CMU: Arc-Factored, Discriminative Semantic Dependency Parsing

Sam Thomson   Brendan O'Connor   Jeffrey Flanigan   David Bamman
Jesse Dodge   Swabha Swayamdipta   Nathan Schneider   Chris Dyer   Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{sthomson,brenocon,jflanigan,dbamman,jessed,
swabha,nschneid,cdyer,nasmith}@cs.cmu.edu

Abstract

We present an arc-factored statistical model for semantic dependency parsing, as defined by the SemEval 2014 Shared Task 8 on Broad-Coverage Semantic Dependency Parsing. Our entry in the open track placed second in the competition.

1 Introduction

The task of broad coverage semantic dependency parsing aims to provide a shallow semantic analysis of text not limited to a specific domain. As distinct from deeper semantic analysis (e.g., parsing to a full lambda-calculus logical form), shallow semantic parsing captures relationships between pairs of words or concepts in a sentence, and has wide application for information extraction, knowledge base population, and question answering (among others).

We present here two systems that produce semantic dependency parses in the three formalisms of the SemEval 2014 Shared Task 8 on Broad-Coverage Semantic Dependency Parsing (Oepen et al., 2014). These systems generate parses by extracting features for each potential dependency arc and learning a statistical model to discriminate between good arcs and bad; the first treats each labeled edge decision as an independent multiclass logistic regression (§3.2.1), while the second predicts arcs as part of a graph-based structured support vector machine (§3.2.2). Common to both models is a rich set of features on arcs, described in §3.2.3. We include a discussion of features found to have no discernable effect, or negative effect, during development (§4).

Our system placed second in the open track of the Broad-Coverage Semantic Dependency Parsing task (in which output from syntactic parsers and other outside resources *can* be used). We present our results in §5.

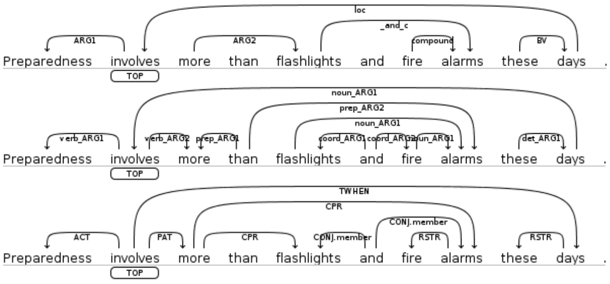
loc
_and_c
ARG1
ARG2
compound
BV
Preparedness involves more than flashlights and fire alarms these days .
TOP
noun_ARG1
prep_ARG2
noun_ARG1
verb_ARG1
verb_ARG2
prep_ARG1
coord_ARG1
coord_ARG2
noun_ARG1
det_ARG1
Preparedness involves more than flashlights and fire alarms these days .
TOP
TWHEN
CPR
CONJ.member
ACT
PAT
CPR
CONJ.member
RSTR
RSTR
Preparedness involves more than flashlights and fire alarms these days .
TOP

Figure 1: Example annotations for DM (top), PAS (middle), and PCEDT (bottom).

2 Formalisms

The Shared Task 8 dataset consists of annotations of the WSJ Corpus in three different semantic dependency formalisms. DM is derived from LinGO English Resource Grammar (ERG) annotations in DeepBank (Flickinger et al., 2012). PAS is derived from the Enju HPSG treebank using the conversion rules of Miyao et al. (2004). PCEDT is derived from the tectogrammatical layer of the Prague Czech-English Dependency Treebank (Hajič, 1998). See Figure 1 for an example.

The three formalisms come from very different linguistic theories, but all are represented as labeled directed graphs, with words as vertices, and all have "top" annotations, corresponding roughly to the semantic focus of the sentence. (A "top" need not be a root of the graph.) This allows us to use the same machinery (§3) for training and testing statistical models for the three formalisms.

