CMILLS: Adapting Semantic Role Labeling Features to Dependency Parsing

Chad Mills
University of Washington
Guggenheim Hall, 4th Floor
Seattle, WA 98195, USA
chills@uw.edu

Gina-Anne Levow
University of Washington
Guggenheim Hall, 4th Floor
Seattle, WA 98195, USA
levow@uw.edu

Abstract

We describe a system for semantic role labeling adapted to a dependency parsing framework. Verb arguments are predicted over nodes in a dependency parse tree instead of nodes in a phrase-structure parse tree. Our system participated in SemEval-2015 shared Task 15, Subtask 1: CPA parsing and achieved an F-score of 0.516. We adapted features from prior semantic role labeling work to the dependency parsing paradigm, using a series of supervised classifiers to identify arguments of a verb and then assigning syntactic and semantic labels. We found that careful feature selection had a major impact on system performance. However, sparse training data still led rule-based systems like the baseline to be more effective than learning-based approaches.

1 Introduction

We describe our submission to the SemEval-2015 Task 15, Subtask 1 on Corpus Pattern Analysis (Baisa et al. 2015). This task is similar to semantic role labeling but with arguments based on nodes in dependency parses instead of a syntactic parse tree. The verb's arguments are identified and labeled with both their syntactic and semantic roles.

For example, consider the sentence "But he said Labour did not agree that Britain could or should abandon development, either for itself or for the developing world." This subtask involves taking that sentence and making the following determinations relative to the given verb "abandon":

- "Britain" is the syntactic subject of "abandon" and falls under the "Institution" semantic type
- "development" is the syntactic object of "abandon" and is of semantic type "Activity"

We organize the remainder of our paper as follows: Section 2 describes our system, Section 3 presents experiments, and Section 4 concludes.

2 System Description

Our system consists of a pipelined five-component system plus source data and resources. A system diagram is shown in Figure 1. A cascading series of MaxEnt classifiers are used to identify arguments, their syntactic labels, and then their semantic labels. Each token in an input sentence was a training example.

Sketch Engine (Kilgarriff 2014) was used to help with featurization. All sentences in the training data were parsed and POS tagged using the Stanford CoreNLP tools (Manning et al. 2014). This data was used to generate features which are then supplied to an Argument Identification Classifier (AIC) that identifies whether or not a particular token is one of the relevant verb's arguments.

For the tokens identified as arguments to the verb, a Syntax Classifier identifies the syntactic role of the token. This is done using a multi-class MaxEnt model with the same features as the AIC plus features derived from the AIC's predictions. A similar Semantics Classifier follows, taking the Syntax

Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 433–437,
Denver, Colorado, June 4-5, 2015.

Classifier's features and output. Finally, a Semantics Consistency Heuristic Filter is applied to clean up some of the predictions using a series of heuristics to ensure the system is outputting semantic predictions that are consistent with the syntax predictions for the same token.

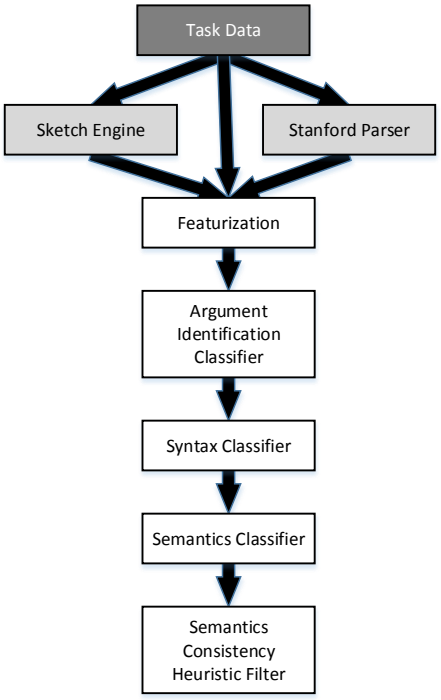

Figure 1: System Architecture Diagram. The input data is parsed by the Stanford Parser and the argument heads are expanded using the Sketch Engine thesaurus. This data is then featurized and passed through three successive classifiers: the Argument Identification Classifier identifies verb arguments, the Syntax Classifier assigns syntax labels to the arguments, and the Semantics Classifier assigns semantic labels to the arguments. Finally, the Semantics Consistency Heuristic Filter eliminates some systematic errors in the Semantics Classifier.

