DPH Parser: A Bottom-Up Grammar-Driven Parser for Joint Constituency and Dependency Analysis

Anonymous ACL submission

Abstract

This paper presents Dependency-Phrase Hierarchy Parser (DPH Parser), a grammar-driven bottom-up unsupervized parsing framework inspired by Generalized Phrase Structure Grammar (GPSG) and Head-driven Phrase Structure Grammar (HPSG). The parser incrementally constructs constituency structures using a compact inventory of feature-based syntactic rules while deriving dependency relations through explicit head annotations. The system combines probabilistic POS tagging, recursive phrase projection, and weighted parse hypotheses to process realistic and partially noisy text input. Unlike purely neural and data-driven parsers, the resulting syntactic derivations remain explicitly interpretable.

We evaluated parser performance on English corpora from the Universal Dependencies (UD) project using Unlabeled Attachment Score (UAS) as the main parsing metric, comparing the outcomes against Stanza and spaCy parsers. For a small inventory of syntactic rules, DPH parser achieved UAS values of 53.32% & 52.58% (UD Devset/Testset respectively). For the same data, Stanza achieved 89.12% & 88.67% while spaCy achieved 56.91% and 58.59%. Although the current system does not yet approach the accuracy of modern neural parsers, the results demonstrate the feasibility of applying transparent rule-based bottom-up parsing to realistic treebank data while jointly producing constituency and dependency structures.

1 Introduction

Natural language parsing remains a fundamental problem in computational linguistics despite the recent success of large neural language models (LLMs). Modern transformer-based systems often achieve strong downstream performance without explicit syntactic representations, and contemporary dependency parsers routinely achieve high attachment accuracy on benchmark corpora. Nevertheless, these systems typically encode syntactic structure implicitly within distributed representations, making their structural decisions difficult to interpret, constrain, or modify explicitly. This is particularly important for sentences with ambiguity, with multiple possible underlying structures. For example, in a prompt such as: "generate a picture of men and women wearing hats", do both men and women wear hats, or only women? Therefore, even for more advanced LLM and neural-based NLP applications, it is still important to recognize ambiguity, while maintaining proper syntactic derivation of the underlying structure.

The goal of the current work is not to replace modern neural parsers, but rather to investigate whether a compact inventory of linguistically motivated syntactic rules can support a practical parsing framework that incrementally constructs interpretable phrase structures while simultaneously preserving dependency information.

The proposed grammar formalism is primarily inspired by Generalized Phrase Structure Grammar (GPSG) (Gazdar, 1985) and Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994). In contrast to transformational approaches of generative syntax (starting with Chomsky 1957), these frameworks attempt to represent long-distance dependencies and syntactic constraints through feature propagation mechanisms rather than explicit movement operations. In particular, GPSG introduced slash features for representing constituents with missing arguments (e.g. the object of the verb "eat" in "What did you eat X?"), while HPSG emphasized typed feature structures and explicit head propagation. These ideas provide useful mechanisms for constructing phrase structures while simultaneously preserving dependency relations between words.

This formalism addresses a common issue for modern parsing systems, which often distinguish

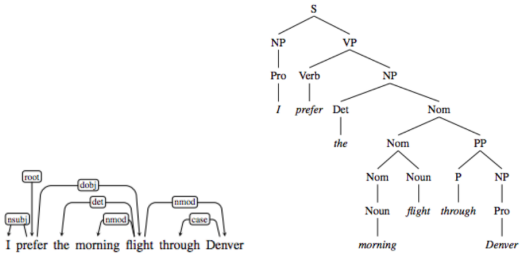

Figure 1: Dependency and Constituency Structures

between constituency parsing (producing hierarchical phrase structure), and dependency parsing (producing binary head-dependent relations between words). Both representations are interrelated, and each can be converted into the other, but each has a certain focus and provides certain information about the structure (Xia and Palmer, 2001). While dependency parsing dominates many contemporary NLP pipelines because of its efficiency and compatibility with downstream applications (Nivre, 2006), constituency structures continue to provide useful information regarding hierarchical organization, scope, and prosodic grouping (Jurafsky and Martin, 2026).

The current work adopts a constituency-oriented parsing strategy while explicitly annotating head information within grammar rules, allowing dependency relations to be derived directly from phrase structures. This design allows both representations to be preserved within a single parsing process. The main contributions of this work includes an inventory of rules and lexical categories and features as well as a bottom-up parsing algorithm based on this inventory, in addition to a hybrid dependency-constituency format that the parser produces, showing full derivation for each parse, and different parse hypotheses.

