Semantic Role Labeling (SRL) is currently a well-defined task with a substantial body of work and comparative evaluation. Given a sentence, the task consists of analyzing the propositions expressed by some target verbs of the sentence. In particular, for each target verb all the constituents in the sentence which fill a semantic role of the verb have to be recognized. Typical semantic arguments include Agent, Patient, Instrument, etc. and also adjuncts such as Locative, Temporal, Manner, Cause, etc.

Last year, the CoNLL-2004 shared task aimed at evaluating machine learning SRL systems based only on partial syntactic information. In (Carreras and Màrquez, 2004) one may find a detailed review of the task and also a brief state-of-the-art on SRL previous to 2004. Ten systems contributed to the task, which was evaluated using the PropBank corpus (Palmer et al., 2005). The best results were around 70 in F_1 measure. Though not directly comparable, these figures are substantially lower than the best results published up to date using full parsing as input information (F_1 slightly over 79). In addition to the CoNLL-2004 shared task, another evaluation exercise was conducted in the Senseval-3 workshop (Litkowski, 2004). Eight systems relying on full parsing information were evaluated in that event using the FrameNet corpus (Fillmore et al., 2001). From the point of view of learning architectures and study of feature relevance, it is also worth mentioning the following recent works (Punyakanok et al., 2004; Moschitti, 2004; Xue and Palmer, 2004; Pradhan et al., 2005a).

Following last year's initiative, the CoNLL-2005 shared task[1] will concern again the recognition of semantic roles for the English language. Compared to the shared task of CoNLL-2004, the novelties introduced in the 2005 edition are:

- Aiming at evaluating the contribution of full parsing in SRL, the complete syntactic trees given by two alternative parsers have been provided as input information for the task. The rest of input information does not vary and corresponds to the levels of processing treated in the previous editions of the CoNLL shared task, i.e., words, PoS tags, base chunks, clauses, and named entities.

- The training corpus has been substantially enlarged. This allows to test the scalability of

[1] The official CoNLL-2005 shared task web page, including data, software and systems' outputs, is available at http://www.lsi.upc.edu/∼srlconll.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 152–164, Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

learning-based SRL systems to big datasets and to compute learning curves to see how much data is necessary to train. Again, we concentrate on the PropBank corpus (Palmer et al., 2005), which is the Wall Street Journal part of the Penn TreeBank corpus enriched with predicate–argument structures.

- In order to test the robustness of the presented systems, a cross-corpora evaluation is performed using a fresh test set from the Brown corpus.

Regarding evaluation, two different settings were devised depending if the systems use the information strictly contained in the training data (*closed challenge*) or they make use of external sources of information and/or tools (*open challenge*). The closed setting allows to compare systems under strict conditions, while the open setting aimed at exploring the contributions of other sources of information and the limits of the current learning-based systems on the SRL task. At the end, all 19 systems took part in the closed challenge and none of them in the open challenge.

The rest of the paper is organized as follows. Section 2 describes the general setting of the task. Section 3 provides a detailed description of training, development and test data. Participant systems are described and compared in section 4. In particular, information about learning techniques, SRL strategies, and feature development is provided, together with performance results on the development and test sets. Finally, section 5 concludes.

2 Task Description

As in the 2004 edition, the goal of the task was to develop a machine learning system to recognize arguments of verbs in a sentence, and label them with their semantic role. A verb and its set of arguments form a *proposition* in the sentence, and typically, a sentence contains a number of propositions.

There are two properties that characterize the structure of the arguments in a proposition. First, arguments do not overlap, and are organized sequentially. Second, an argument may appear split into a number of non-contiguous phrases. For instance, in the sentence "[$_{\text{A1}}$ The apple], said John, [$_{\text{C–A1}}$ is on the table]", the utterance argument (labeled with type `A1`) appears split into two phrases. Thus, there is a set of non-overlapping arguments labeled with semantic roles associated with each proposition. The set of arguments of a proposition can be seen as a chunking of the sentence, in which chunks are parts of the semantic roles of the proposition predicate.

In practice, number of *target verbs* are marked in a sentence, each governing one proposition. A system has to recognize and label the arguments of each target verb. To support the role labeling task, sentences contain input annotations, that consist of syntactic information and named entities. Section 3 describes in more detail the annotations of the data.

2.1 Evaluation

Evaluation is performed on a collection of unseen test sentences, that are marked with target verbs and contain only predicted input annotations.

A system is evaluated with respect to *precision*, *recall* and the F_1 measure of the predicted arguments. *Precision* (p) is the proportion of arguments predicted by a system which are correct. *Recall* (r) is the proportion of correct arguments which are predicted by a system. Finally, the F_1 measure computes the harmonic mean of precision and recall, and is the final measure to compare the performance of systems. It is formulated as: $\text{F}_{\beta=1} = 2pr/(p+r)$.

For an argument to be correctly recognized, the words spanning the argument as well as its semantic role have to be correct. [2]

As an exceptional case, the verb argument of each proposition is excluded from the evaluation. This argument is the lexicalization of the predicate of the proposition. Most of the time, the verb corresponds to the target verb of the proposition, which is provided as input, and only in few cases the verb participant spans more words than the target verb. Except for non-trivial cases, this situation makes the verb fairly easy to identify and, since there is one verb with each proposition, evaluating its recognition over-estimates the overall performance of a system. For this reason, the verb argument is excluded from evaluation.