2.1 Featurization

Many of the features used in our system were inspired by the system produced by Toutanova et al. (2008), which used many features from prior work. This was a top-performing system and we incorporated each of the features that applied to the dependency parsing framework adopted in this task. We then augmented this feature set with a number of novel additional features. Many of these were adaptations of Semantic Role Labeling (SRL) features from the phrase-structure to dependency parsing paradigm (Gildea and Jurafsky 2002, Surdeanu et al. 2003, Pradhan et al. 2004). Others were added to generalize better to unseen verbs, which is critical for our task.

Some of our features depend on having a phrase-structure parse node corresponding to the candidate dependency parse node. Since dependency parse nodes each correspond to a token in the sentence, the tokens corresponding to the candidate node and its descendants in the dependency parse tree were identified. Then, in the phrase-structure parse tree, the lowest ancestor to all of these tokens was taken to be the phrase-structure parse node best corresponding to the candidate dependency parse node.

The baseline features included some inspired by Gildea and Jurafsky (2002):

- *Phrase Type*: the syntactic label of the corresponding node in the parse tree
- *Predicate Lemma*: lemma of the verb
- *Path*: the path in the parse tree between the candidate syntax node and the verb including the vertical direction and syntactic parse label of each node (e.g. "--up-->S--down-->NP")
- *Position*: whether the candidate is before or after the verb in the sentence
- *Voice*: whether the sentence is active or passive voice; due to sparse details in Gildea and Jurafsky this was based on tgrep search pattern heuristics found in Roland and Jurafsky (2001)
- *Head Word of Phrase*: the highest token in the dependency parse under the syntax parse tree node corresponding to the candidate token
- *Sub-Cat CFG*: the CFG rule corresponding to the parent of the verb, defined by the syntactic node labels of the parent and its children

Additional baseline features were obtained from Surdeanu et al. (2003) and Pradhan et al. (2004):

- *First/Last Word/POS*: For the syntactic parse node corresponding to the candidate node, this includes four separate features: the first word in the linear sentence order, its part of speech, the last word, and its part of speech
- *Left/Right Sister Phrase-Type*: The *Phrase Type* of each of the left and right sisters
- *Left/Right Sister Head Word/POS*: The word and POS of the head of the left and right sisters
- *Parent Phrase-Type*: The *Phrase Type* of the parent of the candidate parse node
- *Parent POS/Head-Word*: The word and part of speech of the parent of the parse node corresponding to the candidate node

- *Node-LCA Partial Path*: The *Path* between the candidate node and the lowest common ancestor between the candidate node and the verb
- *PP Parent Head Word*: The head word of the parent node in the syntax tree, if that parent is a prepositional phrase.
- *PP NP Head Word/POS*: If the syntax parse node representing the candidate node is a PP, the head word and POS of the rightmost NP directly under the PP.

Finally, baseline features that consisted entirely of pairs of already-mentioned features were also taken from Xue and Palmer (2004):

- *Predicate Lemma* & *Path*
- *Predicate Lemma* & *Head Word of Phrase*
- *Predicate Lemma* & *Phrase Type*
- *Voice* & *Position*
- *Predicate Lemma* & *PP Parent Head Word*

We added additional features adapted from the aforementioned features to generalize better given the sparse training data relative to other SRL tasks:

- *Head POS of Phrase*: the tagged POS of the *Head Word of Phrase*
- *Head Lemma of Phrase*: the lemma of the *Head Word of Phrase*
- *First/Last Lemma*: the lemma of the first and last word under the candidate parse node
- *Left/Right Sister Head Lemma*: the lemmas of the *Left/Right Sister Head Word*s
- *Parent Head Lemma*: the lemma of the *Parent Head Word*
- *PP Parent Head Lemma/POS*: the lemma and part of speech of the *PP Parent Head Word*
- *PP NP Head Lemma*: the lemma of the *PP NP Head Word*
- *Candidate CFG*: the context-free grammar rule rooted at the syntax parse node corresponding to the candidate node (one step down from *Sub-Cat CFG*)

Additional features were added to extend these features or to adapt them to dependency parsing:

- *Candidate DP CFG:* a CFG-like expansion of the dependency parse of the candidate node plus children, each represented by its POS (e.g. "NNS->PRP$" or "NNS->DT JJ NNS")
- *Sub-Cat DP CFG:* a similar CFG expansion of the dependency parse of the parent of the verb
- *First/Last DP Word/Lemma/POS* – of all of the descendants of the candidate node in the dependency parse, inclusive, the first/last word/lemma/POS from the linear sentence
- *Dependency Path*: the path in the dependency parse from the candidate node to the verb
- *Dependency Node-LCA Partial Path*: path in the dependency parse from the candidate node to its lowest common ancestor with the verb
- *Dependency Depth*: the depth in the dependency parse of the candidate node
- *Dependency Descendant Coverage*: of all of the tokens under the candidate syntax parse node, the percentage of those also under the candidate node in the dependency parse tree. This measures the candidate syntax and dependency parse node alignment.

