Valency-Augmented Dependency Parsing

Tianze Shi  
Cornell University  
tianze@cs.cornell.edu  

Lillian Lee  
Cornell University  
lle@cs.cornell.edu


Abstract

We present a complete, automated, and efficient approach for utilizing valency analysis in making dependency parsing decisions. It includes extraction of valency patterns, a probabilistic model for tagging these patterns, and a joint decoding process that explicitly considers the number and types of each token’s syntactic dependents. On 53 treebanks representing 41 languages in the Universal Dependencies data, we find that incorporating valency information yields higher precision and F1 scores on the core arguments (subjects and complements) and functional relations (e.g., auxiliaries) that we employ for valency analysis. Precision on core arguments improves from 80.87 to 85.43. We further show that our approach can be applied to an ostensibly different formalism and dataset, Tree Adjoining Grammar as extracted from the Penn Treebank; there, we outperform the previous state-of-the-art labeled attachment score by 0.7. Finally, we explore the potential of extending valency patterns beyond their traditional domain by confirming their helpfulness in improving PP attachment decisions.¹

1 Introduction

Many dependency parsers treat attachment decisions and syntactic relation labeling as two independent tasks, despite the fact that relation labels carry important subcategorization information. For example, the number and types of the syntactic arguments that a predicate may take is rather restricted for natural languages — it is not common for an English verb to have more than one syntactic subject or more than two objects.

In this work, we present a parsing approach that explicitly models subcategorization of (some) syntactic dependents as valency patterns (see Fig. 1 for examples), and operationalize this notion as extracted supertags. An important distinction from prior work is that our definition of valency-pattern supertags is relativized to a user-specified subset of all possible syntactic relations (see §3). We train supertaggers that assign probabilities of potential valency patterns to each token, and leverages these probabilities during decoding to guide our parsers so that they favor more linguistically plausible output structures.

We mainly focus on two subsets of relations in our analysis, those involving core arguments and those that represent functional relations, and perform experiments over a collection of 53 treebanks in 41 languages from the Universal Dependencies dataset (UD; Nivre et al., 2017). Our valency-aware parsers improve upon strong baseline systems in terms of output linguistic validity, measured as the accuracy of the assigned valency patterns. They also have higher precision and F1 scores on the subsets of relations under analysis, suggesting a potentially controlled way to balance precision-recall trade-offs.

We further show that our approach is not limited to a particular treebank annotation style. We apply our method to parsing another grammar formalism, Tree Adjoining Grammar, where dependency and valency also play an important role in both theory and parser evaluation. Our parser reaches a new state-of-the-art LAS score of 92.59, with more than 0.6 core-argument F1-score improve-

¹Our implementation is available at https://github.com/tzshi/valency-parser-emnlp18
ment over our strong baseline parser.

Finally, we demonstrate the applicability of our valency analysis approach to other syntactic phenomena less associated with valency in its traditional linguistic sense. In a case study of PP attachment, we analyze the patterns of two syntactic relations commonly used in PP attachment, and include them in the joint decoding process. Precision of the parsers improves by an absolute 3.30% on these two relation types.

2 Syntactic Dependencies and Valencies

According to Nivre (2005), the modern dependency grammar can be traced back to Tesnière (1959), with its roots reaching back several centuries before the Common Era. The theory is centered on the notion of dependency, an asymmetrical relation between words of a sentence. Tesnière distinguishes three node types when analyzing simple predicates: verb equivalents that describe actions and events, noun equivalents as the arguments of the events, and adverb equivalents for detailing the (temporal, spatial, etc.) circumstances. There are two types of relations: (1) verbs dominate nouns and adverbs through a dependency relation; (2) verbs and nouns are linked through a valency relation. Tesnière compares a verb to an atom: a verb can attract a certain number of bonds it can engage in (Ágel and Fischer, 2015). In many descriptive lexicographic works (Hellig and Schenkel, 1959; Herbst et al., 2004), valency is not limited to verbs, but also includes nouns and adjectives. For more on the linguistic theory, see Ágel et al. (2003, 2006).