[2] The `srl-eval.pl` program is the official program to evaluate the performance of a system. It is available at the Shared Task web page.

```
And           CC      *   (S*      (S*         *     -        (AM-DIS*)  (AM-DIS*)
to            TO   (VP*   (S*   (S(VP*         *     -               *   (AM-PNC*
attract       VB      *)    *     (VP*         *     attract       (V*)         *
younger       JJR  (NP*     *     (NP*         *     -            (A1*          *
listeners     NNS     *)    *)       *))))     *     -               *)         *)
,             ,       *     *        *         *     -               *          *
Radio         NNP  (NP*     *     (NP*     (ORG*     -            (A0*       (A0*
Free          NNP     *     *        *         *     -               *          *
Europe        NNP     *)    *        *)        *)    -               *)         *)
intersperses  VBZ  (VP*)    *     (VP*         *     intersperse     *        (V*)
the           DT   (NP*     *  (NP(NP*         *     -               *       (A1*
latest        JJS     *)    *        *)        *     -               *          *
in            IN   (PP*)    *     (PP*         *     -               *          *
Western       JJ   (NP*     *     (NP*    (MISC*)    -               *          *
rock          NN      *     *        *         *     -               *          *
groups        NNS     *)    *        *))))     *     -               *          *)
.             .       *     *)       *)        *     -               *          *
```

Figure 1: An example of an annotated sentence, in columns. Input consists of words (1st column), PoS tags (2nd), base chunks (3rd), clauses (4th), full syntactic tree (5th) and named entities (6th). The 7th column marks target verbs, and their propositions are found in remaining columns. According to the PropBank Frames, for `attract` (8th), the `A0` annotates the attractor, and the `A1` the thing attracted; for `intersperse` (9th), `A0` is the arranger, and `A1` the entity interspersed.

2.2 Closed Challenge Setting

The organization provided training, development and test sets derived from the standard sections of the Penn TreeBank (Marcus et al., 1993) and PropBank (Palmer et al., 2005) corpora.

In the closed challenge, systems have to be built strictly with information contained in the training sections of the TreeBank and PropBank. Since this collection contains the gold reference annotations of both syntactic and predicate-argument structures, the closed challenge allows: (1) to make use of any preprocessing system strictly developed within this setting, and (2) to learn from scratch any annotation that is contained in the data. To support the former, the organization provided the output of state-of-the-art syntactic preprocessors, described in Section 3.

The development set is used to tune the parameters of a system. The gold reference annotations are also available in this set, but only to evaluate the performance of different parametrizations of a system, and select the optimal one. Finally, the test set is used to evaluate the performance of a system. It is only allowed to use predicted annotations in this set.

Since all systems in this setting have had access to the same training and development data, the evaluation results on the test obtained by different systems are comparable in a fair manner.

3 Data

The data consists of sections of the Wall Street Journal part of the Penn TreeBank (Marcus et al., 1993), with information on predicate-argument structures extracted from the PropBank corpus (Palmer et al., 2005). In this edition of the CoNLL shared task, we followed the standard partition used in syntactic parsing: sections 02-21 for training, section 24 for development, and section 23 for test. In addition, the test set of the shared task includes three sections of the Brown corpus (namely, ck01-03). The predicate-argument annotations of the latter test material were kindly provided by the PropBank team, and are very valuable, as they allow to evaluate learning systems on a portion of data that comes from a different source than training.

We first describe the annotations related to argument structures. Then, we describe the preprocessing systems that have been selected to predict the input part of the data. Figure 1 shows an example of a fully-annotated sentence.

3.1 PropBank

The Proposition Bank (PropBank) (Palmer et al., 2005) annotates the Penn TreeBank with verb argument structure. The semantic roles covered by PropBank are the following:

- **Numbered arguments (A0–A5, AA):** Arguments defining verb-specific roles. Their semantics depends on the verb and the verb usage in a sentence, or *verb sense*. The most frequent roles are A0 and A1 and, commonly, A0 stands for the *agent* and A1 corresponds to the *patient* or *theme* of the proposition. However, no consistent generalization can be made across different verbs or different senses of the same verb. PropBank takes the definition of verb senses from VerbNet, and for each verb and each sense defines the set of possible roles for that verb usage, called the *roleset*. The definition of rolesets is provided in the PropBank *Frames files*, which is made available for the shared task as an *official resource* to develop systems.

- **Adjuncts (AM-):** General arguments that any verb may take optionally. There are 13 types of adjuncts:

  AM-ADV : general-purpose
  AM-CAU : cause
  AM-DIR : direction
  AM-DIS : discourse marker
  AM-EXT : extent
  AM-LOC : location
  AM-MNR : manner
  AM-MOD : modal verb
  AM-NEG : negation marker
  AM-PNC : purpose
  AM-PRD : predication
  AM-REC : reciprocal
  AM-TMP : temporal

- **References (R-):** Arguments representing arguments realized in other parts of the sentence. The role of a reference is the same as the role of the referenced argument. The label is an R- tag prefixed to the label of the referent, e.g. R-A1.

- **Verbs (V):** Argument corresponding to the verb of the proposition. Each proposition has exaclty one verb argument.

We used PropBank-1.0. Most predicative verbs were annotated, although not all of them (for example, most of the occurrences of the verb "to have" and "to be" were not annotated). We applied procedures to check consistency of propositions, looking for overlapping arguments, and incorrect semantic role labels. Also, co-referenced arguments were annotated as a single item in PropBank, and we automatically distinguished between the referent and the reference with simple rules matching pronominal expressions, which were tagged as R arguments.

	Train.	Devel.	tWSJ	tBrown
Sentences	39,832	1,346	2,416	426
Tokens	950,028	32,853	56,684	7,159
Propositions	90,750	3,248	5,267	804
Verbs	3,101	860	982	351
Arguments	239,858	8,346	14,077	2,177
A0	61,440	2,081	3,563	566
A1	84,917	2,994	4,927	676
A2	19,926	673	1,110	147
A3	3,389	114	173	12
A4	2,703	65	102	15
A5	68	2	5	0
AA	14	1	0	0
AM	7	0	0	0
AM-ADV	8,210	279	506	143
AM-CAU	1,208	45	73	8
AM-DIR	1,144	36	85	53
AM-DIS	4,890	202	320	22
AM-EXT	628	28	32	5
AM-LOC	5,907	194	363	85
AM-MNR	6,358	242	344	110
AM-MOD	9,181	317	551	91
AM-NEG	3,225	104	230	50
AM-PNC	2,289	81	115	17
AM-PRD	66	3	5	1
AM-REC	14	0	2	0
AM-TMP	16,346	601	1,087	112
R-A0	4,112	146	224	25
R-A1	2,349	83	156	21
R-A2	291	5	16	0
R-A3	28	0	1	0
R-A4	7	0	1	0
R-AA	2	0	0	0
R-AM-ADV	5	0	2	0
R-AM-CAU	41	3	4	2
R-AM-DIR	1	0	0	0
R-AM-EXT	4	1	1	0
R-AM-LOC	214	9	21	4
R-AM-MNR	143	6	6	2
R-AM-PNC	12	0	0	0
R-AM-TMP	719	31	52	10

Table 1: Counts on the data sets.