Additionally, due to the importance of the *Predicate Lemma* feature in prior SRL work and the need to generalize entirely to unseen verbs for evaluation in this task, we used Sketch Engine (Kilgarriff 2014) word sketches for each verb. A word sketch is obtained for each unseen test verb and the most similar verb from the training data is used as the *Similar Predicate Lemma* feature.

We use a novel similarity function to identify similar verbs. A word sketch for each verb v_i identifies an ordered set of n grammatical relations r_{1i}, r_{2i}, r_{3i}, ..., r_{ni} that tend to co-occur with v_i. These are relations like "object", "subject", prepositional phrases head by "of", etc. The word sketch for each relation r_{ji} associated with v_i also includes a significance value $s_i(r_{ji})$. For a given verb v_i we calculate a directional similarity d_{ik} with verb v_k as:

$$d_{ik} = \sum_{j=1}^{n} (0.8)^{j-1} \left| s_i(r_{ji}) - s_k(r_{ji}) \right|$$

$\left| s_i(r_{ji}) - s_k(r_{ji}) \right|$ is defined as zero if the relation r_{ji} doesn't appear in both word sketches. The final similarity score u_{ik} between v_i and v_k is then:

$$u_{ik} = u_{ki} = \frac{d_{ik} + d_{ki}}{2}$$

2.2 Classifiers

We used a series of three classifiers with similar features, each trained using the mallet implementation of MaxEnt (McCallum 2002).

First, the AIC is a binary model predicting if a given candidate token is an argument of the verb. In the dependency parsing framework used for this

task, a single token in the dependency parse would represent a verbal argument. This was different from previous SRL tasks where a node in the parse tree was taken as the argument; this is more similar to identifying the headword of the phrase that's an argument rather than identifying the full phrase. Each token was treated as one example, with all of the features described in Section 2.1 calculated for each example. We filtered out features that did not appear at least five times in the training data, and trained with the default learning parameters.

Next, the multi-class Syntax Classifier uses the same features as the AIC plus a binary feature of AIC's score rounded to the nearest tenth, the AIC's predicted class, and these last two combined. The labels predicted were the syntactic label associated with the argument in the train data.

Finally, the multi-class Semantics Classifier predicts the semantic label of the argument using the features from the Syntax Classifier plus its output score rounded to the nearest tenth as a binary feature, its output label, and these last two combined.

2.3 Semantics Consistency Heuristic Filter

After running the classifiers, overgeneration by the semantic component was cleaned up using heuristics. Semantic predictions for tokens without a syntactic prediction were removed. For tokens with a syntactic but not semantic label prediction, if the token appeared in the train data with a semantic label the most common one was taken; if not, the most prominent distributional synonym (determined by the Sketch Engine thesaurus) found in the training data that has a semantic label was used.

3 Experiments

The system was evaluated using leave-one-out cross-validation on each verb in the train data. For the initial baseline configuration, only the features present in prior work were included, with a total of 31 feature classes. This configuration achieved an f-score of 0.238. The system was then run with our new features added, which outperformed the baseline by a relative 4% with an f-score of 0.248. In these cross-validation experiments, for each training example we used its *Similar Predicate Lemma* in place of its *Predicate Lemma* feature. This was a pessimistic assumption that we did not apply to the final system submitted for evaluation.[1] We suspect this explains why the final f-score on the test data was twice as good as that of the cross-validation experiments. The argument identification module performed well on its own with an f-score of 0.627, which is an upper bound on our overall system performance.

We used a hill climbing heuristic search for the best possible subset of the available features. This was a time-consuming process that involved running cross-validation for each feature class being evaluated with our three-stage classifier resulting in 63 classifiers being trained per iteration. All the feature removals or additions that improved performance were greedily accepted, yielding 22% feature churn. The best individual feature changes predicted 0.5% improvements to overall performance, but together they produced only a 0.9% improvement.