Strictly following the original notion of valency requires distinguishing between arguments and adjuncts, as well as obligatory and optional dependents. However, there is a lack of consensus as to how these categorizations may be distinguished (Tutunjian and Boland, 2008), and thus we adopt a more practical definition in this paper.

3 Computational Representation

Formally, we fix a set of syntactic relations \( \mathcal{R} \), and define the valency pattern of a token \( w_i \) with respect to \( \mathcal{R} \) as the linearly-ordered\(^2\) sequence \( a_{-j} \cdots a_{-1} \circ a_1 \cdots a_k \): the \( \circ \) symbol denotes the center word \( w_i \), and each \( a_l \) asserts the existence of a word \( w \) dominated by \( w_i \) via relation \( a_l \in \mathcal{R} \), \( w_i \xrightarrow{a_l} w \). For \( a_l \) and \( a_m \), when \( l < m \), the syntactic dependent for \( a_l \) linearly precedes the syntactic dependent for \( a_m \). As an example, consider the UD-annotated sentence in Fig. 1. The token says has a core-dependency\(^3\) valency pattern \( \text{nsubj} \circ \text{ccomp} \), and like has the pattern \( \text{nsubj} \circ \text{xcomp} \). If we consider only functional relations, both like and swim have the pattern \( \text{mark} \circ \).\(^4\) We sometimes employ the abbreviated notation \( \alpha^L \circ \alpha^R \), where \( \alpha \) indicates a sequence and the letters \( L \) and \( R \) distinguish left dependencies from right dependencies.

We make our definition of valency patterns dependent on choice of \( \mathcal{R} \) not only because some dependency relations are more often obligatory and closer to the original theoretical definition of valency, but also because the utility of different types of syntactic relations can depend on the downstream task. For example, purely functional dependency labels are semantically vacuous, so they are often omitted in the semantic representations extracted from dependency trees for question answering (Reddy et al., 2016, 2017). There are also recent proposals for parser evaluation that downplay the importance of functional syntactic relations (Nivre and Fang, 2017).

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
Dataset & Subset & Syntactic Relations \\
\hline
UD & Core & nsubj, obj, iobj, csubj, ccomp, xcomp \\
\hline
UD & Func. & aux, cop, mark, det, clf, case \\
\hline
PP (§8) & & nmod, obl \\
\hline
TAG & Core & 0 (subject), 1 (object), 2 (indirect object) \\
\hline
Co-head & & CO \\
\hline
\end{tabular}
\caption{Sets of syntactic relations we used for valency analysis. UD subsets come from the official categorization in the annotation guidelines.}
\end{table}

\( ^2 \)Our approach, whose full description is in §5, can be adapted to cases where linear ordering is de-emphasized. The algorithm merely requires a distinction between left and right dependents. We choose to encode linearity since it appears that most languages empirically exhibit word order preferences even if they allow for relatively free word order.

\( ^3 \)UD core and functional relations are listed in Table 1.

\( ^4 \)The (possibly counterintuitive) direction for that and to is a result of UD’s choice of a content-word-oriented design.
4 Pilot Study: Sanity Checks

We consider two questions that need to be addressed at the outset:⁵

1. How well do the extracted patterns generalize to unseen data?

2. Do state-of-the-art parsers already capture the notion of valency implicitly, though they are not explicitly optimized for it?

The first question checks the feasibility of learning valency patterns from a limited amount of data; the second probes the potential for any valency-informed parsing approach to improve over current state-of-the-art systems.

To answer these questions, we use the UD 2.0 dataset for the CoNLL 2017 shared task (Zeman et al., 2017) and the system outputs⁶ of the top five performing submissions (Dozat et al., 2017; Shi et al., 2017b; Björkelund et al., 2017; Che et al., 2017; Lim and Poibeau, 2017). Selection of treebanks is the same as in §6. We extract valency patterns relative to the set of 6 UD core arguments given in Table 1 because they are close to the original notion of valency and we hypothesize that these patterns should exhibit few variations. This is indeed the case: the average number of valency patterns we extract is 110.4 per training treebank, with Turkish (tr) having the fewest at 34, and Galician (gl) having the most at 298 patterns. We observe that in general, languages with a higher degree of flexibility in word order tend to generate more patterns in the data, as our patterns encode linear word order information.