A total number of 80 propositions were not compliant with our procedures (one in the Brown files, the rest in WSJ) and were filtered out from the CoNLL data sets.

Table 1 provides counts of the number of sentences, tokens, annotated propositions, distinct verbs, and arguments in the four data sets.

3.2 Preprocessing Systems

In this section we describe the selected processors that computed input annotations for the SRL systems. The annotations are: part-of-speech (PoS) tags, chunks, clauses, full syntactic trees and named entities. As it has been noted, participants were also

allowed to use any processor developed within the same WSJ partition.

The preprocessors correspond to the following state-of-the-art systems:

- UPC processors, consisting of:
  - PoS tagger: (Giménez and Màrquez, 2003), based on Support Vector Machines, and trained on WSJ sections 02-21.
  - Base Chunker and Clause Recognizer: (Carreras and Màrquez, 2003), based on Voted Perceptrons, trained on WSJ sections 02-21. These two processors form a coherent partial syntax of a sentence, that is, chunks and clauses form a partial syntactic tree.
- Full parser of Collins (1999), with "model 2". Predicts WSJ full parses, with information of the lexical head for each syntactic constituent. The PoS tags (required by the parser) have been computed with (Giménez and Màrquez, 2003).
- Full parser of Charniak (2000). Jointly predicts PoS tags and full parses.
- Named Entities predicted with the Maximum-Entropy based tagger of Chieu and Ng (2003). The tagger follows the CoNLL-2003 task setting (Tjong Kim Sang and De Meulder, 2003), and thus is not developed with WSJ data. However, we allowed its use because there is no available named entity recognizer developed with WSJ data. The reported performance on the CoNLL-2003 test is $F_1 = 88.31$, with Prec/Rec. at 88.12/88.51.

Tables 2 and 3 summarize the performance of the syntactic processors on the development and test sets. The performance of full parsers on the WSJ test is lower than that reported in the corresponding papers. The reason is that our evaluation figures have been computed in a strict manner with respect to punctuation tokens, while the full parsing community usually does not penalize for punctuation wrongly placed in the tree.[3] As it can be observed, the performance of all syntactic processors suffers a substantial loss in the Brown test set. Noticeably, the parser of Collins (1999) seems to be the more robust when moving from WSJ to Brown.

[3]Before evaluating Collins', we raised punctuation to the highest point in the tree, using a script that is available at the shared task webpage. Otherwise, the performance would have Prec./Recall figures below 37.

	Dev.	tWSJ	tBrown
UPC PoS-tagger	97.13	97.36	94.73
Charniak (2000)	92.01	92.29	87.89

Table 2: Accuracy (%) of PoS taggers.

4 A Review of Participant Systems

Nineteen systems participated in the CoNLL-2005 shared task. They approached the task in several ways, using different learning components and labeling strategies. The following subsections briefly summarize the most important properties of each system and provide a qualitative comparison between them, together with a quantitative evaluation on the development and test sets.

4.1 Learning techniques

Up to 8 different learning algorithms have been applied to train the learning components of participant systems. See the "ML-method" column of table 4 for a summary of the following information. Log–linear models and vector-based linear classifiers dominated over the rest. Probably, this is due to the versatility of the approaches and the availability of very good software toolkits.

In particular, 8 teams used the Maximum Entropy (ME) statistical framework (Che et al., 2005; Haghighi et al., 2005; Park and Rim, 2005; Tjong Kim Sang et al., 2005; Sutton and McCallum, 2005; Tsai et al., 2005; Yi and Palmer, 2005; Venkatapathy et al., 2005). Support Vector Machines (SVM) were used by 6 teams. Four of them with the standard polynomial kernels (Mitsumori et al., 2005; Tjong Kim Sang et al., 2005; Tsai et al., 2005; Pradhan et al., 2005b), another one using Gaussian kernels (Ozgencil and McCracken, 2005), and a last group using tree-based kernels specifically designed for the task (Moschitti et al., 2005). Another team used also a related learning approach, SNoW, which is a Winnow-based network of linear separators (Punyakanok et al., 2005).

Decision Tree learning (DT) was also represented

	Devel.			Test WSJ			Test Brown		
	P(%)	R(%)	F_1	P(%)	R(%)	F_1	P(%)	R(%)	F_1
UPC Chunker	94.66	93.17	93.91	95.26	94.52	94.89	92.64	90.85	91.73
UPC Clauser	90.38	84.73	87.46	90.93	85.94	88.36	84.21	74.32	78.95
Collins (1999)	85.02	83.55	84.28	85.63	85.20	85.41	82.68	81.33	82.00
Charniak (2000)	87.60	87.38	87.49	88.20	88.30	88.25	80.54	81.15	80.84

Table 3: Results of the syntactic parsers on the development, and WSJ and Brown test sets. Unlike in full parsing, the figures have been computed on a strict evaluation basis with respect to punctuation.

by Ponzetto and Strube (2005), who used C4.5. Ensembles of decision trees learned through the AdaBoost algorithm (AB) were applied by Màrquez et al. (2005) and Surdeanu and Turmo (2005). Tjong Kim Sang et al. (2005) applied, among others, Memory-Based Learning (MBL).

Regarding novel learning paradigms not applied in previous shared tasks, we find Relevant Vector Machine (RVM), which is a kernel–based linear discriminant inside the framework of Sparse Bayesian Learning (Johansson and Nugues, 2005) and Tree Conditional Random Fields (T-CRF) (Cohn and Blunsom, 2005), that extend the sequential CRF model to tree structures. Finally, Lin and Smith (2005) presented a proposal radically different from the rest, with very light learning components. Their approach (Consensus in Pattern Matching, CPM) contains some elements of Memory-based Learning and ensemble classification.