2 Related Work

Modern syntactic parsing research is dominated by supervised statistical and neural approaches, particularly transition-based and graph-based dependency parsers, as well as neural constituency parsers. These systems typically achieve high parsing accuracy on benchmark corpora through large-scale supervised learning, typically relying on learned representations rather than manually specified grammatical constraints. Despite their empirical success, there remains continued interest in grammar-constrained and linguistically interpretable parsing systems, particularly in areas involving formal syntax, semantic compositionality, and explainable language processing.

Transition-based dependency parsers incrementally construct dependency trees through sequences of parser actions (Nivre, 2008), while graph-based parsers formulate parsing as a global optimization problem over candidate dependency arcs (McDonald et al., 2005). In parallel, constituency parsing has continued through neural chart parsers and span-based architectures, many of which retain bottom-up parsing principles inherited from earlier CFG and CKY-style parsing algorithms (Kasami 1966 and Younger 1967).

Grammar-based parsing approaches based on feature-rich phrase structure formalisms remain an important line of research in computational linguistics. In particular, Generalized Phrase Structure Grammar (GPSG) and Head-driven Phrase Structure Grammar (HPSG) demonstrated that many syntactic phenomena could be represented through explicit grammatical constraints and feature propagation mechanisms without relying exclusively on transformational operations, such as movement in generative grammar.

Early computational work on GPSG and HPSG focused primarily on parsing efficiency and constraint satisfaction. For example, Fisher (1989) introduced parsing methods for GPSG, while Torisawa et al. (2000) proposed CFG filtering techniques to reduce the computational complexity of HPSG parsing. Subsequent work explored broad-coverage grammar engineering, including corpus-oriented Japanese HPSG parsing (Yoshida, 2005) and hybrid constituent-dependency formulations of HPSG (Zhou and Zhao, 2019).

Recent work has increasingly combined symbolic grammatical formalisms with neural architectures. Nguyen et al. (2024) developed a neural parser based on simplified HPSG representations for Vietnamese, while Zamaraeva and Gómez-Rodríguez (2024) incorporated HPSG-inspired supertagging into neural constituency parsing pipelines. Similarly, Li et al. (2021) explored training HPSG parsers directly from constituency and dependency annotations. These approaches illustrate a broader trend toward neuro-symbolic parsing systems that integrate explicit grammatical structure with statistical learning methods.

A related line of work explored parsing within Minimalist Grammar and transformational frameworks. These studies investigated both formal parsing properties and probabilistic inference mecha-

nisms for movement-based grammars, including bottom-up parsing methods (Harkema, 2000), probabilistic Minimalist parsing (Mainguy 2010 , Portelance et al. 2017), and wide-coverage applications of Minimalist Grammar parsing (Torr et al., 2019). While these approaches differ theoretically from GPSG and HPSG, they similarly attempt to connect formal syntactic theory with computational parsing implementations.

In parallel, deep grammar engineering projects such as the DELPH-IN ecosystem and the English Resource Grammar [1] demonstrated the feasibility of broad-coverage HPSG-based parsing systems using platforms such as PET (Callmeier, 2000) and ACE (Crysmann and Packard, 2012). At the same time, neural constituency parsers, including span-based and chart-based approaches (Stern et al. 2017 and Kitaev and Klein 2018), showed that bottom-up phrase structure parsing remains computationally effective within modern neural NLP systems, although without relying on explicit symbolic grammar rules.

Building on these approaches, the current work explores building a bottom-up parsing approach, inspired by GPSG and HPSG grammars.

3 Methodology

3.1 Parsing Algorithm Overview

The goal of this research is to develop a parsing approach capable of producing syntactically interpretable sentence structures while addressing practical challenges such as ambiguity, noisy or incomplete input, syntactic gaps, and the joint representation of constituency and dependency relations. The proposed parser follows a bottom-up incremental approach inspired by GPSG and HPSG, where syntactic structures are constructed recursively from words and their probable categories using a compact inventory of feature-based grammatical rules.

The parser proceeds left-to-right, beginning with multiple POS-tag hypotheses and their probabilistic weights. Each hypothesis is converted into an initial phrase object and recursively projected upward through unary and binary syntactic rules. Binary projections search for compatible preceding phrases while enforcing category and feature constraints. Each rule explicitly identifies the head child of the phrase, allowing dependency relations to be derived directly from constituency projections and enabling a unified syntactic representation.