Next, we extract valency patterns from the test set and compare them against those from the training set. On average, out of the 55.4 patterns observed in the gold-standard test sets, only 5.5, or 9.98%, are new and unseen with respect to training. In comparison, 36.2% of the word types appearing in the test sets are not seen during training. This suggests that the valency pattern space is relatively restricted, and the patterns extracted from training sets do generalize well to test sets.

Finally, we consider the average number of valency patterns extracted from the top-performing system outputs and the number of those not observed in training.⁷ All 5 systems are remarkably “hallucinatory” in inventing valency relations, introducing 16.8 to 35.5 new valency patterns, significantly larger than the actual number of unseen patterns. Below we show an error committed by the state-of-the-art Dozat et al. (2017) parser (upper half) as compared to the gold-standard annotation (lower half), and we highlight the core argument valency relations of the verb bothers in bold. The system incorrectly predicts how come to be a clausal subject.

Each such non-existent new pattern implies at least some (potentially small) parsing error that can contribute to the degradation of downstream task performance.

5 Valency-Aware Dependency Parsing

5.1 Overview

Our model is based on the following probability factorization for a given sentence \( x = w_1, \ldots, w_n \) and parse tree \( y \) for \( x \):

\[
P(y|x) = \frac{1}{Z_x} \prod_{i=1}^{n} P(v_i|w_i) P(h_i|w_i) P(r_i|w_i, h_i),
\]

where \( Z_x \) is the normalization factor, \( v_i \) is the valency pattern extracted for \( w_i \) from \( y \), \( h_i \) is the index of the syntactic governor of \( w_i \), and \( r_i \) is the syntactic relation label of the dependency relation between \( w_{h_i} \) and \( w_i \). We first assume that we have a feature extractor that associates each token in the sentence \( w_i \) with a contextualized feature vector \( w_i \), and explain how to calculate the factored probabilities (§5.2). Then we discuss decoding (§5.3) and training (§5.4). Our decoder can be viewed as a special-case implementation of head-automaton grammars (Alshawi, 1996; Eisner and Satta, 1999). Finally, we return to the issue of feature extraction (§5.5).

⁵We actually performed these sanity checks after implementation and experiments of our approach, because we missed this idea and because it requires access to test sets that we abstained from looking at during model development.


⁷The CoNLL 2017 shared task is an end-to-end parsing task, so the participating systems do not have access to gold-standard tokenization, which is a potential explanation for the presented analysis. On the other hand, the conclusion still holds even if we restrict to system outputs with perfect or nearly perfect segmentations.
5.2 Parameterization

We parameterize \( P(v_i|w_i) \) as a softmax distribution over all candidate valency patterns:

\[
P(v_i|w_i) \propto \exp(\text{score}^\text{VAL}_i(w_i)),
\]

where \( \text{score}^\text{VAL}_i \) is a multi-layer perceptron (MLP).

For each word \( w_i \), we generate a probability distribution over all potential syntactic heads in the sentence (Zhang et al., 2017). After we have selected the head of \( w_i \) to be \( w_h \), we decide on the syntactic relation label based on another probability distribution. We use two softmax functions:

\[
P(h_i|w_i) \propto \exp(\text{score}^\text{HEAD}_{hi}(w_h, w_i)),
\]

\[
P(r_i|w_i, h_i) \propto \exp(\text{score}^\text{LABEL}_{ri}(w_h, w_i)),
\]

where both \( \text{score}^\text{HEAD} \) and \( \text{score}^\text{LABEL} \) are parameterized by deep biaffine scoring functions (Dozat and Manning, 2017).

