A Shortest-path Method for Arc-factored Semantic Role Labeling

Xavier Lluís
TALP Research Center
Universitat Politècnica de Catalunya
xlluis@cs.upc.edu

Xavier Carreras
Xerox Research Centre Europe
xavier.carreras@xrce.xerox.com

Lluís Màrquez
ALT Research Group
Qatar Computing Research Institute
lmarquez@qf.org.qa

Abstract

We introduce a Semantic Role Labeling (SRL) parser that finds semantic roles for a predicate together with the syntactic paths linking predicates and arguments. Our main contribution is to formulate SRL in terms of shortest-path inference, on the assumption that the SRL model is restricted to arc-factored features of the syntactic paths behind semantic roles. Overall, our method for SRL is a novel way to exploit larger variability in the syntactic realizations of predicate-argument relations, moving away from pipeline architectures. Experiments show that our approach improves the robustness of the predictions, producing arc-factored models that perform closely to methods using unrestricted features from the syntax.

1 Introduction

Semantic role labeling (SRL) consists of finding the arguments of a predicate and labeling them with semantic roles (Gildea and Jurafsky, 2002; Màrquez et al., 2008). The arguments fill roles that answer questions of the type "who" did "what" to "whom", "how", and "why" for a given sentence predicate. Most approaches to SRL are based on a pipeline strategy, first parsing the sentence to obtain a syntactic tree and then identifying and classifying arguments (Gildea and Jurafsky, 2002; Carreras and Màrquez, 2005).

SRL methods critically depend on features of the syntactic structure, and consequently parsing mistakes can harm the quality of semantic role predictions (Gildea and Palmer, 2002). To alleviate this dependence, previous work has explored k-best parsers (Johansson and Nugues, 2008), combination systems (Surdeanu et al., 2007) or joint syntactic-semantic models (Johansson, 2009; Henderson et al., 2008; Lluís et al., 2013).

In this paper we take a different approach. In our scenario SRL is the end goal, and we assume that syntactic parsing is only an intermediate step to extract features to support SRL predictions. In this setting we define a model that, given a predicate, identifies each of the semantic roles together with the syntactic path that links the predicate with the argument. Thus, following previous work (Moschitti, 2004; Johansson, 2009), we take the syntactic path as the main source of syntactic features, but instead of just conditioning on it, we predict it together with the semantic role. The main contribution of this paper is a formulation of SRL parsing in terms of efficient shortest-path inference, under the assumption that the SRL model is restricted to arc-factored features of the syntactic path linking the argument with the predicate.

Our assumption —that features of an SRL model should factor over dependency arcs— is supported by some empirical frequencies. Table 1 shows the most frequent path patterns on CoNLL-2009 (Hajič et al., 2009) data for several languages, where a path pattern is a sequence of ascending arcs from the predicate to some ancestor, followed by descending arcs to the argument. For English the distribution of path patterns is rather simple: the majority of paths consists of a number of ascending arcs followed by zero or one descending arc. Thus a common strategy in SRL systems, formulated by Xue and Palmer (2004), is to look for arguments in the ancestors of the predicate and their direct descendants. However, in Czech and Japanese data we observe a large portion of paths with two or more descending arcs, which makes it difficult to characterize the syntactic scope in which arguments are found. Also, in the datasets for German, Czech and Chinese the three most frequent patterns cover over the 90% of all arguments. In contrast, Japanese exhibits much more variability and a long tail of infrequent types