[1] https://delph-in.github.io/docs/erg/ErgTop/

Unlike transition-based and graph-based dependency parsers such as spaCy and Stanza, the current system builds syntactic structure through explicit grammatical constraints rather than parser actions or global arc optimization. The parser also incorporates weighted ambiguity resolution, limited tolerance for disfluencies and extra-textual elements, and fallback mechanisms for partially parseable input.

3.2 Syntactic Rules

One of the primary contributions of this work is the development of an extensible inventory of syntactic rules that encode grammatical structures in a transparent and maintainable format[2]. These rules are inspired by principles from GPSG and HPSG while remaining computationally practical.

The rules resemble Context-Free Grammar (CFG) productions but incorporate additional feature structures and head annotations. Each rule contains one parent category and either one or two child categories. In binary rules, exactly one child is designated as the head child, enabling direct recovery of dependency relations.

A rule may be represented schematically as:

PARENT[features] → CHILD-1^[features] CHILD-2[features]

Categories may contain positive or negative features that constrain grammatical combinations. These features allow the grammar to capture distinctions that would otherwise be difficult to represent using ordinary CFG rules alone.

For example, noun phrases in English can recursively combine in constructions such as "United Nations Development Program." A naive CFG rule such as:

NP → NP NP

would incorrectly license ungrammatical structures (e.g. * [NP the book the man] is good). To address this issue, the grammar introduces refined categories such as NP-U for noun phrases lacking determiners and NP-DT for noun phrases containing determiners. This permits more constrained constructions such as:

NP → NP-U NP-U and NP-DT → DT NP-U

Similarly, verb phrase structures are refined to avoid unrestricted recursion. Rather than allowing unlimited object attachment through a rule such as:

VP → VP NP

[2] Rule spreadsheets can be found at: https://github.com/chplu-dev/dph-parser/tree/main/rules