From the Machine Learning perspective, system combination is another interesting component observed in many of the proposals. This fact, which is a difference from last year shared task, is explained as an attempt of increasing the robustness and coverage of the systems, which are quite dependent on input parsing errors. The different outputs to combine are obtained by varying input information, changing learning algorithm, or considering n-best solution lists. The combination schemes presented include very simple voting-like combination heuristics, stacking of classifiers, and a global constraint satisfaction framework modeled with Integer Linear Programming. Global models trained to re-rank alternative outputs represent a very interesting alternative that has been proposed by two systems. All these issues are reviewed in detail in section 4.2.

4.2 SRL approaches

SRL is a complex task, which may be decomposed into a number of simpler decisions and annotating schemes in order to be addressed by learning techniques. Table 4 contains a summary of the main properties of the 19 systems presented. In this section we will explain the contents of that table by columns (from left-to-right).

One first issue to consider is the input structure to navigate in order to extract the constituents that will form labeled arguments. The majority of systems perform parse tree node labeling, searching for a one–to–one map between arguments and parse constituents. This information is summarized in the "synt" column of Table 4. "col", "cha", "upc" stand for the syntactic parse trees (the latter is partial) provided as input by the organization. Additionally, some teams used lists of n-best parsings generated by available tools ("n-cha" by Charniak parser; "n-bikel" by Bikel's implementation of Collins parser). Interestingly, Yi and Palmer (2005) retrained Ratnaparkhi's parser using the WSJ training sections enriched with semantic information coming from PropBank annotations. These are referred to as AN and AM parses. As it can be seen, Charniak parses were used by most of the systems. Collins parses were used also in some of the best performing systems based on combination.

The exceptions to the hierarchical processing are the systems by Pradhan et al. (2005b) and Mitsumori et al. (2005), which perform a chunking-based sequential tokenization. As for the former, the system is the same than the one presented in the 2004 edition. The system by Màrquez et al. (2005) explores hierarchical syntactic structures but selects, in a preprocess, a sequence of tokens to perform a sequential tagging afterwards.

	ML-method	synt	pre	label	embed	glob	post	comb	type
punyakanok	SNoW	n-cha,col	x&p	i+c	defer	yes	no	n-cha+col	ac-ILP
haghighi	ME	n-cha	?	i+c	dp-prob	yes	no	n-cha	re-rank
marquez	AB	cha,upc	seq	bio	!need	no	no	cha+upc	s-join
pradhan	SVM	cha,col/chunk	?	c/bio	?	no	no	cha+col→chunk	stack
surdeanu	AB	cha	prun	c	g-top	no	yes	no	–
tsai	ME,SVM	cha	x&p	c	defer	yes	no	ME+SVM	ac-ILP
che	ME	cha	no	c	g-score	no	yes	no	–
moschitti	SVM	cha	prun	i+c	!need	no	no	no	–
tjongkimsang	ME,SVM,TBL	cha	prun	i+c	!need	no	yes	ME+SVM+TBL	s-join
yi	ME	cha,AN,AM	x&p	i+c	defer	no	no	cha+AN+AM	ac-join
ozgencil	SVM	cha	prun	i+c	g-score	no	no	no	–
johansson	RVM	cha	softp	i+c	?	no	no	no	–
cohn	T-CRF	col	x&p	c	g-top	yes	no	no	–
park	ME	cha	prun	i+c	?	no	no	no	–
mitsumori	SVM	chunk	no	bio	!need	no	no	no	–
venkatapathy	ME	col	prun	i+c	frames	yes	no	no	–
ponzetto	DT	col	prun	c	g-top	no	yes	no	–
lin	CPM	cha	gt-para	i+c	!need	no	no	no	–
sutton	ME	n-bikel	x&p	i+c	dp-prob	yes	no	n-bikel	re-rank

Table 4: Main properties of the SRL strategies implemented by the participant teams, sorted by F_1 performance on the WSJ+Brown test set. **synt** stands for the syntactic structure explored; **pre** stands for pre-processing steps; **label** stands for the labeling strategy; **embed** stands for the technique to ensure non-embedding of arguments; **glob** stands for global optimization; **post** stands for post-processing; **comb** stands for system output combination, and **type** stands for the type of combination. Concrete values appearing in the table are explained in section 4.1. The symbol "?" stands for unknown values not reported by the system description papers.

In general, the presented systems addressed the SRL problem by applying different chained processes. In Table 4 the column "pre" summarizes preprocessing. In most of the cases this corresponds to a pruning procedure to filter out constituents that are not likely to be arguments. As in feature development, the related bibliography has been followed for pruning. For instance, many systems used the pruning strategy described in (Xue and Palmer, 2004) ("x&p") and other systems used the soft pruning rules described in (Pradhan et al., 2005a) ("softp"). Remarkably, Park and Rim (2005) parametrize the pruning procedure and then study the effect of being more or less aggressive at filtering constituents. In the case of Màrquez et al. (2005), pre-processing corresponds to a sequentialization of syntactic hierarchical structures. As a special case, Lin and Smith (2005) used the GT-PARA analyzer for converting parse trees into a flat representation of all predicates including argument boundaries.

The second stage, reflected in column "label" of Table 4, is the proper labeling of selected candidates. Most of the systems used a two-step procedure consisting of first identifying arguments (e.g., with a binary "null" vs. "non-null" classifier) and then classifying them. This is referred to as "i+c" in the table. Some systems address this phase in a single classification step by adding a "null" category to the multiclass problem (referred to as "c'). The methods performing a sequential tagging use a BIO tagging scheme ("bio"). As a special case, Moschitti et al. (2005) subdivide the "i+c" strategy into four phases: after identification, heuristics are applied to assure compatibility of identified arguments; and, before classifying arguments into roles, a preclassification into core vs. adjunct arguments is performed. Venkatapathy et al. (2005) use three labels instead of two in the identification phase : "null", "mandatory", and "optional".