English			German			Czech			Chinese			Japanese		
$\sum$%	%	path	$\sum$%	%	path	$\sum$%	%	path	$\sum$%	%	path	$\sum$%	%	path
63.63	63.6298	↓	77.22	77.2202	↓	63.90	63.8956	↓	78.09	78.0949	↓	37.20	37.1977	↓↓
73.97	10.3429	↑↓	93.51	16.2854	↑↓	86.26	22.3613	↓↓	85.36	7.26962	↑↓	51.52	14.3230	↓
80.63	6.65915	∘	97.43	3.92111	↑↑↓	90.24	3.98078	↑↓	91.27	5.90333	↑↑↓	60.79	9.27270	↓↓↓
85.97	5.33352	↑	98.19	0.76147	↓↓	93.95	3.71713	↓↓↓	95.93	4.66039	↑↑	70.03	9.23857	↑
90.78	4.81104	↑↑↓	98.70	0.51640	↑↑↑↓	95.48	1.52168	↑↓↓	97.53	1.60392	↑	74.17	4.13359	↓↓↓↓
93.10	2.31928	↑↑↑↓	99.17	0.46096	↑	96.92	1.44091	↑	98.28	0.75086	↑↑↑↓	76.76	2.59117	↑↑
95.19	2.09043	↑↑	99.43	0.26841	↑↓↓	97.68	0.76714	↑↑↓	98.77	0.48734	↓↓	78.82	2.06111	↑↑↓↓
96.26	1.07468	↑↑↑↑↓	99.56	0.12837	↑↑↓↓	98.28	0.59684	↓↓↓↓	99.13	0.36270	↑↑↑	80.85	2.03381	↓↓↓↓↓
97.19	0.92482	↓↓	99.67	0.10503	↑↑↑↑↓	98.60	0.31759	↑↓↓↓	99.45	0.31699	↑↑↑↑↓	82.66	1.80631	↑↓↓
97.93	0.74041	↑↑↑	99.77	0.10503	↑↑	98.88	0.28227	↑↑↓↓	99.72	0.27041	↑↑↑↑	83.71	1.05558	↑↑↑
98.41	0.48565	↑↑↑↑↑↓	99.82	0.04960	↓↓↓	99.15	0.26721	↑↑↑↓	99.82	0.10049	↓↓↓	84.74	1.02828	↑↑↑↓↓
98.71	0.29769	↑↑↑↑	99.87	0.04960	↑↑↑	99.27	0.12430	↓↓↓↓↓	99.86	0.03623	↑↓↓	85.68	0.93500	↑↑↓↓↓
98.94	0.22733	↑↑↑↑↑↑↓	99.90	0.02626	∘	99.37	0.10103	↑↑↑↑↓	99.89	0.02890	↑↑↓↓	86.61	0.93273	↓↓↓↓↓↓
99.11	0.17805	↑↓↓	99.92	0.02042	↑↑↑↓↓	99.47	0.09747	↑↑	99.92	0.02890	↑↑↑↑↑↓	87.29	0.68249	↑↑↑↑↓↓
99.27	0.15316	↓↓↓	99.94	0.02042	↑↑↑↑↑↓	99.56	0.08515	↑↑↓↓↓	99.94	0.02846	∘	87.90	0.60969	↑↓↓↓
99.39	0.12065	↑↑↑↑↑	99.95	0.01459	↑↑↓↓↓	99.63	0.07419	↑↑↑↓↓	99.96	0.02070	↑↑↑↑↑	88.47	0.56646	↑↑↓↓↓↓
99.50	0.11024	↑↑↓↓	99.96	0.01167	↓↓↓↓	99.69	0.05667	↑↓↓↓↓	99.97	0.00992	↑↑↓↓↓	89.01	0.53689	↓↓↓↓↓↓↓
99.60	0.09931	↑↑↑↑↑↑↑↓	99.97	0.00875	↑↓↓↓	99.73	0.04216	↑↑↑↑↑↓	99.98	0.00733	↑↑↑↑↑↑↓	89.49	0.48684	↑↑↑↓↓↓
99.65	0.05283	↑↓↓↓	99.98	0.00875	↑↑↑↑↑↑↓	99.76	0.02875	↑↑↑↓↓↓	99.99	0.00431	↑↑↑↑↓↓	89.94	0.45044	↑↑↑↑

Table 1: Summary of the most frequent paths on the CoNLL-2009 Shared Task datasets. ↑ indicates that we traverse a syntactic dependency upwards from a modifier to a head. ↓ is for dependencies following a descending head to modifier edge. The symbol ∘ represents that the argument is the predicate itself. We exclude from this table Catalan and Spanish as predicates and arguments are always trivially related by a single syntactic dependency that descends.

of patterns. In general it is not feasible to capture path patterns manually, and it is not desirable that a statistical system depends on rather sparse non-factored path features. For this reason in this paper we explore arc-factored models for SRL.