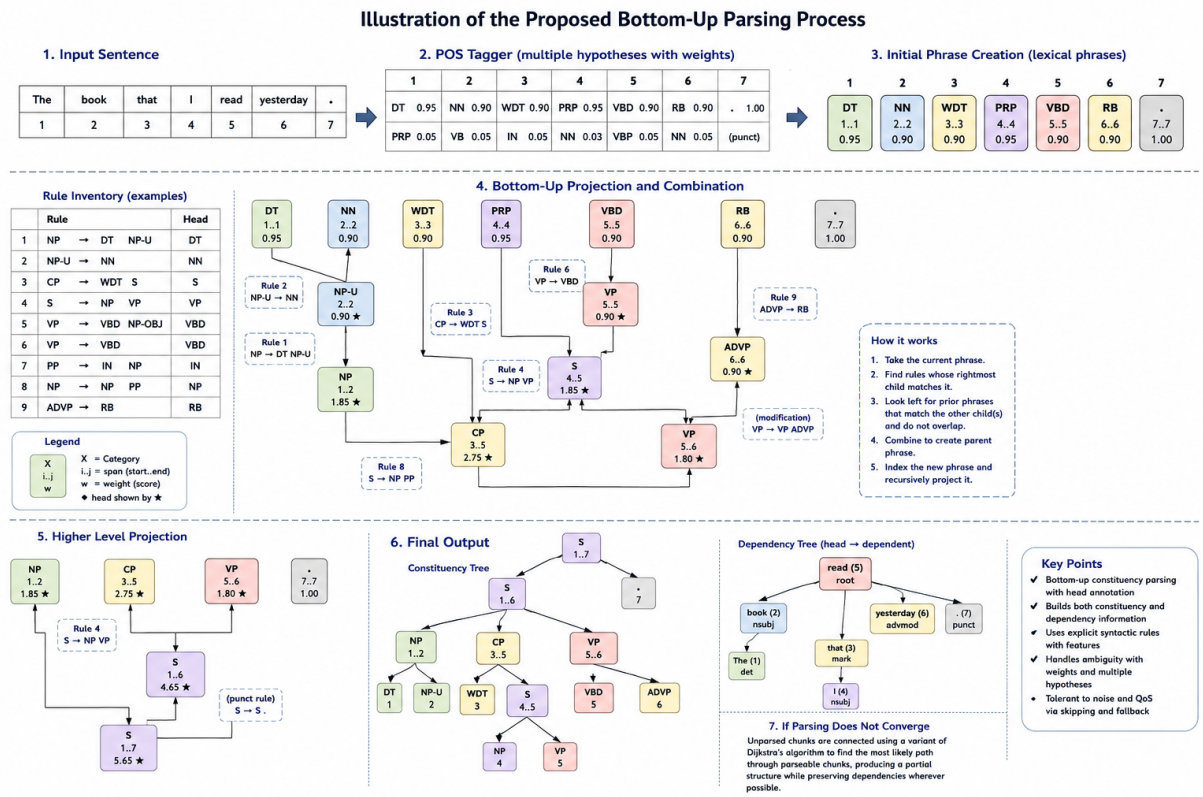

Figure 2: Parsing Algorithm Overview - Generated by ChatGPT for illustration purposes only

the grammar distinguishes between verb phrases containing one object and those containing two objects:

VP-O → V NP and VP-2O → VP-O NP

This constrains the grammar to structures more consistent with English syntax.

The grammar also encodes auxiliary verb sequences and feature agreement. For example, forms of the verb "to be" are projected into a BE category using feature constraints:

BE → AUX[+be] and BE → HAVE AUX[+been]

These rules allow valid auxiliary sequences such as "would have been" while disallowing invalid combinations (e.g. * have would been).

The inventory additionally incorporates limited GPSG-style slash features for representing syntactic gaps and extraction phenomena. Head information and feature percolation mechanisms are inspired by HPSG, allowing parent categories to inherit features from their head children.

The rule inventory consists of three major components: 1) Phrase structure rules defining non-terminal projections; 2) Terminal rules mapping POS tags into syntactic categories and features; and 3) Lexical tags defining categories and features for closed-class words

The grammar is intentionally extensible and maintainable through spreadsheet-style representations and version-controlled rule inventories. The purpose of the current work is not to exhaustively define all grammatical structures of English, but rather to demonstrate a flexible methodology for integrating linguistic theory into practical parsing systems.

3.3 POS Tagging

The parser depends on probabilistic POS tagging as the initial stage of the parsing pipeline. Although many POS taggers are available, the current system requires specific properties: 1) The tag set must align with the categories used in the syntactic rules; 2) The tagger must provide confidence scores or probabilities; and 3) The tagger should produce multiple tag hypotheses for each token.

Producing multiple hypotheses is particularly important because the most probable POS tag is not always the tag that yields the best global parse.

To satisfy these requirements, a simplified POS tagger based on Recurrent Neural Networks (RNNs) was developed. The input consists of sequences of tokens, while the output is a probability

distribution over both Universal POS (UPOS) and language-specific XPOS tags from the Universal Dependencies framework. XPOS tags are preferred over generic UPOS tags because they provide finer-grained distinctions necessary for syntactic rules. For example, different rules apply to gerunds, infinitives, participles, and finite verbs.

The features used by the POS tagger include leading and training characters within each token, capitalization patterns, and orthographic properties such as hyphenation. The model was trained using training data from Universal Dependencies treebanks. Parameters: LSTM network - hidden_size=64, n_layers=3, learning rate=0.0001, Adam Optimizer, Cross Entropy Loss, applying sigmoid at the output. This tagger learns from training data, therefore it is the only supervised part of the current study.

The output of the POS tagger forms the initial set of weighted terminal phrase objects used by the parser.

3.4 Lexical Tags

In addition to the output of POS tagger, and as part of the rule inventory, we introduce a custom list for lexical items and their categories and features. This list of lexical tags is mainly for function words, or words with specific grammatical role. Such tags are not typically captured by standard UPOS or XPOS tags. For example, the word "is" takes "AUX" and "VBZ" tags, although it has specific grammatical properties separating it from other AUX and Verb tags. Therefore, we assign a custom category "BE" to verb to-be words (is, was, were, am . . . etc). A similar situation applies for the verb to-have and to-do. Additional features are also introduced to identify how for example personal pronouns can be used as determiners before nouns (e.g. my book - the book). This custom lexical information augments the output of the POS tagger, while matching categories and features used in the syntactic rules.

3.5 Phrase Representation

The central data structure in the parser is the phrase object. Each phrase represents either a terminal lexical category or a non-terminal syntactic constituent.

Each phrase contains the following information:

[Phrase start position, Phrase end position, Syntactic category, Feature set, Unique phrase identifier, Child phrase identifiers, Head child information, Weight or confidence score, Dependency relation information]

These phrase objects form the basis for all subsequent parsing operations.