Since arguments in a solution do not embed and most systems identify arguments as nodes in a hierarchical structure, non-embedding constraints must be resolved in order to generate a coherent argument labeling. The "embed" column of Table 4 accounts for this issue. The majority of systems applied specific greedy procedures that select a subset of consistent arguments. The families of heuristics to do that selection include prioritizing better scored

constituents ("g-score"), or selecting the arguments that are first reached in a top-down exploration ("g-top"). Some probabilistic systems include the non-embedding constraints within the dynamic programming inference component, and thus calculate the most probable coherent labeling ("dp-prob"). The "defer" value means that this is a combination system and that coherence of the individual system predictions is not forced, but deferred to the later combination step. As a particular case, Venkatapathy et al. (2005) use PropBank subcategorization frames to force a coherent solution. Note that tagging-based systems do not need to check non-embedding constraints ("!need" value).

The "glob" column of Table 4 accounts for the locality/globality of the process used to calculate the output solution given the argument prediction candidates. Systems with a "yes" value in that column define some kind of scoring function (possibly probabilistic) that applies to complete candidate solutions, and then calculate the solution that maximizes the scoring using an optimization algorithm.

Some systems use some kind of postprocessing to improve the final output of the system by correcting some systematic errors, or treating some types of simple adjunct arguments. This information is included in the "post" column of Table 4. In most of the cases, this postprocess is performed on the basis of simple ad-hoc rules. However, it is worth mentioning the work of Tjong Kim Sang et al. (2005) in which spelling error correction techniques are adapted for improving the resulting role labeling. In that system, postprocessing is applied before system combination.

Most of the best performing systems included a combination of different base subsystems to increase robustness of the approach and to gain coverage and independence from parse errors. Last 2 columns of Table 4 present this information. In the "comb" column the source of the combination is reported. Basically, the alternative outputs to combine can be generated by different input syntactic structures or n-best parse candidates, or by applying different learning algorithms to the same input information.

The type of combination is reported in the last column. Màrquez et al. (2005) and Tjong Kim Sang et al. (2005) performed a greedy merging of the arguments of base complete solutions ("s-join"). Yi and Palmer (2005) did also a greedy merging of arguments but taking into account not complete solutions but all candidate arguments labeled by base systems ("ac-join"). In a more sophisticated way, Punyakanok et al. (2005) and Tsai et al. (2005) performed global inference as constraint satisfaction using Integer Linear Programming, also taking into account all candidate arguments ("ac-ILP"). It is worth noting that the generalized inference applied in those papers allows to include, jointly with the combination of outputs, a number of linguistically-motivated constraints to obtain a coherent solution.

Pradhan et al. (2005b) followed a stacking approach by learning a chunk-based SRL system including as features the outputs of two syntax-based systems. Finally, Haghighi et al. (2005) and Sutton and McCallum (2005) performed a different approach by learning a re-ranking function as a global model on top of the base SRL models. Actually, Haghighi et al. (2005) performed a double selection step: an inner re-ranking of n-best solutions coming from the base system on a single tree; and an outer selection of the final solution among the candidate solutions coming from n-best parse trees. The re-ranking approach allows to define global complex features applying to complete candidate solutions to train the rankers.

4.3 Features

Looking at the description of the different systems, it becomes clear that the general type of features used in this edition is strongly based on previous work on the SRL task (Gildea and Jurafsky, 2002; Surdeanu et al., 2003; Pradhan et al., 2005a; Xue and Palmer, 2004). With no exception, all systems have made intensive use of syntax to extract features. While most systems work only on the output of a parser —Charniak's being the most preferred— some systems depend on many syntactic parsers. In the latter situation, either a system is a combination of many individual systems (each working with a different parser), or a system extracts features from many different parse trees while exploring the nodes of only one parse tree. Most systems have also considered named entities for extracting features.

The main types of features seen in this SRL edition can be divided into four general categories: (1) Features characterizing the structure of a candidate

	sources		argument							verb		arg–verb						p
	synt	ne	at	aw	ab	ac	ai	pp	sd	v	sc	rp	di	ps	pv	pi	sf	as
punyakanok	cha,col,upc	+	+	h	+	t	+	+	·	+	+	+	c	+	·	+	+	·
haghighi	cha	·	+	h	+	p,s	·	+	+	+	+	+	t	+	+	·	·	+
marquez	cha,upc	+	+	h	+	t	+	·	+	+	+	+	w,c	+	+	·	+	·
pradhan	cha,col,upc	+	+	h,c	+	p,s,t	+	+	·	+	+	+	c,t	+	+	+	+	·
surdeanu	cha	+	+	h,c	+	p,s	+	·	+	+	+	+	w,t	+	+	+	·	·
tsai	cha,upc	+	+	h	+	p,s,t	·	·	·	+	+	+	w	+	·	·	·	·
che	cha	+	+	h	+	·	·	+	·	+	+	+	t	+	+	·	·	·
moschitti	cha	·	+	h	+	p	+	+	·	+	+	+	t	+	+	·	+	·
tjongkimsang	cha	+	+	·	+	p,t	·	+	·	+	+	+	w,t	+	+	+	·	·
yi	cha,an,am	·	+	h,c	·	p,s	·	+	·	+	+	+	w	+	·	·	+	·
ozgencil	cha	·	+	h	·	p	·	·	·	+	+	+	·	+	+	·	·	·
johansson	cha,upc	+	+	h	·	·	·	·	·	+	+	+	·	+	+	·	·	·
cohn	col	·	+	h	+	p,s	·	+	·	+	+	+	w	+	·	+	+	·
park	cha	·	+	h,c	·	p	·	·	·	+	+	+	·	+	·	+	·	·
mitsumori	upc,cha	+	+	·	+	t	·	·	+	+	·	+	c,t	·	+	·	·	·
venkatapathy	col	+	+	h	+	·	·	·	·	+	·	+	·	+	·	·	·	·
ponzetto	col,upc	+	+	h	+	·	+	·	·	+	·	·	w,c,t	·	·	+	·	·
lin	cha	·	+	h	+	·	·	·	·	+	·	+	w	·	·	·	·	·
sutton	bik	·	+	h	+	p,s	·	·	·	+	·	+	·	+	·	·	·	+