Our method might be specially useful in applications were we are interested in some target semantic role, i.e. retrieving agent relations for some verb, since it processes semantic roles independently of each other. Our method might also be generalizable to other kinds of semantic relations which strongly depend on syntactic patterns such as relation extraction in information extraction or discourse parsing.

2 Arc-factored SRL

We define an SRL parsing model that retrieves predicate-argument relations based on arc-factored syntactic representations of paths connecting predicates with their arguments. Throughout the paper we assume a fixed sentence $\mathbf{x} = x_1, \ldots, x_n$ and a fixed predicate index p. The SRL output is an indicator vector $\mathbf{z}$, where $\mathbf{z}_{r,a} = 1$ indicates that token a is filling role r for predicate p. Our SRL parser performs $\text{argmax}_{\mathbf{z} \in \mathcal{Z}(\mathbf{x},p)}\, s(\mathbf{x}, p, \mathbf{z})$, where $\mathcal{Z}(\mathbf{x}, p)$ defines the set of valid argument structures for p, and $s(\mathbf{x}, p, \mathbf{z})$ computes a plausibility score for $\mathbf{z}$ given $\mathbf{x}$ and p. Our first assumption is that the score function factors over role-argument pairs:

$$s(\mathbf{x}, p, \mathbf{z}) = \sum_{z_{r,a}=1} s(\mathbf{x}, p, r, a) \quad . \qquad (1)$$

Then we assume two components in the model, one that scores the role-argument pair alone, and another that considers the best (max) syntactic dependency path $\boldsymbol{\pi}$ that connects the predicate p with the argument a:

$$s(\mathbf{x}, p, r, a) = s_0(\mathbf{x}, p, r, a) + \max_{\boldsymbol{\pi}} s_{\text{syn}}(\mathbf{x}, p, r, a, \boldsymbol{\pi}) \quad . \qquad (2)$$

The model does not assume access to the syntactic structure of $\mathbf{x}$, hence in Eq. (2) we locally retrieve the maximum-scoring path for an argument-role pair. A path $\boldsymbol{\pi}$ is a sequence of dependencies $\langle h, m, l \rangle$ where h is the head, m the modifier and l the syntactic label. We further assume that the syntactic component factors over the dependencies in the path:

$$s_{\text{syn}}(\mathbf{x}, p, r, a, \boldsymbol{\pi}) = \sum_{\langle h,m,l \rangle \in \boldsymbol{\pi}} s_{\text{syn}}(\mathbf{x}, p, r, a, \langle h, m, l \rangle) \; . \qquad (3)$$

This will allow to employ efficient shortest-path inference, which is the main contribution of this paper and is described in the next section. Note that since paths are locally retrieved per role-argument pair, there is no guarantee that the set of paths across roles forms a (sub)tree.

As a final note, in this paper we follow Lluís et al. (2013) and consider a constrained space of valid argument structures $\mathcal{Z}(\mathbf{x}, p)$: (a) each role is realized at most once, and (b) each token fills at most one role. As shown by Lluís et al. (2013), this can be efficiently solved as a linear assign-

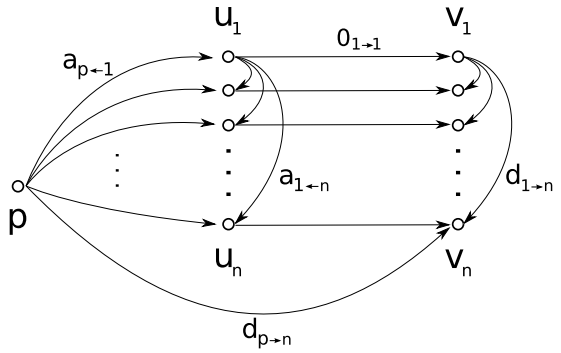

Figure 1: Graph representing all possible syntactic paths from a single predicate to their arguments. We find in this graph the best SRL using a shortest-path algorithm. Note that many edges are omitted for clarity reasons. We labeled the nodes and arcs as follows: p is the predicate and source vertex; $u_1, \ldots, u_n$ are tokens reachable by an ascending path; $v_1, \ldots, v_n$ are tokens reachable by a ascending path (possibly empty) followed by a descending path (possibly empty); $a_{i \leftarrow j}$ is an edge related to an *ascending* dependency from node u_i to node u_j; $d_{i \rightarrow j}$ is a *descending* dependency from node v_i to node v_j; $0_{i \rightarrow i}$ is a 0-weighted arc that connects the ascending portion of the path ending at u_i with the descending portion of the path starting at v_i.

ment problem as long as the SRL model factors over role-argument pairs, as in Eq. (1).