3.6 Parsing and Phrase Projection

The parsing process begins by converting each POS-tag hypothesis, as well as those produced from lexical tags, into an initial phrase object. Each phrase is then recursively projected using the syntactic rule inventory.

Projection refers to the process of identifying rules whose child categories match the current phrase. Unary rules involve projection from one category into another (e.g. NP $\rightarrow$ NN), while binary rules combine two adjacent phrases into a larger constituent. For binary rules, the parser first identifies rules whose final child matches the current phrase. It then searches previously created phrases to locate a compatible preceding phrase that satisfies the first child of the rule. For example, given the rule:

PP $\rightarrow$ IN NP

if the current phrase is NP, the parser scans preceding phrases to identify a compatible IN phrase. If found, the two phrases are combined into a PP phrase.

When phrases are combined, the parser creates a new parent phrase with updated information from the two child phrases, including the total weight of both, combined span, and which child is the head. The resulting phrase is indexed and recursively projected further.

3.7 Parser Indexing and Optimization

Efficient indexing is essential because parsing generates many overlapping phrase hypotheses. The parser therefore maintains several indexing structures for rapid lookup of phrases and rules. The primary indexing structure organizes phrases according to their spans and categories:

Dictionary[end][start][category] = [phrase1, phrase2, ...]

Additional indexes are maintained for quickly retrieving rules associated with particular categories and features. These indexing strategies substantially reduce the search space and improve parsing efficiency.

3.8 Noise Handling and Fallback Parsing

Natural language input frequently contains disfluencies, interruptions, punctuation artifacts, or incomplete constructions. To improve robustness,

the parser allows limited skipping between adjacent phrases during binary matching. This permits phrases to combine even when separated by small amounts of noise. Nevertheless, some sentences may remain partially unparsable because of grammatical irregularities or missing rules. In such cases, the parser applies a fallback mechanism inspired by Dijkstra's shortest-path algorithm. Rather than forcing a fully convergent parse, the system identifies the most likely sequence of parseable chunks and connects them into a partial structure.

3.9 Output Representation and Evaluation

The final parser output includes both constituency and dependency representations. The highest-scoring phrase spanning the full sentence, or the largest parseable span, is selected as the primary parse. Recursive traversal of phrase children produces:

- A constituency tree representation (converted to PTB bracketed format)
- Dependency relations between words (converted to CoNLL tabular format)

The parser is evaluated both qualitatively and quantitatively. Qualitative evaluation involves manual inspection of parse outputs and their consistency with predefined syntactic rules. Quantitative evaluation involves measuring parsing performance on multiple corpora and comparing the results against widely used systems such as spaCy and Stanza.

4 Experimental Setup

4.1 Metrics

In order to evaluate the performance of the current parsing approach, we use Unlabelled Attachment Score (UAS). This metric reveals the quality of parsing mainly in terms of showing which words depend on which. The measurement counts how many tokens the parser identified their heads correctly, divided by the total number of tokens in the dataset.

4.2 Data

This parser is tested on the English dev/test datasets from Universal Dependencies (UD) [3] (Nivre et al., 2020). It includes multiple corpora with different distributions for train/dev/test sets. It includes 8 files in the train set, 7 files in the dev set, and 13 files for the test sets. Each file includes sentences annotated according to the CoNLL format for dependency parsing. This parser is applied on the tokenized sentences in each CoNLL item from the treebank.

For benchmarking, we use spaCy dependency parser[4], transition based parser (Honnibal and Johnson, 2014) . We also use Stanza [5], a graph-based parser (Qi et al., 2020) . The results are based on the standard models for both spaCy and Stanza without any training or fine-tuning, by applying it to the tokenized sentences from the treebank. We use the UD evaluation tools [6] to measure the UAS of produced parses in CoNLL format against the Gold-Standard parses of the respective corpora.

4.3 Code

The code used for these experiments, including notebooks, POS models, python files, Excel sheets for rules and other lists, is included in a GitHub repository for easy access and reproducibility [7] . The readme file includes relevant instructions for using the code.

5 Results and Discussion

For quantitative evaluation, we compare the UAS metrics from DPH parser to those from the spaCy and Stanza parsers. Results are obtained on Development/Test sections of UD datasets (devset/testset). The performance figures reflect how many phrases and dependencies are identified correctly by the parser. The average UAS for each parser across devset/testet is weighted according to the number of tokens within each dataset.