5.3 Decoding

For joint decoding, we adopt the Eisner’s (1996) algorithm annotated with valency patterns as the state information in Eisner and Satta (1999). The algorithm is depicted in Fig. 2. For each complete and incomplete span, visualized as triangles and trapezoids respectively, we annotate the head with its valency pattern. We adopt Earley’s (1970) notation of \( \bullet \) to outward-delimit the portion of a valency pattern, starting from the center word \( \circ \), that has already been collected within the span. \textsc{Init} generates a minimal complete span with hypothesized valency pattern; the \( \bullet \) is put adjacent to \( \circ \).

\textsc{Comb} matches an incomplete span to a complete span with compatible valency pattern, yielding a complete analysis on the relevant side of \( \bullet \). \textsc{Link} either advances the \( \bullet \) by attaching a syntactic dependent with the corresponding relation label, or attaches a dependent with a relation label irrelevant to the current valency analysis. This algorithm can be easily extended to cases where we analyze multiple subsets of valency relations simultaneously: we just need to annotate each head with multiple layers of valency patterns, one for each subset.8

The time complexity of a naive dynamic programming implementation is \( O(|V|^2 |\alpha| n^4) \), where \( |V| \) is the number of valency patterns and \( |\alpha| \) is the maximum length of a valency pattern. In practice, \( |V| \) is usually larger than \( n \), making the algorithm prohibitively slow. We thus turn to \( A^* \) parsing for a more practical solution.

\textbf{A* parsing} We take inspiration from \( A^* \) CCG parsing (Lewis and Steedman, 2014; Lewis et al., 2016; Yoshikawa et al., 2017). The idea (see Alg. 1) is to estimate the best compatible full parse for every chart item (in our case, complete and incomplete spans), and expand the chart based on the estimated priority scores. Our factorization of probability scores allows the following admissible heuristic: for each span, we can optimistically estimate its best full parse score by assigning to

\[8\text{To allow our model to account for unseen patterns in new data, we create a special wildcard valency pattern that allows dependents with arbitrary relations in the decoding process, and during training, treat valency patterns occurring fewer than 5 times as examples of the wildcard pattern.} \]
Algorithm 1 Agenda-based best-first parsing algorithm, adapted from Lewis et al. (2016), Alg. 1.

**Helper Functions:**\( \text{INIT}(s) \) returns the set of spans generated by \( \text{INIT} \). \( \text{C.RULES}(p) \) returns the set of spans that can be derived by combining \( p \) with existing entries in \( C \) through \( \text{COMB} \) or \( \text{LINK} \).

1: \( \text{procedure} \ \text{PARSE}(s) \)
2:   \( \langle / \text{Empty priority queue} \ A \rangle \)
3:   \( A \leftarrow \emptyset \)
4:   \( \langle / \text{Initialize } A \text{ with minimal complete spans} \rangle \)
5:   \( \text{for } p \in \text{INIT}(s) \text{ do} \)
6:      \( A.\text{INSERT}(p); \)
7:   \( \langle / \text{Empty chart} \ C \rangle \)
8:   \( C \leftarrow \emptyset \)
9:   \( \text{while } A \neq \emptyset \text{ do} \)
10:      \( p \leftarrow A.\text{POPMAX}(); \)
11:      \( \langle / \text{Found the global optimal solution} \rangle \)
12:     \( \text{if } p \text{ is a full parse then return } p \)
13:     \( \text{else if } p \notin C \text{ then} \)
14:        \( C.\text{ADD}(p) \)
15:      \( \langle / \text{Extend the chart} \rangle \)
16:     \( \text{for } p' \in C.\text{RULES}(p) \text{ do} \)
17:        \( A.\text{INSERT}(p') \)

every token outside the span the best possible valency pattern, best possible attachment and best relation label.

5.4 Training

We train all components jointly and optimize for the cross entropy between our model prediction and the gold standard, or, equivalently, the sum of the log-probabilities for the three distributions comprising our factorization from §5.1. This can be thought of as an instance of multi-task learning (MTL; Caruana, 1997), which has been shown to be useful in parsing (Kasai et al., 2018). To further reduce error propagation, instead of using part-of-speech tags as features, we train a tagger jointly with our main parser components (Zhang and Weiss, 2016).