3 Models

We treat the problem as a three-stage pipeline. The first stage prunes words by predicting whether they have any incoming or outgoing edges at all (§3.1); if a word does not, then it is not considered for any attachments in later stages. The second stage predicts where edges are present, and their labels (§3.2). The third stage predicts whether a predicate word is a *top* or not (§3.3). Formalisms sometimes annotate more than one "top" per sentence, but we found that we achieve the best performance on all formalisms by predicting only the one best-scoring "top" under the model.

3.1 Singleton Classification

For each formalism, we train a classifier to recognize *singletons*, nodes that have no parents or children. (For example, punctuation tokens are often singletons.) This makes the system faster without affecting accuracy. For singleton prediction, we use a token-level logistic regression classifier, with features including the word, its lemma, and its part-of-speech tag. If the classifier predicts a probability of 99% or higher the token is pruned; this removes around 10% of tokens. (The classifier performs differently on different formalisms; on PAS it has perfect accuracy, while on DM and PCEDT accuracy is in the mid-90's.)

3.2 Edge Prediction

In the second stage of the pipeline, we predict the set of labeled directed edges in the graph. We use the same set of edge-factored features (§3.2.3) in two alternative models: an edge-independent multiclass logistic regression model (LOGISTICEDGE, §3.2.1); and a structured SVM (Taskar et al., 2003; Tsochantaridis et al., 2004) that enforces a *determinism* constraint for certain labels, which allows each word to have at most one outgoing edge with that label (SVMEDGE, §3.2.2). For each formalism, we trained both models with varying features enabled and hyperparameter settings and submitted the configuration that produced the best labeled F_1 on the development set. For DM and PCEDT, this was LOGISTICEDGE; for PAS, this was SVMEDGE. We report results only for the submitted configurations, with different features enabled. Due to time constraints, full hyperparameter sweeps and comparable feature sweeps were not possible.

3.2.1 LOGISTICEDGE Parser

The LOGISTICEDGE model considers only token index pairs (i, j) where $|i - j| \leq 10$, $i \neq j$, and both t_i and t_j have been predicted to be non-singletons by the first stage. Although this prunes some gold edges, among the formalisms, 95%–97% of all gold edges are between tokens of distance 10 or less. Both directions $i \rightarrow j$ and $j \rightarrow i$ are considered between every pair.

Let L be the set of $K + 1$ possible output labels: the formalism's original K edge labels, plus the additional label NOEDGE, which indicates that no edge exists from i to j. The model treats every pair of token indices (i, j) as an independent multiclass logistic regression over output space L. Let x be an input sentence. For candidate parent index i, child index j, and edge label ℓ, we extract a feature vector $\boldsymbol{f}(x, i, j, \ell)$, where ℓ is conjoined with every feature described in §3.2.3. The multiclass logistic regression model defines a distribution over L, parametrized by weights $\boldsymbol{\phi}$:

$$P(\ell \mid \boldsymbol{\phi}, x, i, j) = \frac{\exp\{\boldsymbol{\phi} \cdot \boldsymbol{f}(x, i, j, \ell)\}}{\sum_{\ell' \in L} \exp\{\boldsymbol{\phi} \cdot \boldsymbol{f}(x, i, j, \ell')\}}.$$

$\boldsymbol{\phi}$ is learned by minimizing total negative log-likelihood of the above (with weighting; see below), plus ℓ_2 regularization. AdaGrad (Duchi et al., 2011) is used for optimization. This seemed to optimize faster than L-BFGS (Liu and Nocedal, 1989), at least for earlier iterations, though we did no systematic comparison. Stochastic gradient steps are applied one at a time from individual examples, and a gradient step for the regularizer is applied once per epoch.

The output labels have a class imbalance; in all three formalisms, there are many more NOEDGE examples than true edge examples. We improved F_1 performance by downweighting NOEDGE examples through a weighted log-likelihood objective, $\sum_{i,j} \sum_{\ell} w_\ell \log P(\ell \mid \boldsymbol{\phi}, x, i, j)$, with $w_{\text{NOEDGE}} = 0.3$ (selected on development set) and $w_\ell = 1$ otherwise.