3 SRL as a Shortest-path Problem

We now focus on solving the maximization over syntactic paths in Eq. (2). We will turn it into a minimization problem which can be solved with a polynomial-cost algorithm, in our case a shortest-path method. Assume a fixed argument and role, and define $\theta_{\langle h,m,l \rangle}$ to be a *non-negative penalty* for the syntactic dependency $\langle h, m, l \rangle$ to appear in the predicate-argument path. We describe a shortest-path method that finds the path of arcs with the smaller penalty:

$$\min_{\boldsymbol{\pi}} \sum_{\langle h,m,l \rangle \in \boldsymbol{\pi}} \theta_{\langle h,m,l \rangle} \quad . \tag{4}$$

We find these paths by appropriately constructing a weighted graph $G = (V, E)$ that represents the problem. Later we show how to adapt the arc-factored model scores to be non-negative penalties, such that the solution to Eq. (4) will be the negative of the maximizer of Eq. (2).

It remains only to define the graph construction where paths correspond to arc-factored edges weighted by θ penalties. We start by noting that any path from a predicate p to an argument v_i is formed by a number of *ascending* syntactic arcs followed by a number of *descending* arcs. The ascending segment connects p to some ancestor q (q might be p itself, which implies an empty ascending segment); the descending segment connects q with v_i (which again might be empty). To compactly represent all these possible paths we define the graph as follows (see Figure 1):

1. Add node p as the *source* node of the graph.
2. Add nodes $u_1, \ldots, u_n$ for every token of the sentence except p.
3. Link every pair of these nodes u_i, u_j with a directed edge $a_{i \leftarrow j}$ weighted by the corresponding ascending arc, namely $\min_l \theta_{\langle j,i,l \rangle}$. Also add ascending edges from p to any u_i weighted by $\min_l \theta_{\langle i,p,l \rangle}$. So far we have a connected component representing all ascending path segments.
4. Add nodes $v_1, \ldots, v_n$ for every token of the sentence except p, and add edges $d_{i \rightarrow j}$ between them weighted by descending arcs, namely $\min_l \theta_{\langle i,j,l \rangle}$. This adds a second strongly-connected component representing descending path segments.
5. For each i, add an edge from u_i to v_i with weight 0. This ensures that ascending and descending path segments are connected consistently.
6. Add direct descending edges from p to all the v_i nodes to allow for only-descending paths, weighted by $\min_l \theta_{\langle p,i,l \rangle}$.

Dijkstra's algorithm (Dijkstra, 1959) will find the optimal path from predicate p to all tokens in time $O(V^2)$ (see Cormen et al. (2009) for an in-depth description). Thus, our method runs this algorithm for each possible role of the predicate, obtaining the best paths to all arguments at each run.

4 Adapting and Training Model Scores

The shortest-path problem is undefined if a negative cycle is found in the graph as we may indefinitely decrease the cost of a path by looping over this cycle. Furthermore, Dijkstra's algorithm requires all arc scores to be non-negative penalties. However, the model in Eq. (3) computes plausibility scores for dependencies, not penalties. And, if we set this model to be a standard feature-based linear predictor, it will predict unrestricted real-valued scores.

