Priberam: A Turbo Semantic Parser with Second Order Features

André F. T. Martins[*†] **Mariana S. C. Almeida**[*†]
[*]Priberam Labs, Alameda D. Afonso Henriques, 41, 2º, 1000-123 Lisboa, Portugal
[†]Instituto de Telecomunicações, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
{atm,mla}@priberam.pt

Abstract

This paper presents our contribution to the SemEval-2014 shared task on Broad-Coverage Semantic Dependency Parsing. We employ a feature-rich linear model, including scores for first and second-order dependencies (arcs, siblings, grandparents and co-parents). Decoding is performed in a global manner by solving a linear relaxation with alternating directions dual decomposition (AD^3). Our system achieved the top score in the open challenge, and the second highest score in the closed track.

1 Introduction

The last decade saw a considerable progress in statistical modeling for dependency syntactic parsing (Kübler et al., 2009). Models that incorporate rich global features are typically more accurate, even if pruning is necessary or decoding needs to be approximate (McDonald et al., 2006; Koo and Collins, 2010; Bohnet and Nivre, 2012; Martins et al., 2009, 2013). This paper applies the same rationale to **semantic dependency parsing**, in which the output variable is a **semantic graph**, rather than a syntactic tree. We extend a recently proposed dependency parser, *TurboParser* (Martins et al., 2010, 2013), to be able to perform semantic parsing using any of the three formalisms considered in this shared task (DM, PAS, and PCEDT). The result is *TurboSemanticParser*, which we release as open-source software.[1]

We describe here a **second order model** for semantic parsing (§2). We follow prior work in semantic role labeling (Toutanova et al., 2005; Johansson and Nugues, 2008; Das et al., 2012; Flanigan et al., 2014), by adding constraints and modeling interactions among arguments within the same frame; however, we go beyond such sibling interactions to consider more complex grandparent and co-parent structures, effectively correlating different predicates. We formulate parsing as a global optimization problem and solve a relaxation through AD^3, a fast dual decomposition algorithm in which several simple local subproblems are solved iteratively (§3). Through a rich set of features (§4), we arrive at top accuracies at parsing speeds around 1,000 tokens per second, as described in the experimental section (§5).

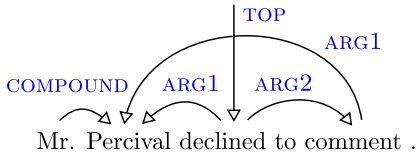

Figure 1: Example of a semantic graph in the DM formalism (sentence #22006003). We treat top nodes as a special semantic role TOP whose predicate is a dummy root symbol.

2 A Second Order Model for Parsing

Figure 1 depicts a sentence and its semantic graph. We cast semantic parsing as a structured prediction problem. Let x be a sentence and $\mathcal{Y}(x)$ the set of possible dependency graphs. We assume each candidate graph $y \in \mathcal{Y}(x)$ can be represented as a set of substructures (called **parts**) in an underlying set $\mathcal{S}$ (*e.g.*, predicates, arcs, pairs of adjacent arcs). We design a score function f which decomposes as a sum over these substructures, $f(x, y) := \sum_{s \in \mathcal{S}} f_s(x, y_s)$. We parametrize this function using a weight vector $\boldsymbol{w}$, and write each atomic function as $f_s(x, y_s) := \boldsymbol{w} \cdot \boldsymbol{\phi}_s(x, y_s)$, where $\boldsymbol{\phi}_s(x, y_s)$ is a vector of local features. The **decoding problem** consists in obtaining the best-

[1] http://labs.priberam.com/Resources/TurboSemanticParser

Algorithm 1 Decoding in an Arc-Factored Model

input: Predicate scores $\sigma_P(p)$, arc scores $\sigma_A(p \to a)$, labeled arc scores $\sigma_{LA}(p \xrightarrow{r} a)$.
Initialize semantic graph $G \leftarrow \varnothing$
for $p = 0$ **to** L **do**
  Initialize $\sigma \leftarrow \sigma_P(p)$, frame $A(p) \leftarrow \varnothing$
  for $a = 1$ **to** L **do**
    Set $r' \leftarrow \arg\max_r \sigma_{LA}(p \xrightarrow{r} a)$
    if $\sigma_A(p \to a) + \sigma_{LA}(p \xrightarrow{r'} a) > 0$ **then**
      $A(p) \leftarrow A(p) \cup \{\langle p, a, r' \rangle\}$
      $\sigma \leftarrow \sigma + \sigma_A(p \to a) + \sigma_{LA}(p \xrightarrow{r'} a)$
    end if
  end for
  if $\sigma > 0$ **then** set $G \leftarrow G \cup \{\langle p, A(p) \rangle\}$
end for
output: semantic graph G.