Decoding: To predict a graph structure at test-time for a new sentence, the most likely edge label is predicted for every candidate (i, j) pair of unpruned tokens. If an edge is predicted for both directions for a single (i, j) pair, only the edge with the higher score is chosen. (There are no such bidirectional edges in the training data.) This post-processing actually did not improve accuracy on DM or PCEDT; it did improve PAS by ≈0.2% absolute F_1, but we did not submit LOGISTICEDGE for PAS.

3.2.2 SVMEDGE Parser

In the SVMEDGE model, we use a structured SVM with a determinism constraint. This constraint ensures that each word token has at most one outgoing edge for each label in a set of deterministic labels L_d. For example, in DM a predicate never has more than one child with edge label “ARG1.” L_d was chosen to be the set of edges that were $> 99.9\%$ deterministic in the training data.[1]

Consider the fully dense graph of all edges between all words predicted as not singletons by the singleton classifier §3.1 (in all directions with all possible labels). Unlike LogisticEdge, the label set L does not include an explicit NoEdge label. If $\boldsymbol{\psi}$ denotes the model weights, and $\boldsymbol{f}$ denotes the features, then an edge from i to j with label ℓ in the dense graph has a weight $c(i, j, \ell)$ assigned to it using the linear scoring function $c(i, j, \ell) = \boldsymbol{\psi} \cdot \boldsymbol{f}(x, i, j, \ell)$.

Decoding: For each node and each label ℓ, if $\ell \in L_d$, the decoder adds the highest scoring outgoing edge, if its weight is positive. For $\ell \notin L_d$, every outgoing edge with positive weight is added. This procedure is guaranteed to find the highest scoring subgraph (largest sum of edge weights) of the dense graph subject to the determinism constraints. Its runtime is $O(n^2)$.

The model weights are trained using the structured SVM loss. If x is a sentence and y is a graph over that sentence, let the features be denoted $\boldsymbol{f}(x, y) = \sum_{(i,j,\ell)\in y} \boldsymbol{f}(x, i, j, \ell)$. The SVM loss for each training example (x_i, y_i) is:

$$-\boldsymbol{\psi}^\top \boldsymbol{f}(x_i, y_i) + \max_y \boldsymbol{\psi}^\top \boldsymbol{f}(x_i, y) + cost(y, y_i)$$

where $cost(y, y_i) = \alpha|y \setminus y_i| + \beta|y_i \setminus y|$. α and β trade off between precision and recall for the edges (Gimpel and Smith, 2010). The loss is minimized with AdaGrad using early-stopping on a development set.

3.2.3 Edge Features

Table 1 describes the features we used for predicting edges. These features were computed over an edge e with parent token s at index i and child token t at index j. Unless otherwise stated, each feature template listed has an indicator feature that fires for each value it can take on. For the submitted results, LogisticEdge uses all features except Dependency Path v2, POS Path, and Distance Thresholds, and SvmEdge uses all features except Dependency Path v1. This was due to SvmEdge being faster to train than LogisticEdge when including POS Path features, and due to time constraints for the submission we were unable to retrain LogisticEdge with these features.