One approach to map plausibility scores to penalties is to assume a log-linear form for our

model. Let us denote by $\bar{x}$ the tuple $\langle \mathbf{x}, p, r, a \rangle$, which we assume fixed in this section. The log-linear model predicts:

$$\Pr(\langle h, m, l \rangle \mid \bar{x}) = \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle)\}}{Z(\bar{x})}\,, \tag{5}$$

where $\mathbf{f}(\bar{x}, \langle h, m, l \rangle)$ is a feature vector for an arc in the path, $\mathbf{w}$ are the parameters, and $Z(\bar{x})$ is the normalizer. We can turn predictions into non-negative penalties by setting $\theta_{\langle h,m,l \rangle}$ to be the negative log-probability of $\langle h, m, l \rangle$; namely $\theta_{\langle h,m,l \rangle} = -\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) + \log Z(\bar{x})$. Note that $\log Z(\bar{x})$ shifts all values to the non-negative side.

However, log-linear estimation of $\mathbf{w}$ is typically expensive since it requires to repeatedly compute feature expectations. Furthermore, our model as defined in Eq. (2) combines arc-factored path scores with path-independent scores, and it is desirable to train these two components jointly. We opt for a mistake-driven training strategy based on the Structured Averaged Perceptron (Collins, 2002), which directly employs shortest-path inference as part of the training process.

To do so we predict plausibility scores for a dependency directly as $\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle)$. To map scores to penalties, we define

$$\theta_0 = \max_{\langle h,m,l \rangle} \mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle)$$

and we set

$$\theta_{\langle h,m,l \rangle} = -\mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) + \theta_0 \quad .$$

Thus, θ_0 has a similar purpose as the log-normalizer $Z(\bar{x})$ in a log-linear model, i.e., it shifts the negated scores to the positive side; but in our version the normalizer is based on the max value, not the sum of exponentiated predictions as in log-linear models. If we set our model function to be

$$s_{\text{syn}}(\bar{x}, \langle h, m, l \rangle) = \mathbf{w} \cdot \mathbf{f}(\bar{x}, \langle h, m, l \rangle) - \theta_0$$

then the shortest-path method is exact.

5 Experiments

We present experiments using the CoNLL-2009 Shared Task datasets (Hajič et al., 2009), for the verbal predicates of English. Evaluation is based on precision, recall and F_1 over correct predicate-argument relations[1]. Our system uses the feature set of the state-of-the-art system by Johansson (2009), but ignoring the features that do not factor over single arcs in the path.

The focus of these experiments is to see the performance of the shortest-path method with respect to the syntactic variability. Rather than running the method with the full set of possible dependency arcs in a sentence, i.e. $O(n^2)$, we only consider a fraction of the most likely dependencies. To do so employ a probabilistic dependency-based model, following Koo et al. (2007), that computes the distribution over head-label pairs for a given modifier, $\Pr(h, l \mid \mathbf{x}, m)$. Specifically, for each modifier token we only consider the dependencies or heads whose probability is above a factor γ of the most likely dependency for the given modifier. Thus, $\gamma = 1$ selects only the most likely dependency (similar to a pipeline system, but without enforcing tree constraints), and as γ decreases more dependencies are considered, to the point where $\gamma = 0$ would select all possible dependencies. Table 2 shows the ratio of dependencies included with respect to a pipeline system for the development set. As an example, if we set $\gamma = 0.5$, for a given modifier we consider the most likely dependency and also the dependencies with probability larger than 1/2 of the probability of the most likely one. In this case the total number of dependencies is 10.3% larger than only considering the most likely one.

Table 3 shows results of the method on development data, when training and testing with different γ values. The general trend is that testing with the most restricted syntactic graph results in the best performance. However, we observe that as we allow for more syntactic variability during training, the results largely improve. Setting $\gamma = 1$ for both training and testing gives a semantic F_1 of 75.9. This configuration is similar to a pipeline approach but considering only factored features. If we allow to train with $\gamma = 0.1$ and we test with $\gamma = 1$ the results improve by 1.96 points to a semantic F_1 of 77.8 points. When syntactic variability is too large, e.g., $\gamma = 0.01$, no improvements are observed.

Finally, table 4 shows results on the verbal English WSJ test set using our best configuration

[1] Unlike in the official CoNLL-2009 evaluation, in this work we exclude the predicate sense from the features and the evaluation.

Threshold γ	1	0.9	0.5	0.1	0.01
Ratio	1	1.014	1.103	1.500	2.843

Table 2: Ratio of additional dependencies in the graphs with respect to a single-tree pipeline model ($\gamma = 1$) on development data.