5.5 Feature Extraction

We adopt bi-directional long short-term memory networks (bi-LSTMs; Hochreiter and Schmidhuber, 1997) as our feature extractors, since they have proven successful in a variety of syntactic parsing tasks (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016; Stern et al., 2017; Shi et al., 2017a). As inputs to the bi-LSTMs, we concatenate one pre-trained word embedding, one randomly-initialized word embedding, and the output of character-level LSTMs for capturing sub-token level information (Ballesteros et al., 2015). The bi-LSTM output vectors at each timestep are then assigned to each token as its contextualized representation \( w_i \).

6 Experiments

**Data and Evaluation** Our main experiments are based on UD version 2.0, which was prepared for the CoNLL 2017 shared task (Zeman et al., 2017). We used 53 of the treebanks9 across 41 languages that have train and development splits given for the shared task. In contrast to the shared-task setting, where word and sentence segmentation are to be performed by the system, we directly use the test-set gold segmentations in order to focus directly on parsing; but this does mean that the performance of our models cannot be directly compared to the officially-reported shared-task results. For evaluation, we report unlabeled and labeled attachment scores (UAS and LAS respectively). Further, we explicitly evaluate precision, recall and F1 scores (P/R/F) for the syntactic relations from Table 1, as well as valency pattern accuracies (VPA) involving those relations.

**Implementation Details** We use three-layer bi-LSTMs with 500 hidden units (250 in each direction) for feature extraction. The valency analyzer uses a one-hidden-layer MLP with ReLU activation function (Nair and Hinton, 2010), while the head selector and labeler use 512- and 128-dimensional biaffine scoring functions respectively. Our models are randomly initialized (Glorot and Bengio, 2010) and optimized with AMSgrad (Reddi et al., 2018) with initial learning rate 0.002. We apply dropout (Srivastava et al., 2014) to our MLPs and variational dropout (Gal and Ghahramani, 2016) to our LSTMs with a keep rate of 0.67 during training.

**Efficiency** Our A* parsers are generally reasonably efficient; for the rare (< 1%) cases where the A* search does not finish within 500,000 chart expansion steps, we back off to a model without valency analysis. When analyzing three or more relation subsets, the initialization steps become pro-

---

9We exclude the two large treebanks cs and ru_syntagrus due to experiment resource constraints. There are other Czech and Russian treebanks in our selected collection.
Table 2: Macro-averaged results on UD 2.0 across 53 treebanks. Detailed results in the Suppl. Material. VPA=valency pattern accuracy; MTL=multi-task learning; #=average number of predicted attachments per sentence. Best results for each metrics are highlighted in bold.

<table>
<thead>
<tr>
<th>Subset</th>
<th>Core MTL</th>
<th>Function MTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>92.24</td>
<td>93.46</td>
</tr>
<tr>
<td>Core MTL + Joint Decoding</td>
<td>96.02/81.98/81.96</td>
<td>96.68/85.43/85.53</td>
</tr>
<tr>
<td>Func. MTL</td>
<td>97.44</td>
<td>93.16</td>
</tr>
<tr>
<td>Core + Func. MTL + Joint Decoding</td>
<td>97.51</td>
<td>97.74 (92.57/92.42)</td>
</tr>
</tbody>
</table>

Table 3: Treebank-specific F1 scores on core argument relations, comparing the baseline models to our Core MTL + joint decoding models, sorted by the error reduction (ER, %). When comparing a model with performance $s_2$ against baseline score $s_1$, ER is defined as $(s_2 - s_1)/(1 - s_1)$. For languages with two or three treebanks, we include multiple entries differentiated by the subscripts MAX/MID/MIN, corresponding to the treebanks with the highest/median/lowest ER, respectively. A. Greek = Ancient Greek.
hibitively slow due to the large number of valency pattern combinations. Thus, we limit the number of combinations for each token to the highest-scoring 500.