scored semantic graph $\widehat{y}$ given a sentence x:

$$\widehat{y} = \arg \max_{y \in \mathcal{Y}(x)} f(x, y). \quad (1)$$

Our choice of parts is given in Figure 2. The second order parts are inspired by prior work in syntactic parsing, modeling interactions for pairs of (unlabeled) dependency arcs, such as **grandparents** (Carreras, 2007) and **siblings** (Smith and Eisner, 2008; Martins et al., 2009). The main novelty is *co-parent* parts, which, to the best of our knowledge, were never considered before, as they only make sense when multiple parents are allowed.

If all parts were basic, decoding could be done independently for each predicate p, as illustrated in Algorithm 1. The total runtime, for a sentence with L words, is $O(L^2|\mathcal{R}|)$, where $\mathcal{R}$ is the set of semantic roles. Adding consecutive siblings still permits independent decoding for each predicate, but dynamic programming is necessary to decode the best argument frame, increasing the runtime to $O(L^3|\mathcal{R}|)$. The addition of consecutive co-parents, grandparents, and arbitrary siblings and co-parents breaks this independency and sets a demand for approximate decoding. Even without second-order parts, the inclusion of hard constraints (such as requiring some roles to be unique, see §3) also makes the problem harder.[2]

Rather than looking for a model in which exact decoding is tractable, which could be even more stringent for parsing semantic graphs than for dependency trees, we embrace approximate decoding strategies. Namely, our approach is based on

[2] Albeit the dynamic program could still incorporate constraints for unique roles (by appending a bit-string to the state to mark semantic roles that have been filled), runtime becomes exponential in the number of unique roles, only being feasible when this number is small.

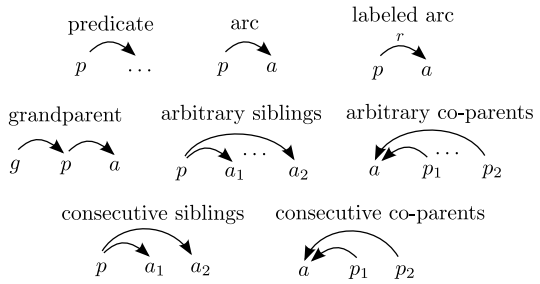

Figure 2: Parts considered in this paper. The top row illustrate the *basic parts*, representing the event that a word is a predicate, or the existence of an arc between a predicate and an argument, eventually labeled with a semantic role. Our *second-order model* looks at some pairs of arcs: arcs bearing a grandparent relationship, arguments of the same predicate, predicates sharing the same argument, and consecutive versions of these two.

dual decomposition, a class of optimization techniques that tackle the dual of combinatorial problems in a modular and extensible manner (Komodakis et al., 2007; Rush et al., 2010). We employ **alternating directions dual decomposition** (AD^3; Martins et al., 2011). Like the subgradient algorithm of Rush et al. (2010), AD^3 splits the original problem into **local subproblems**, and seeks an agreement on the overlapping variables. The difference is that the AD^3 subproblems have an additional *quadratic* term to accelerate consensus, achieving a faster convergence rate both in theory and in practice (Martins et al., 2012, 2013). For several factors (such as logic factors representing AND, OR and XOR constraints, budget constraints, and binary pairwise factors), these quadratic subproblems can be solved efficiently. For dense or structured factors, the quadratic subproblems can be solved as a sequence of local Viterbi decoding steps, via an active set method (Martins, 2014); this local decoding operation is the same that needs to be performed in the subgradient algorithm. We describe these subproblems in detail in the next section.

3 Solving the Subproblems

Predicate and Arc-Factored Parts. We capture all the basic parts with a single component. As stated in §2, local decoding in this component has a runtime of $O(L^2|\mathcal{R}|)$, by using Algorithm 1.