Threshold	**prec** (%)	**rec** (%)	$\mathbf{F_1}$
training γ = 1			
1	77.91	73.97	75.89
0.9	77.23	74.17	75.67
0.5	73.30	**75.03**	74.16
0.1	58.22	68.75	63.05
0.01	32.83	53.69	40.74
training γ = 0.5			
1	81.17	73.57	77.18
0.9	80.74	73.78	77.10
0.5	78.40	74.79	76.55
0.1	65.76	71.61	68.56
0.01	42.95	57.68	49.24
training γ = 0.1			
1	**84.03**	72.52	**77.85**
0.9	83.76	72.66	77.82
0.5	82.75	73.33	77.75
0.1	77.25	72.20	74.64
0.01	63.90	65.98	64.92
training γ = 0.01			
1	81.62	69.06	74.82
0.9	81.45	69.19	74.82
0.5	80.80	69.80	74.90
0.1	77.92	68.94	73.16
0.01	74.12	65.92	69.78

Table 3: Results of our shortest-path system for different number of allowed dependencies showing precision, recall and F_1 on development set for the verbal predicates of the *English* language.

from the development set. We compare to the state-of-the art system by Zhao et al. (2009) that was the top-performing system for the English language in SRL at the CoNLL-2009 Shared Task. We also show the results for a shortest-path system trained and tested with $\gamma = 1$. In addition we include an equivalent pipeline system using all features, both factored and non-factored, as defined in Johansson (2009). We observe that by not being able to capture non-factored features the final performance drops by 1.6 F_1 points.

6 Conclusions

We have formulated SRL in terms of shortest-path inference. Our model predicts semantic roles together with associated syntactic paths, and assumes an arc-factored representation of the path. This property allows for efficient shortest-path algorithms that, given a predicate and a role, retrieve the most likely argument and its path.

System	**prec(%)**	**rec(%)**	$\mathbf{F_1}$
Zhao et al. 2009	86.91	81.22	83.97
Non-factored	86.96	75.92	81.06
Factored $\gamma = 1$	79.88	76.12	77.96
Factored best	85.26	74.41	79.46

Table 4: Test set results for verbal predicates of the in-domain *English* dataset. The configurations are labeled as follows. *Factored* $\gamma = 1$: our shortest-path system trained and tested with $\gamma = 1$, similar to a pipeline system but without enforcing tree constraints and restricted to arc-factored features. *Factored best*: our shortest-path system with the best results from table 3. *Non-factored*: an equivalent pipeline system that includes both factored and non-factored features.

In the experimental section we prove the feasibility of the approach. We observe that arc-factored models are in fact more restricted, with a drop in accuracy with respect to unrestricted models. However, we also observe that our method largely improves the robustness of the arc-factored method when training with a degree of syntactic variability. Overall, ours is a simple strategy to bring arc-factored models close to the performance of unrestricted models. Future work should explore further approaches to parse partial syntactic structure specific to some target semantic relations.

Acknowledgments

This work was financed by the European Commission for the XLike project (FP7-288342); and by the Spanish Government for projects Tacardi (TIN2012-38523-C02-00) and Skater (TIN2012-38584-C06-01). For a large part of this work Xavier Carreras was at the Universitat Politècnica de Catalunya under a Ramón y Cajal contract (RYC-2008-02223).

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in*

Natural Language Processing, pages 1–8. Association for Computational Linguistics, July.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. The MIT Press.

Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

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

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 239–246, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, Boulder, Colorado, USA.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of CoNLL-2008 Shared Task*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with propbank and nombank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187, Manchester, England, August. Coling 2008 Organizing Committee.

Richard Johansson. 2009. Statistical bistratal dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 561–569, Singapore, August. Association for Computational Linguistics.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic, June. Association for Computational Linguistics.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint Arc-factored Parsing of Syntactic and Semantic Dependencies. *Transactions of the Association for Computational Linguistics (TACL)*, 1(1):219–230, May.

Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159, June.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 335–342, Barcelona, Spain, July.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 88–94, Barcelona, Spain, July. Association for Computational Linguistics.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 61–66, Boulder, Colorado, June. Association for Computational Linguistics.