**Results on UD** We present our main experimental results on UD in Table 2. The baseline system does not leverage any valency information (we only train the head selectors and labelers, and use the original Eisner decoder). We compare the baseline to settings where we train the parsers jointly with our proposed valency analyzers, distinguishing the effect of using this information only at training (multi-task learning; MTL) vs. both at training and decoding.

Including valency analysis into the training objective already provides a slight improvement in parsing performance, in line with the findings of Kasai et al. (2018). With our proposed joint decoding, there is a mild improvement to the overall UAS and LAS, and a higher boost to VPA. The output parse trees are now more precise in the analyzed valency relations: on core arguments, precision increases by as much as 4.56. As shown by Table 3, the performance gain of joint decoding varies across treebanks, ranging from an error reduction rate of over 30% (Dutch Lassy Small Treebank) on core argument relations to nearly 0% (Japanese). Overall, our approach exhibits a clearly positive impact on most of the treebanks in UD. We do not see performance correlating to language typology, although we do observe smaller error-reduction rates on treebanks with lower baseline performances, that is, on “harder” languages.

### 7 Parsing Tree Adjoining Grammar

Dependency and valency relations also play an important role in formalisms other than dependency grammar. In this section, we apply our proposed valency analysis to Tree Adjoining Grammar (TAG; Joshi and Schabes, 1997), because TAG derivation trees, representing the process of inserting obligatory arguments and adjoining modifiers, can be treated as a dependency representation (Rambow and Joshi, 1997). We follow prior art and use Chen’s (2001) automatic conversion of the Penn Treebank (Marcus et al., 1993) into TAG derivation trees. The dataset annotation has labels 0, 1 and 2, corresponding to subject, direct object, and indirect object; we treat these as our core argument subset in valency analysis. Additionally, we also analyze CO (co-head for phrasal verbs) as a separate singleton subset. We leave out adj (adjuncts) in defining our valency patterns. We strictly follow the experiment protocol of previous work (Bangalore et al., 2009; Chung et al., 2016; Friedman et al., 2017; Kasai et al., 2017, 2018), and report the results in Table 4. The findings are consistent with our main experiments: MTL helps parsing performance, and joint decoding further improves on core argument F1 scores, reaching a new state-of-the-art result of 92.59 LAS. The precision recall trade-off is pronounced for the CO relation subset.

### 8 Case Study on PP Attachment

Although valency information has traditionally been used to analyze complements or core arguments, in this section, we show the utility of our approach in analyzing other types of syntactic relations. We choose the long-standing problem of prepositional phrase (PP) attachment (Hindle and Rooth, 1993; Brill and Resnik, 1994; Collins and Brooks, 1995; de Kok et al., 2017), which is known to be a major source of parsing mistakes (Kummerfeld et al., 2012; Ng and Curran, 2015). In UD analysis, PPs usually have the labels obl or nmod with respect to their syntactic parents, whereas adpositions are attached via a case relation, which is included in the functional relation subset. Thus, we add another relation subset, obl and nmod, to our valency analysis.

Table 5 presents the results for different combinations of valency relation subsets. We find that PP-attachment decisions are generally harder to make, compared with core and functional relations. Including them during training distracts other parsing objectives (compare Core + PP with only analyzing Core in §6). However, they do permit improvements on precision for PP attachment by 3.30, especially with our proposed joint decoding. This demonstrates the usage of our algorithm outside the traditional notions of valency — it can be a general method for training parsers to focus on specific subsets of syntactic relations.


Table 4: Experimental results on parsing TAGs.