Table 5: Main feature types used by the 19 participating systems in the CoNLL-2005 shared task, sorted by performance on the WSJ+Brown test set. **Sources: synt**: use of parsers, namely Charniak (cha), Collins (col), UPC partial parsers (upc), Bikel's Collins model (bik) and/or argument-enriched parsers (an,am); **ne**: use of named entities. **On the argument: at**: argument type; **aw**: argument words, namely the head (h) and/or content words (c); **ab**: argument boundaries, i.e. form and PoS of first and/or last argument words; **ac**: argument context, capturing features of the parent (p) and/or left/right siblings (s), or the tokens surrounding the argument (t); **ai**: indicators of the structure of the argument (e,g., on internal constituents, surrounding/boundary punctuation, governing category, etc.); **pp**: specific features for prepositional phrases; **sd**: semantic dictionaries. **On the verb: v**: standard verb features (voice, word/lemma, PoS); **sc**: subcategorization. **On the arg-verb relation: rp**: relative position; **di**: distance, based on words (w), chunks (c) or the syntactic tree (t); **ps**: standard path; **pv**: path variations; **pi**: scalar indicator variables on the path (of chunks, clauses, or other phrase types), common ancestor, etc.; **sf**: syntactic frame (Xue and Palmer, 2004); **On the complete proposition: as**: sequence of arguments of a proposition.

argument; (2) Features describing properties of the target verb predicate; (3) Features that capture the relation between the verb predicate and the constituent under consideration; and (4) Global features describing the complete argument labeling of a predicate. The rest of the section describes the most common feature types in each category. Table 5 summarizes the type of features exploited by systems.

To represent an argument itself, all systems make use of the syntactic type of the argument. Almost all teams used the heuristics of Collins (1999) to extract the head word of the argument, and used features that capture the form, lemma and PoS tag of the head. In the same line, some systems also use features of the content words of the argument, using the heuristics of Surdeanu et al. (2003). Very generally also, many systems extract features from the first and last words of the argument. Regarding the syntactic elements surrounding the argument, many systems working on full trees have considered the parent and siblings of the argument, capturing their syntactic type and head word. Differently, other systems have captured features from the left/right tokens surrounding the argument, which are typically words, but can be chunks or general phrases in systems that sequentialize the task (Màrquez et al., 2005; Pradhan et al., 2005b; Mitsumori et al., 2005). Many systems use a variety of indicator features that capture properties of the argument structure and its local syntactic annotations. For example, indicators of the immediate syntactic types that form the argument, flags raised by punctuation tokens in or nearby the argument, or the *governing category* feature of Gildea and Jurafsky (2002). It is also somewhat gen-

eral the use of specific features that apply when the constituent is a prepositional phrase, such as looking for the head word of the noun phrase within it. A few systems have also built semantic dictionaries from training data, that collect words appearing frequently in temporal, locative or other arguments.

To represent the predicate, all systems have used features codifying the form, lemma, PoS tag and voice of the verb. It is also of general use the subcategorization feature, capturing the syntactic rule that expands the parent of the predicate. Some systems captured statistics related to the frequency of a verb in training data (not in Table 5).

Regarding features related to an argument-verb pair, almost all systems use the simple feature describing the relative position between them. To a lesser degree, systems have computed distances from one to the other, based on the number of words or chunks between them, or based on the syntactic tree. Not surprisingly, all systems have extracted the path from the argument to the verb. While almost all systems use the standard path of (Gildea and Jurafsky, 2002), many have explored variations of it. A common one consists of the path from the argument to the lowest common ancestor of the verb and the argument. Another variation is the partial path, that is built of chunks and clauses only. Indicator features that capture scalar values of the path are also common, and concentrate mainly on looking at the common ancestor, capturing the difference of clausal levels, or looking for punctuation and other linguistic elements in the path. In this category, it is also noticeable the use of the *syntactic frame* feature, proposed by Xue and Palmer (2004).

Finally, in this edition two systems apply learning at a global context (Haghighi et al., 2005; Sutton and McCallum, 2005) and, consequently, they are able to extract features from a complete labeling of a predicate. Basically, the central feature in this context extracts the sequential pattern of predicate arguments. Then, this pattern can be enriched with syntactic categories, broken down into role-specific indicator variables, or conjoined with the predicate lemma.

Apart from basic feature extraction, combination of features has also been explored in this edition. Many of the combinations depart from the manually selected conjunctions of Xue and Palmer (2004).

4.4 Evaluation

A baseline rate was computed for the task. It was produced using a system developed in the past shared task edition by Erik Tjong Kim Sang, from the University of Amsterdam, The Netherlands. The baseline processor finds semantic roles based on the following seven rules:

- Tag target verb and successive particles as `V`.
- Tag `not` and `n't` in target verb chunk as `AM-NEG`.
- Tag modal verbs in target verb chunk as `AM-MOD`.
- Tag first `NP` before target verb as `A0`.
- Tag first `NP` after target verb as `A1`.
- Tag `that`, `which` and `who` before target verb as `R-A0`.
- Switch `A0` and `A1`, and `R-A0` and `R-A1` if the target verb is part of a passive `VP` chunk. A `VP` chunk is considered in passive voice if it contains a form of `to be` and the verb does not end in `ing`.

Table 6 presents the overall results obtained by the nineteen systems plus the baseline, on the development and test sets (i.e., Development, Test WSJ, Test Brown, and Test WSJ+Brown). The systems are sorted by the performance on the combined WSJ+Brown test set.