Unique Roles. We assume some roles are unique, *i.e.*, they can occur at most once for the

same predicate.[3] To cope with unique roles, we add hard constraints of the kind

$$\sum_a \mathbb{I}(p \xrightarrow{r} a \in y) \leq 1, \ \forall p, \forall r \in \mathcal{R}_{\text{uniq}}, \quad (2)$$

where $\mathcal{R}_{\text{uniq}}$ is the set of unique roles. This set is obtained from the training data by looking at the roles that never occur multiple times in the gold argument frames.[4] The constraint above corresponds to a ATMOSTONE factor, which is built-in in AD^3 and can be decoded in linear time (rendering the runtime $O(L^2|\mathcal{R}_{\text{uniq}}|)$ when aggregating all such factors). These have also been used by Das et al. (2012) in frame-semantic parsing.

Grandparents, Arbitrary Siblings and Coparents. The second-order parts in the middle row of Figure 2 all involve the simultaneous inclusion of a pair of arcs, without further dependency on the remaining arcs. We handle each of these parts using a simple pairwise factor (called PAIR in the AD^3 toolkit). The total runtime to locally decode these factors is $O(L^3)$.

Predicate Automata. To handle consecutive siblings, we adapt the simple head automaton model (Alshawi, 1996; Smith and Eisner, 2008; Koo et al., 2010) to semantic parsing. We introduce one automaton for each predicate p and attachment direction (left or right). We describe right-side predicate automata; their left-side counterparts are analogous. Let $\langle a_0, a_1, \ldots, a_{k+1} \rangle$ be the sequence of right modifiers of p, with $a_0 = \text{START}$ and $a_{k+1} = \text{END}$. Then, we have the following component capturing consecutive siblings:

$$f^{\text{CSIB}}_{p,\rightarrow}(p \rightarrow a_1, \ldots, p \rightarrow a_k) = \sum_{j=1}^{k+1} \sigma_{\text{CSIB}}(p, a_{j-1}, a_j). \quad (3)$$

Maximizing $f^{\text{CSIB}}_{p,\rightarrow}$ via dynamic programming has a cost of $O(L^2)$, yielding $O(L^3)$ total runtime.

Argument Automata. For consecutive coparents, we introduce another automaton which is analogous to the predicate automaton, but where arrows are reversed. Let $\langle p_0, p_1, \ldots, p_{k+1} \rangle$ be the sequence of right predicates that take a as argument (the left-side case is analagous), with $p_0 = \text{START}$ and $p_{k+1} = \text{END}$. We define:

$$f^{\text{CCP}}_{a,\leftarrow}(a \leftarrow p_1, \ldots, a \leftarrow p_k) = \sum_{j=1}^{k+1} \sigma_{\text{CCP}}(a, p_{j-1}, p_j). \quad (4)$$

The total runtime is also $O(L^3)$.

4 Features

We define binary features for each part represented in Figure 2. Most of the features are taken from *TurboParser* (Martins et al., 2013), while others are inspired by the semantic parser of Johansson and Nugues (2008). Those features marked with † require information from the dependency syntactic parser, and are only used in the open track.[5]

Predicate Features. Our predicate features are:

- PREDWORD, PREDLEMMA, PREDPOS. Lexical form, lemma, and POS tag of the predicate.
- PREDREL.† Syntactic dependency relation between the predicate and its head.
- PREDHEADWORD/POS.† Form and POS tag of the predicate syntactic head, conjoined with the predicate word and POS tag.
- PREDMODWORD/POS/REL.† Form, POS tag, and dependency relation of the predicate syntactic dependents, conjoined with the predicate word and POS tag.

Arc Features. All features above, plus the following (conjoined with arc direction and label):

- ARGWORD, ARGLEMMA, ARGPOS. The lexical form, lemma, and POS tag of the argument.
- ARGREL.† Syntactic dependency relation between the argument and its head.
- LEFTWORD/POS,† RIGHTWORD/POS.† Form/POS tag of the leftmost/rightmost dependent of the argument, conjoined with the predicate word and POS tag.
- LEFTSIBWORD/POS,† RIGHTSIBWORD/POS.† Form/POS tag of the left/right sibling of the argument, conjoined with the predicate tag.
- PREDCONTEXTWORD, PREDCONTEXTPOS, PREDCONTEXTLEMMA. Word, POS, and lemma on the left and right context of the predicate (context size is 2).
- PREDCONTEXTPOSBIGRAM/TRIGRAM. Bigram and trigram of POS tags on the left and right side of the predicate.
- PREDVOICE.† Predicate voice: active, passive, or none. Determined from the syntactic dependency tree as in Johansson and Nugues (2008).