<table>
<thead>
<tr>
<th></th>
<th>UAS</th>
<th>LAS</th>
<th>Core P / R / F</th>
<th>Func. P / R / F</th>
<th>PP P / R / F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>87.59</td>
<td>83.64</td>
<td>80.87 / 81.31 / 81.08</td>
<td>91.99 / 92.43 / 92.20</td>
<td>77.29 / 77.99 / 77.62</td>
</tr>
<tr>
<td>PP MTL</td>
<td>87.67</td>
<td>83.70</td>
<td>80.61 / 81.23 / 80.91</td>
<td>92.03 / 92.50 / 92.26</td>
<td>78.30 / 78.38 / 78.32</td>
</tr>
<tr>
<td>+ Joint Decoding</td>
<td>87.68</td>
<td>83.69</td>
<td>79.93 / 81.50 / 80.69</td>
<td>91.92 / 92.51 / 92.21</td>
<td>80.59 / 77.68 / 79.04</td>
</tr>
<tr>
<td>Core + PP MTL</td>
<td>87.70</td>
<td>83.77</td>
<td>81.62 / 81.81 / 81.71</td>
<td>91.93 / 92.52 / 92.22</td>
<td>77.93 / 78.25 / 78.08</td>
</tr>
<tr>
<td>+ Joint Decoding</td>
<td>87.80</td>
<td>83.91</td>
<td>84.18 / 81.97 / 83.05</td>
<td>91.68 / 92.65 / 92.16</td>
<td>79.71 / 78.03 / 78.83</td>
</tr>
<tr>
<td>Core + Func. + PP MTL</td>
<td>87.67</td>
<td>83.75</td>
<td>81.35 / 81.68 / 81.50</td>
<td>92.18 / 92.61 / 92.39</td>
<td>77.99 / 78.22 / 78.08</td>
</tr>
<tr>
<td>+ Joint Decoding</td>
<td>87.81</td>
<td>83.94</td>
<td>83.88 / 81.97 / 82.90</td>
<td>92.78 / 92.63 / 92.70</td>
<td>79.54 / 78.11 / 78.78</td>
</tr>
</tbody>
</table>

Table 5: Experimental results involving analyzing PPs as valency patterns.

9 Further Related Work

Supertagging Supertagging (Bangalore and Joshi, 2010) has been proposed for and used in parsing TAG (Bangalore and Joshi, 1999; Nasr and Rambow, 2004), CCG (Curran and Clark, 2003; Curran et al., 2006), and HPSG (Ninomiya et al., 2006; Blunsom and Baldwin, 2006). Within dependency parsing, supertags have also been explored in the literature, but prior work mostly treats them as additional features. Ambati et al. (2013, 2014) use CCG supertags to improve dependency parsing results, while Ouchi et al. (2014, 2016) leverage dependency-based supertags as features. Faleńska et al. (2015) compare supertagging to parser stacking, where they extract supertags from base parsers to provide additional features for stacked parsers, instead of having a supertagger as a separate component.

Constrained Dependency Grammar Another line of research (Wang and Harper, 2004; Foth et al., 2006; Foth and Menzel, 2006; Bharati et al., 2002, 2009; Husain et al., 2011) utilizes supertags in dependency parsing within the framework of constraint dependency grammar (CDG; Maruyama, 1990; Heinecke et al., 1998). Constraints in CDG may be expressed in very general terms (and are usually hand-crafted for specific languages), so prior work in CDG involves a constraint solver that iteratively or greedily updates hypotheses without optimality guarantees. In contrast, our work focuses on a special form of constraints — the valency patterns of syntactic dependents within a subset of relations — and we provide an efficient A*-based exact decoding algorithm.

Valency in Parsing To the best of our knowledge, there have been few attempts to utilize lexical valency information or to improve specifically on core arguments in syntactic parsing apart from CDG. Övrelid and Nivre (2007) target parsing core relations in Swedish with specifically-designed features such as animacy and definiteness that are useful in argument realization. Jakubiček and Kovář (2013) leverage external lexicons of verb valency frames for reranking. Mirroshandel et al. (2012, 2013) and Mirroshandel and Nasr (2016) extract selectional constraints and subcategorization frames from large unannotated corpora, and enforce them through forest reranking. Our approach does not rely on external resources or lexicons, but directly extracts valency patterns from labeled dependency parse trees. Earlier works in this spirit include Collins (1997).