As it can be observed, all systems clearly outperformed the baseline. There are seven systems with a final F_1 performance in the 75-78 range, seven more with performances in the 70-75 range, and five with a performance between 65 and 70. The best performance was obtained by Punyakanok et al. (2005), which almost reached an F_1 at 80 in the WSJ test set and almost 78 in the combined test. Their results on the WSJ test equal the best results published so far on this task and datasets (Pradhan et al., 2005a), though they are not directly comparable due to a different setting in defining arguments not perfectly matching the predicted parse constituents. Since the evaluation in the shared task setting is more strict, we believe that the best results obtained in the shared task represent a new breakthrough in the SRL task.

It is also quite clear that the systems using combination are better than the individuals. It is worth noting that the first 4 systems are combined. The

	Development			Test WSJ			Test Brown			Test WSJ+Brown		
	P(%)	R(%)	F_1	P(%)	R(%)	F_1	P(%)	R(%)	F_1	P(%)	R(%)	F_1
punyakanok	80.05	74.83	77.35	82.28	76.78	79.44	73.38	62.93	67.75	81.18	74.92	77.92
haghighi	77.66	75.72	76.68	79.54	77.39	78.45	70.24	65.37	67.71	78.34	75.78	77.04
marquez	78.39	75.53	76.93	79.55	76.45	77.97	70.79	64.35	67.42	78.44	74.83	76.59
pradhan	80.90	75.38	78.04	81.97	73.27	77.37	73.73	61.51	67.07	80.93	71.69	76.03
surdeanu	79.14	71.57	75.17	80.32	72.95	76.46	72.41	59.67	65.42	79.35	71.17	75.04
tsai	81.13	72.42	76.53	82.77	70.90	76.38	73.21	59.49	65.64	81.55	69.37	74.97
che	79.65	71.34	75.27	80.48	72.79	76.44	71.13	59.99	65.09	79.30	71.08	74.97
moschitti	74.95	73.10	74.01	76.55	75.24	75.89	65.92	61.83	63.81	75.19	73.45	74.31
tjongkimsang	76.79	70.01	73.24	79.03	72.03	75.37	70.45	60.13	64.88	77.94	70.44	74.00
yi	75.70	69.99	72.73	77.51	72.97	75.17	67.88	59.03	63.14	76.31	71.10	73.61
ozgencil	73.57	71.87	72.71	74.66	74.21	74.44	65.52	62.93	64.20	73.48	72.70	73.09
johansson	73.40	70.85	72.10	75.46	73.18	74.30	65.17	60.59	62.79	74.13	71.50	72.79
cohn	73.51	68.98	71.17	75.81	70.58	73.10	67.63	60.08	63.63	74.76	69.17	71.86
park	72.68	69.16	70.87	74.69	70.78	72.68	64.58	60.31	62.38	73.35	69.37	71.31
mitsumori	71.68	64.93	68.14	74.15	68.25	71.08	63.24	54.20	58.37	72.77	66.37	69.43
venkatapathy	71.88	64.76	68.14	73.76	65.52	69.40	65.25	55.72	60.11	72.66	64.21	68.17
ponzetto	71.82	61.60	66.32	75.05	64.81	69.56	66.69	52.14	58.52	74.02	63.12	68.13
lin	70.11	61.96	65.78	71.49	64.67	67.91	65.75	52.82	58.58	70.80	63.09	66.72
sutton	64.43	63.11	63.76	68.57	64.99	66.73	62.91	54.85	58.60	67.86	63.63	65.68
baseline	50.00	28.98	36.70	51.13	29.16	37.14	62.66	33.07	43.30	52.58	29.69	37.95

Table 6: Overall precision, recall and F_1 rates obtained by the 19 participating systems in the CoNLL-2005 shared task on the development and test sets. Systems sorted by F_1 score on the WSJ+Brown test set.

best individual system on the task is that of Surdeanu and Turmo (2005), which obtained F_1=75.04 on the combined test set, about 3 points below than the best performing combined system. On the development set, that system achieved a performace of 75.17 (slightly below than the 75.27 reported by Che et al. (2005) on the same dataset). According to the description papers, we find that other individual systems, from which the combined systems are constructed, performed also very well. For instance, Tsai et al. (2005) report F_1=75.76 for a base system on the development set, Màrquez et al. (2005) report F_1=75.75, Punyakanok et al. (2005) report F_1=74.76, and Haghighi et al. (2005) report F_1=74.52.

The best results in the CoNLL-2005 shared task are 10 points better than those of last year edition. This increase in performance should be attributed to a combination of the following factors: 1) training sets have been substantially enlarged; 2) predicted parse trees are available as input information; and 3) more sophisticated combination schemes have been implemented. In order to have a more clear idea of the impact of enriching the syntactic information, we refer to (Màrquez et al., 2005), who developed an individual system based only on partial parsing ("upc" input information). That system performed F_1=73.57 on the development set, which is 2.18 points below the F_1=75.75 obtained by the same architecture using full parsing, and 4.47 points below the best performing combined system on the development set (Pradhan et al., 2005b).

Comparing the results across development and WSJ test corpora, we find that, with two exceptions, all systems experienced a significant increase in performance (normally between 1 and 2 F_1 points). This fact may be attributed to the different levels of difficulty found across WSJ sections. The linguistic processors and parsers perform slightly worse in the development set. As a consequence, the matching between parse nodes and actual arguments is lower.

Regarding the evaluation using the Brown test set, all systems experienced a severe drop in performance (about 10 F_1 points), even though the baseline on the Brown test set is higher than that of the WSJ test set. As already said in previous sections, all the linguistic processors, from PoS tagging to full parsing, showed a much lower performance than in the WSJ test set, evincing that their performance cannot be extrapolated across corpora. Presumably, this fact is the main responsible of the performace drop, though we do not discard an additional overfitting effect due to the design of specific features that do not generalize well. More im-

portantly, this results impose (again) a severe criticism on the current pipelined architecture for Natural Language Processing. Error propagation and amplification through the chained modules make the final output generalize very badly when changing the domain of application.

5 Conclusion

We have described the CoNLL-2005 shared task on semantic role labeling. Contrasting with the CoNLL-2004 edition, the current edition has incorporated the use of full syntax as input to the SRL systems, much larger training sets, and cross-corpora evaluation. The first two novelties have most likely contributed to an improvement of results. The latter has evinced a major drawback of natural language pipelined architectures.