Tokens: The tokens s and t themselves.
Lemmas: Lemmas of s and t.
POS tags: Part of speech tags of s and t.
Linear Order: Fires if $i < j$.
Linear Distance: $i - j$.
Dependency Path v1 (LogisticEdge only): The concatenation of all POS tags, arc labels and up/down directions on the path in the syntactic dependency tree from s to t. Conjoined with s, with t, and without either.
Dependency Path v2 (SvmEdge only): Same as Dependency Path v1, but with the lemma of s or t instead of the word, and substituting the token for any “IN” POS tag.
Up/Down Dependency Path: The sequence of upward and downward moves needed to get from s to t in the syntactic dependency tree.
Up/Down/Left/Right Dependency Path: The unlabeled path through the syntactic dependency tree from s to t, annotated with whether each step through the tree was up or down, and whether it was to the right or left in the sentence.
Is Parent: Fires if s is the parent of t in the syntactic dependency parse.
Dependency Path Length: Distance between s and t in the syntactic dependency parse.
POS Context: Concatenated POS tags of tokens at $i-1$, i, $i+1$, $j-1$, j, and $j+1$. Concatenated POS tags of tokens at $i-1$, i, $j-1$, and j. Concatenated POS tags of tokens at i, $i+1$, j, and $j+1$.
Subcategorization Sequence: The sequence of dependency arc labels out of s, ordered by the index of the child. Distinguish left children from right children. If t is a direct child of s, distinguish its arc label with a “+”. Conjoin this sequence with the POS tag of s.
Subcategorization Sequence with POS: As above, but add the POS tag of each child to its arc label.
POS Path (SvmEdge only): Concatenated POS tags between and including i and j. Conjoined with head lemma, with dependent lemma, and without either.
Distance Thresholds (SvmEdge only): Fires for every integer between 1 and $\lfloor \log(|i-j|+1)/\log(1.39) \rfloor$ inclusive.

Table 1: Features used in edge prediction

3.2.4 Feature Hashing

The biggest memory usage was in the map from feature names to integer indices during feature extraction. For experimental expedience, we implemented multitask feature hashing (Weinberger et al., 2009), which hashes feature names to indices, under the theory that errors due to collisions tend to cancel. No drop in accuracy was observed.

3.3 Top Prediction

We trained a separate token-level binary logistic regression model to classify whether a token’s node had the “top” attribute or not. At decoding time, all predicted predicates (i.e., nodes where there is at

[1] By this we mean that of the nodes that have at least one outgoing ℓ edge, 99.9% of them have only one outgoing ℓ edge. For DM, $L_d = L \setminus \{$“_and_c,” “_or_c,” “_then_c,” “loc,” “mwe,” “subord”$\}$; for PAS, $L_d = L$; and for PCEDT, $L_d = \{$“DPHR,” “INTF,” “VOCAT”$\}$.

least one outbound edge) are possible candidates to be "top"; the classifier probabilities are evaluated, and the highest-scoring node is chosen to be "top." This is suboptimal, since some graphs have multiple tops (in PCEDT this is more common); but selection rules based on probability thresholds gave worse F_1 performance on the dev set. For a given token t at index i, the top classifier's features included t's POS tag, i, those two conjoined, and the depth of t in the syntactic dependency tree.

4 Negative Results

We followed a forward-selection process during feature engineering. For each potential feature, we tested the current feature set versus the current feature set plus the new potential feature. If the new feature did not improve performance, we did not add it. We list in table 2 some of the features which we tested but did not improve performance.

In order to save time, we ran these feature selection experiments on a subsample of the training data, for a reduced number of iterations. These results thus have a strong caveat that the experiments were not exhaustive. It may be that some of these features could help under more careful study.

Word vectors: Features derived from 64-dimensional vectors from (Faruqui and Dyer, 2014), including the concatenation, difference, inner product, and element-wise multiplication of the two vectors associated with a parent-child edge. We also trained a Random Forest on the word vectors using Liaw and Wiener's (2002) *R* implementation. The predicted labels were then used as features in LogisticEdge.
Brown clusters Features derived from Brown clusters (Brown et al., 1992) trained on a large corpus of web data. Parent, child, and conjoined parent-child edge features from cluster prefixes of length 2, 4, 6, 8, 10, and 12. Conjunctions of those features with the POS tags of the parent and child tokens.
Active/passive: Active/passive voice feature (as in Johansson and Nugues (2008)) conjoined with both the Linear Distance features and the Subcategorization Sequence features. Voice information may already be captured by features from the Stanford dependency–style parses, which include passivization information in arc labels such as *nsubjpass* and *auxpass* (de Marneffe and Manning, 2008).
Connectivity constraint: Enforcing that the graph is connected (ignoring singletons), similar to Flanigan et al. (2014). Almost all semantic dependency graphs in the training data are connected (ignoring singletons), but we found that enforcing this constraint significantly hurt precision.
Tree constraint: Enforces that the graph is a tree. Unsurprisingly, we found that enforcing a tree constraint hurt performance.