Semantic Dependency Parsing and Semantic Role Labeling The notion of valency is also used to describe predicate-argument structures that are adopted in semantic dependency parsing and semantic role labeling (Surdeanu et al., ...
While semantic frames clearly have patterns, previous work (Punyakanok et al., 2008; Flanigan et al., 2014; Töckström et al., 2015; Peng et al., 2017; He et al., 2017) incorporates several types of constraints, including uniqueness and determinism constraints that require that certain labels appear as arguments for a particular predicate only once. They perform inference through integer linear programming, which is usually solved approximately, and cannot easily encode linear ordering constraints for the arguments.

**A* parsing** Best-first search uses a heuristic to expand the parsing chart instead of doing so exhaustively. It was first applied to PCFGs (Ratnaparkhi, 1997; Caraballo and Charniak, 1998; Sagae and Lavie, 2006), and then to dependency parsing (Sagae and Tsujii, 2007; Zhao et al., 2013; Vaswani and Sagae, 2016). Our probability factorization permits a simple yet effective A* heuristic. A* parsing was introduced for parsing PCFGs (Klein and Manning, 2003; Pauls and Klein, 2009), and has been widely used for grammar formalisms and parsers with large search spaces, for example CCG (Auli and Lopez, 2011) and TAG (Wasczczuk et al., 2016, 2017). Our decoder is similar to the supertag and dependency factored A* CCG parser (Yoshikawa et al., 2017), which in turn builds upon the work of Lewis and Steedman (2014) and Lewis et al. (2016). Our model additionally adds syntactic relations into the probability factorizations.

**10 Conclusions**

We have presented a probability factorization and decoding process that integrates valency patterns into the parsing process. The joint decoder favors syntactic analyses with higher valency-pattern supertagging probabilities. Experiments on a large set of languages from UD show that our parsers are more precise in the subset of syntactic relations chosen for valency analysis, in addition to enjoying the benefits gained from jointly training the parsers and supertaggers in a multi-task learning setting.

Our method is not limited to a particular type of treebank annotation or a fixed subset of relations. We draw similar conclusions when we parse TAG derivation trees. Most interestingly, in a case study on PP attachment, we confirm the utility of our parsers in handling syntactic relations beyond the traditional domain of valency.

A key insight of this paper that departs from prior work on automatic extraction of supertags from dependency annotations is that our definition of valency patterns is relativized to a subset of syntactic relations. This definition is closer to the linguistic notion of valency and alleviates the data sparsity problems in that the number of extracted valency patterns is small. At the same time, the patterns generalize well, and empirically, they are effective in our proposed joint decoding process.

Our findings point to a number of directions for future work. First, the choice of subsets of syntactic relations for valency analysis impacts the parsing performance in those categories. This may suggest a controllable way to address precision-recall trade-offs targeting specific relation types. Second, we experimented with a few obvious subsets of relations; characterizing what subsets can be most improved with valency augmentation is an open question. Finally, our decoder builds upon projective dependency-tree decoding algorithms. In the future, we will explore the possibility of removing the projective constraint and the tree requirement, extending the applicability of valency patterns to other tasks such as semantic role labeling.

**Acknowledgments**

We thank the three anonymous reviewers for their insightful comments, Jungo Kasai for assistance in setting up the TAG parsing experiments, and Xilun Chen, Jason Eisner, Jungo Kasai and Ana Smith for discussion and comments. We also thank CoNLL’17 shared task organizers and participants for publicizing system outputs. TS and LL were supported in part by a Google Focused Research Grant to Cornell University. LL was also supported in part by NSF grant SES-1741441. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other sponsors.

**References**


Vilmos Ágel and Klaus Fischer. 2015. Dependency grammar and valency theory. In Bernd Heine and


Adwait Ratnaparkhi. 1997. A linear observed time
Owen Rambow and Aravind Joshi. 1997. A formal
Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008.
Lilja Øvrelid and Joakim Nivre. 2007. When word