We repeated this a second time but only made the five most valuable changes, yielding a 0.8% point improvement. We did not have time to continue this greedy search, leaving further performance gains from searching for the best collection of features unrealized. We ended our search with 39 feature classes included, with only 21 of these from the original set. Through the course of these experiments, 10 of the original feature classes were removed while 18 new feature classes were added in our best model.

A final series of experiments were used to heuristically improve the semantic component which was significantly overgenerating. This yielded the Semantics Consistency Heuristics Filter which results in a 5% improvement to the overall system performance.

The final results on the test data are shown in Table 1. The baseline system still outperformed all teams including ours. The baseline was a heuristic system that used two dependency parsers to be more robust to parsing errors. It mapped dependency parse relations to syntax output directly, with logic to handle conjunctions, passives, and other phenomena. Semantic labels were a mixture of hard-coded

[1] *Predicate Lemma* is a critical feature in prior SRL work. In the test data, which only included unseen verbs, we used Sketch Engine data to identify the verb in the train data most similar to the verb in the test sentence, the *Similar Predicate Lemma* feature. In an attempt to mirror the features and avoid the possibility of cheating during our experiments, we repeated the same process during the cross-validation experiments, treating the other most similar verb in the training data as the *Similar Predicate Lemma*.

values for particular syntactic predictions and the most common value in the train data for the corresponding word or syntactic label.

Team	F-score
Baseline	0.624
FANTASY	0.589
BLCUNLP	0.530
CMILLS (our system)	**0.516**

Table 1: Performance on Test Data. Systems were evaluated on predicting the syntactic and semantic labels for the arguments of seven test verbs not present in the train data. Each system was evaluated by independently measuring the f-scores of its syntactic and semantic label predictions on each verb, averaged together by verb and then across verbs to arrive at the final f-score.

4 Conclusion

The experiments suggest that more iterations of the search for the best possible collection of features could yield significant additional improvements in system performance. However, we ran out of time before being able to complete more iterations of the search. While we trailed the second-place system by only 1.4% in overall f-score, the first-place system was ahead by 7.3% indicating significant improvements are still possible.

Additionally, the heuristic baseline outperformed all systems including ours, indicating that important patterns and intuitions were not encoded into features effectively. Given the sparsity of training data, it is possible that having more data could have also helped our approach based on pipelined classifiers.

In the future, we will evaluate using a single dev set instead of using cross-validation to reduce the computational cost of experiments. We were concerned about the sparse training data, but given the missed opportunity to further optimize the feature sets used by our models due to computational resource constraints, a single dev set could have been a much better approach. We would also like to use features from the semantic ontology rather than treating the semantic labels as unrelated tokens.

With our precision and recall within 2% of one another and relatively low, it would be challenging to reliably generate real-world lexical entries using this system, even with a delimited scope. However, approaches like this could be valuable at giving lexicographers a starting point to verify or modify, rather than starting from scratch.

This was a valuable learning experience, and while our efforts improved performance over our own baseline by nearly 12%, there is still plenty of room to improve and we have a clear path to do so by incorporating more features and improving experimental design.

Acknowledgments

Thank you to the anonymous reviewers, Ismail El Maarouf, and Daniel Cer for their helpful comments. Any mistakes that remain are our own.

References

Baisa, Vit, et al. (2015). SemEval-2015 Task 15: A CPA dictionary-entry-building task. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Co, USA, Association for Computational Linguistics.

Gildea, Daniel; Jurafsky, Daniel. (2002). "Automatic Labeling of Semantic Roles." *Computational Linguistics* 28(3): 245-288.

Kilgarriff, Adam, et al. The Sketch Engine: ten years on. In *Lexicography* (2014): 1-30.

Manning, Christopher; Surdeanu, Mihai; Bauer, John; Finkel, Jenny; Bethard, Steven; and McClosky, David. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60.

McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.

Pradhan, Sameer, et al. (2004). Shallow Semantic Parsing using Support Vector Machines. HLT-NAACL.

Roland, Douglas and Jurafsky, Daniel (2002). "Verb sense and verb subcategorization probabilities." The lexical basis of sentence processing: Formal, computational, and experimental issues 4: 325-45.

Surdeanu, Mihai, et al. (2003). Using predicate-argument structures for information extraction. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics-Volume 1*, Association for Computational Linguistics.

Toutanova, Kristina, et al. (2008). "A Global Joint Model for Semantic Role Labeling." *Computational Linguistics* 34(2): 161-191.

Xue, Nianwen and Palmer, Martha (2004). Calibrating Features for Semantic Role Labeling. *EMNLP*.