Table 2: Features and constraints giving negative results.

5 Experimental Setup

We participated in the Open Track, and used the syntactic dependency parses supplied by the organizers. Feature engineering was performed on a development set (§20), training on §§00–19. We evaluate labeled precision (LP), labeled recall (LR), labeled F_1 (LF), and labeled whole-sentence match (LM) on the held-out test data using the evaluation script provided by the organizers. LF was averaged over the formalisms to determine the winning system. Table 3 shows our scores.

	LP	LR	LF	LM
DM	0.8446	0.8348	0.8397	0.0875
PAS	0.9078	0.8851	0.8963	0.2604
PCEDT	0.7681	0.7072	0.7364	0.0712
Average	0.8402	0.8090	0.8241	0.1397

Table 3: Labeled precision (LP), recall (LR), F_1 (LF), and whole-sentence match (LM) on the held-out test data.

6 Conclusion and Future Work

We found that feature-rich discriminative models perform well at the task of mapping from sentences to semantic dependency parses. While our final approach is fairly standard for work in parsing, we note here additional features and constraints which did not appear to help (contrary to expectation). There are a number of clear extensions to this work that could improve performance. While an edge-factored model allows for efficient inference, there is much to be gained from **higher-order features** (McDonald and Pereira, 2006; Martins et al., 2013). The amount of information shared between the three formalisms suggests that a **multi-task learning** (Evgeniou and Pontil, 2004) framework could lead to gains. And finally, there is additional structure in the formalisms which could be exploited (such as the deterministic processes by which an original PCEDT tree annotation was converted into a graph); formulating more subtle **graph constraints** to capture this a priori knowledge could lead to improved performance. We leave such explorations to future work.

Acknowledgements

We are grateful to Manaal Faruqui for his help in word vector experiments, and to reviewers for helpful comments. The research reported in this paper was sponsored by the U.S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533, DARPA grant FA8750-12-2-0342 funded under the DEFT program, U.S. NSF grants IIS-1251131 and IIS-1054319, and Google's support of the Reading is Believing project at CMU.

References

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proc. of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Manchester, UK.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multitask learning. In *Proc. of KDD*, pages 109–117. Seattle, WA, USA.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. of EACL*, pages 462–471. Gothenburg, Sweden.

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 ACL*, pages 1426–1436. Baltimore, MD, USA.

Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank: a dynamically annotated treebank of the Wall Street Journal. In *Proc. of the Eleventh International Workshop on Treebanks and Linguistic Theories*, pages 85–96. Lisbon, Portugal.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin training for structured log-linear models. Technical Report CMU-LTI-10-008, Carnegie Mellon University. URL http://lti.cs.cmu.edu/sites/default/files/research/reports/2010/cmulti10008.pdf.

Jan Hajič. 1998. Building a syntactically annotated corpus: the Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning. Studies in Honour of Jarmila Panevová*, pages 106–132. Prague Karolinum, Charles University Press, Prague.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of PropBank. In *Proc. of EMNLP*, pages 69–78. Honolulu, HI, USA.

Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomForest. *R News*, 2(3):18–22. URL http://cran.r-project.org/web/packages/randomForest/.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528.

André F. T. Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*, pages 617–622. Sofia, Bulgaria.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*, pages 81–88. Trento, Italy.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn Treebank. In *Proc. of IJCNLP*, pages 684–693. Hainan Island, China.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*. Dublin, Ireland.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin Markov networks. In *Proc. of NIPS*, pages 25–32. Vancouver, British Columbia, Canada.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*, pages 104–111. Banff, Alberta, Canada.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proc. of ICML*, pages 1113–1120. Montreal, Quebec, Canada.