[3] Such roles have been called "deterministic" by Flanigan et al. (2014).

[4] For PAS, all 43 roles were found unique; for DM, this number is 40 out of 52, and for PCEDT only 3 out of 69.

[5] For the open track, the only external information used by our system were the provided automatic dependency trees.

- PREDWORDARGWORD, PREDWORDARGPOS, PREDPOSARGWORD, PREDPOSARGPOS. Predicate word/tag conjoined with argument word/tag.
- PREDARGPOSCONTEXT. Several features conjoining the POS of words surrounding the predicate and argument (similar to the contextual features in McDonald et al. (2005)).
- EXACTARCLENGTH, BINNEDARCLENGTH. Exact and binned arc length (distance between predicate and argument), conjoined with the predicate and argument POS tags.
- POSINBETWEEN, WORDINBETWEEN. POS and forms between the predicate and argument, conjoined with their own POS tags and forms.
- RELPATH,† POSPATH.† Path in the syntactic dependency tree between the predicate and the argument. The path is formed either by dependency relations or by POS tags.

Second Order Features. These involve a predicate, an argument, and a "companion word" (which can be a second argument, in the case of siblings, a second predicate, for co-parents, or the argument of another argument, for grandparents). In all cases, features are of the following kind:

- POSTRIPLET. POS tags of the predicate, the argument, and the companion word.
- UNILEXICAL. One word form (for the predicate/argument/companion) and two POS tags.
- BILEXICAL. One POS tag (for the predicate/argument/companion) and two word forms.
- PAIRWISE. Backed-off pair features for the companion word form/POS tag and the word form/POS of the predicate/argument.

5 Experimental Results

All models were trained by running 10 epochs of max-loss MIRA with $C = 0.01$ (Crammer et al., 2006). The cost function takes into account mismatches between predicted and gold dependencies, with a cost c_P on labeled arcs incorrectly predicted (false positives) and a cost c_R on gold labeled arcs that were missed (false negatives). These values were set through cross-validation in the dev set, yielding $c_P = 0.4$ and $c_R = 0.6$ in all runs, except for the DM and PCEDT datasets in the closed track, for which $c_P = 0.3$ and $c_R = 0.7$.

To speed up decoding, we discard arcs whose posterior probability is below 10^{-4}, according to a probabilistic unlabeled first-order pruner. Table 1 shows a significant reduction of the search space with a very small drop in recall.

Table 2 shows our final results in the test set, for a model trained in the train and development partitions. Our system achieved the best score in the open track (an LF score of 86.27%, averaged over DM, PAS, and PCEDT), and the second best in the closed track, after the Peking team. Overall, we observe that the precision and recall in PCEDT are far below the other two formalisms, but this difference is much smaller when looking at unlabeled scores. Comparing the results in the closed and open tracks, we observe a consistent improvement in the three formalisms of around 1% in F_1 from using syntactic information. While this confirms previous findings that syntactic features are important in semantic role labeling (Toutanova et al., 2005; Johansson and Nugues, 2008), these improvements are less striking than expected. We conjecture this is due to the fact that our model in the closed track already incorporates a variety of contextual features which are nearly as informative as those extracted from the dependency trees.

Finally, to assess the importance of the second order features, Table 3 reports experiments in the dev-set that progressively add several groups of features, along with runtimes. We can see that siblings, co-parents, and grandparents all provide valuable information that improves the final scores (with the exception of the PCEDT labeled scores, where the difference is negligible). This comes at only a small cost in terms of runtime, which is around 1,000 tokens per second for the full models.

	UR	# UA/tok	LR	# LA/tok
DM	99.33	3.5 (13.4%)	99.22	34.4 (2.5%)
PAS	99.53	3.3 (12.5%)	99.49	20.8 (1.9%)
PCEDT	99.03	2.1 (8.2%)	98.77	54.5 (3.0%)

Table 1: Pruner statistics in the dev-set, for the open track. Shown are oracle recall scores, considering both unlabeled (UR) and labeled arcs (LR); and the averaged number of unlabeled and labeled arcs per token that remained after the pruning stage (# UA/tok and # LA/tok). In brackets, we show the fraction of unlabeled/labeled arcs that survived the pruning.