Nineteen teams have participated to the task, contributing with a variety of learning algorithms, labeling strategies, feature design and experimentation. While, broadly, all systems make use of the same basic techniques described in existing SRL literature, some novel aspects have also been explored. A remarkable aspect, common in the four top-performing systems and many other, is that of combining many individual SRL systems, each working on different syntactic structures. Combining systems improves robustness, and overcomes the limitations in coverage that working with a single, non-correct syntactic structure imposes. The best system, presented by Punyakanok et al. (2005), achieves an F_1 at 79.44 on the WSJ test. This performance, of the same order than the best reported in literature, is still far from the desired behavior of a natural language analyzer. Furthermore, the performance of such SRL module in a real application will be about ten points lower, as demonstrated in the evaluation on the sentences from Brown.

We conclude with two open questions. First, what semantic knowledge is needed to improve the quality and performance of SRL systems. Second, beyond pipelines, what type of architectures and language learning methodology ensures a robust performance of processors.

Acknowledgements

Authors would like to thank the following people and institutions. The PropBank team, and specially Martha Palmer and Benjamin Snyder, for making available PropBank-1.0 and the prop-banked Brown files. The Linguistic Data Consortium, for issuing a free evaluation license for the shared task to use the TreeBank. Hai Leong Chieu and Hwee Tou Ng, for running their Named Entity tagger on the task data. Finally, the teams contributing to the shared task, for their great enthusiasm.

This work has been partially funded by the European Community (Chil - IP506909; PASCAL - IST-2002-506778) and the Spanish Ministry of Science and Technology (Aliado, TIC2002-04447-C02).

References

Xavier Carreras and Lluís Màrquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of RANLP-2003*, Borovets, Bulgaria.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2004*.

Eugene Charniak. 2000. A maximum-entropy inspired parser. In *Proceedings of NAACL-2000*.

Wanxiang Che, Ting Liu, Sheng Li, Yuxuan Hu, and Huaijun Liu. 2005. Semantic role labeling system using maximum entropy classifier. In *Proceedings of CoNLL-2005*.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of CoNLL-2003*, Edmonton, Canada.

Trevor Cohn and Philip Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Proceedings of CoNLL-2005*.

Michael Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Charles J. Fillmore, Charles Wooters, and Collin F. Baker. 2001. Building a large lexical databank which provides deep semantics. In *Proceedings of the Pacific Asian Conference on Language, Informa tion and Computation*, Hong Kong, China.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Jesús Giménez and Lluís Màrquez. 2003. Fast and accurate part-of-speech tagging: The svm approach revisited. In *Proceedings of RANLP-2003*, Borovets, Bulgaria.

Aria Haghighi, Kristina Toutanova, and Christopher Manning. 2005. A joint model for semantic role labeling. In *Proceedings of CoNLL-2005*.

Richard Johansson and Pierre Nugues. 2005. Sparse bayesian classification of predicate arguments. In *Proceedings of CoNLL-2005*.

Chi-San Lin and Tony C. Smith. 2005. Semantic role labeling via consensus in pattern-matching. In *Proceedings of CoNLL-2005*.

Ken Litkowski. 2004. Senseval-3 task: Automatic labeling of semantic roles. In *Proceedings of the Senseval-3 ACL-SIGLEX Workshop*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19.

Lluís Màrquez, Pere Comas, Jesús Giménez, and Neus Català. 2005. Semantic role labeling as sequential tagging. In *Proceedings of CoNLL-2005*.

Tomohiro Mitsumori, Masaki Murata, Yasushi Fukuda, Kouichi Doi, and Hirohumi Doi. 2005. Semantic role labeling using support vector machines. In *Proceedings of CoNLL-2005*.

Alessandro Moschitti, Ana-Maria Giuglea, Bonaventura Coppola, and Roberto Basili. 2005. Hierarchical semantic role labeling. In *Proceedings of CoNLL-2005*.

Alessandro Moschitti. 2004. A study on convolution kernel for shallow semantic parsing. In *Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics (ACL-2004)*.

Necati Ercan Ozgencil and Nancy McCracken. 2005. Semantic role labeling using libSVM. In *Proceedings of CoNLL-2005*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Kyung-Mi Park and Hae-Chang Rim. 2005. Maximum entropy based semantic role labeling. In *Proceedings of CoNLL-2005*.

Simone Paolo Ponzetto and Michael Strube. 2005. Semantic role labeling using lexical statistical information. In *Proceedings of CoNLL-2005*.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Daniel Jurafsky. 2005a. Support vector learning for semantic argument classification. *Machine Learning. Special issue on Speech and Natural Language Processing*. To appear.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005b. Semantic role chunking combining complementary syntactic views. In *Proceedings of CoNLL-2005*.

Vasin Punyakanok, Dan Roth, Wen-Tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Vasin Punyakanok, Peter Koomen, Dan Roth, and Wen tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of CoNLL-2005*.

Mihai Surdeanu and Jordi Turmo. 2005. Semantic role labeling using complete syntactic analysis. In *Proceedings of CoNLL-2005*.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL 2003*, Sapporo, Japan.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proceedings of CoNLL-2005*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*.

Erik Tjong Kim Sang, Sander Canisius, Antal van den Bosch, and Toine Bogers. 2005. Applying spelling error correction techniques for improving semantic role labelling. In *Proceedings of CoNLL-2005*.

Tzong-Han Tsai, Chia-Wei Wu, Yu-Chun Lin, and Wen-Lian Hsu. 2005. Exploiting full parsing information to label semantic roles using an ensemble of me and svm via integer linear programming. In *Proceedings of CoNLL-2005*.

Sriram Venkatapathy, Akshar Bharati, and Prashanth Reddy. 2005. Inferring semantic roles using subcategorization frames and maximum entropy model. In *Proceedings of CoNLL-2005*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Szu-ting Yi and Martha Palmer. 2005. The integration of syntactic parsing and semantic role labeling. In *Proceedings of CoNLL-2005*.