	UP	UR	UF	LP	LR	LF
DM, closed	90.14	88.65	89.39	88.82	87.35	88.08
PAS, closed	93.18	91.12	92.14	91.95	89.92	90.93
PCEDT, closed	90.21	85.51	87.80	78.80	74.70	76.70
average, closed	–	–	89.77	–	–	85.24
DM, open	91.41	89.26	90.32	90.23	88.11	89.16
PAS, open	93.62	92.01	92.81	92.56	90.97	91.76
PCEDT, open	91.58	86.61	89.03	80.14	75.79	77.90
average, open	–	–	90.72	–	–	86.27

Table 2: Submitted results for the closed and open tracks. For comparison, the best-performing system in the closed track (Peking) obtained averaged UF and LF scores of 91.03% and 85.91%, respectively.

	UF	LF	Tok/sec
DM, arc-factored	89.90	88.96	1,681
DM, arc-factored, pruned	89.85	88.90	2,642
+siblings	90.34	89.34	1,838
+co-parents	90.80	89.76	1,073
+grandparent (full)	90.95	89.90	955
PAS, arc-factored	92.34	91.40	1,927
PAS, arc-factored, pruned	92.35	91.40	2,914
+siblings	92.45	91.45	2,106
+co-parents	92.71	91.71	1,104
+grandparent (full)	92.87	91.87	1,043
PCEDT, arc-factored	87.90	79.90	1,558
PCEDT, arc-factored, pruned	87.74	79.83	2,906
+siblings	88.46	79.98	2,066
+co-parents	90.17	79.90	1,531
+grandparent (full)	90.18	80.03	1,371

Table 3: Results in the dev-set for the open track, progressively adding several groups of features, until the full model is obtained. We report unlabeled/labeled F_1 and parsing speeds in tokens per second. Our speeds include the time necessary for pruning, evaluating features, and decoding, as measured on a Intel Core i7 processor @3.4 GHz.

6 Conclusions

We have described a system for broad-coverage semantic dependency parsing. Our system, which is inspired by prior work in syntactic parsing, implements a linear model with second-order features, being able to model interactions between siblings, grandparents and co-parents. We have shown empirically that second-order features have an impact in the final scores. Approximate decoding was performed via alternating directions dual decomposition (AD^3), yielding fast runtimes of around 1,000 tokens per second.

Acknowledgements

We would like to thank the reviewers for their helpful comments. This work was partially supported by the EU/FEDER programme, QREN/POR Lisboa (Portugal), under the Intelligo project (contract 2012/24803) and by a FCT grant PTDC/EEI-SII/2312/2012.

References

Hiyan Alshawi. 1996. Head automata and bilingual tiling: Translation with minimal representations. In *Proc. of Annual Meeting of the Association for Computational Linguistics*, pages 167–176.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. of the Empirical Methods in Natural Language Processing*, pages 1455–1465.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *International Conference on Natural Language Learning*, pages 957–961.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An Exact Dual Decomposition Algorithm for Shallow Semantic Parsing with Constraints. In *Proc. of First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 209–217.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, pages 1426–1436.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. *International Conference on Natural Language Learning*, pages 183–187.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. of International Conference on Computer Vision*, pages 1–8.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proc. of Annual Meeting of the Association for Computational Linguistics*, pages 1–11.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. of Empirical Methods for Natural Language Processing*, pages 1288–1298.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency parsing*. Morgan & Claypool Publishers.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise Integer Linear Programming Formulations for Dependency Parsing. In *Proc. of Annual Meeting of the Association for Computational Linguistics*, pages 342–350.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turbo Parsers: Dependency Parsing by Approximate Variational Inference. In *Proc. of Empirical Methods for Natural Language Processing*, pages 34–44.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual Decomposition with Many Overlapping Components. In *Proc. of Empirical Methods for Natural Language Processing*, pages 238–249.

André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. 2012. Alternating directions dual decomposition. Arxiv preprint arXiv:1212.6550.

André F. T. Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, pages 617–622.

André F. T. Martins. 2014. AD^3: A Fast Decoder for Structured Prediction. In S. Nowozin, P. Gehler, J. Jancsary, and C. Lampert, editors, *Advanced Structured Prediction*. MIT Press, Cambridge, MA, USA.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of Annual Meeting of the Association for Computational Linguistics*, pages 91–98.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of International Conference on Natural Language Learning*, pages 216–220.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proc. of Empirical Methods for Natural Language Processing*.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of Empirical Methods for Natural Language Processing*, pages 145–156.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *ACL*, pages 589–596.