As we can see from figure 3, DPH parser achieved UAS of 53.32% & 52.58% (UD Devset/Testset respectively). For the same data, Stanza achieved 89.12% & 88.67% while spaCy achieved 56.91% and 58.59%. The performance of spaCy is lower than expected, given how commonly used it is, also given that UAS values for UD datasets for the shared task in 2018 reached up to 80.51% [8]. However, it should be noted that we used the off-the-shelf smallest model, without training on the UD training data or any other fine tuning.

[3] https://universaldependencies.org

[4] https://spacy.io/api/dependencyparser

[5] https://stanfordnlp.github.io/stanza/

[6] https://universaldependencies.org/tools.html

[7] https://github.com/chplu-dev/dph-parser/

[8] https://universaldependencies.org/conll18/results-uas.html

DEV SET	n_sentences	avg sent length	n_tokens	Stanza - UAS	spaCy - UAS	DPH - UAS
ATIS	572	11.6	6644	75.50%	41.66%	55.89%
CHILDES	3860	6.3	24310	90.63%	66.66%	58.06%
ESLSPOK	232	9.3	2167	90.86%	61.10%	61.24%
EWT	2001	12.6	25144	90.81%	56.11%	51.42%
GUM	1575	17.9	28119	91.46%	53.32%	49.90%
LINES	1118	19.4	21637	86.38%	56.11%	53.06%
PARTUT	156	17.4	2716	89.43%	54.68%	53.20%
Weighted Average				**89.12%**	**56.91%**	**53.32%**
TEST SET						
ATIS	586	11.2	6580	76.90%	39.89%	54.35%
CHILDES	9591	6.0	57400	89.18%	67.57%	57.68%
CTETEX	276	33.6	9273	80.98%	52.50%	41.41%
ESLSPOK	232	9.8	2266	92.19%	64.52%	60.15%
EWT	2077	12.1	25091	90.77%	56.67%	50.83%
GENTLE	1334	13.3	17799	84.52%	52.14%	48.30%
GUM	1464	19.4	28397	91.08%	53.97%	49.53%
LINES	1121	17.8	19984	87.81%	57.01%	53.16%
LITTLE PRINCE	500	13.7	6852	91.35%	59.84%	52.92%
PARTUT	153	22.2	3403	88.39%	55.30%	54.22%
PRONOUNS	285	6.0	1705	95.72%	63.70%	56.60%
PUD	1000	21.2	21180	91.14%	57.15%	50.82%
Weighted Average				**88.67%**	**58.59%**	**52.58%**

Figure 3: UAS results from DPH parser against spaCy and Stanza

It is also noteworthy that spaCy performed much worse at ATIS dataset [9] (UAS devset: 41.66% - testet: 39.89%). This dataset warrants further analysis, since also Stanza had the lowest UAS values at this dataset (devset 75.50%, testset: 76.90%). On the other hand, DPH parser achieved average performance for this dataset (devset 55.89%, testset 54.35%).

Although the performance of DPH parser is much lower than Stanza, it is only a few points below spaCy performance. In addition, we can see that the DPH performance is mostly robust across multiple datasets, even surpassing spaCy in multiple datasets across the devset and testset, as we can see from the ATIS example.

At this stage, the parser is only programmed with a small subset of rules (148 rules) of the English language, just as a proof of concept. While these rules do not provide full coverage of English grammar, it shows proof of concept that programming even a small subset can be used for parsing various datasets. Further investigation is needed to measure the effect of wider coverage in improving parsing outcomes.

Qualitatively, we can notice that parser output reflects both constituency and dependency structures. The representation proposed includes the tokens of the input sentence, as well as their IDs, heads and tags, which captures some of the main information from the dependency structure. It also includes a hierarchical listing of the phrases, their levels and their spans, as we can see in figure 4.

An important point to examine is the applicability of this parsing approach to custom syntactic rules pertaining to certain linguistic structures and phenomena. For this case, we address, as an example, the slash features within GPSG. We can see in figure 4 that the parser was able to correctly handle and represent the structure of the phrase:

"the man my friend saw X"

The output shows the derivation of all the rules applied. Such rules include the slash rule VP-O/NP corresponding to a VP missing a direct object (the gap construction), and its projection into S/NP. Eventually, we see the application of the rule to fill the gap:

NP $\rightarrow$ NP^ S/NP

We can apply similar rules for Prepositional

[9] https://github.com/UniversalDependencies/UD_English-Atis

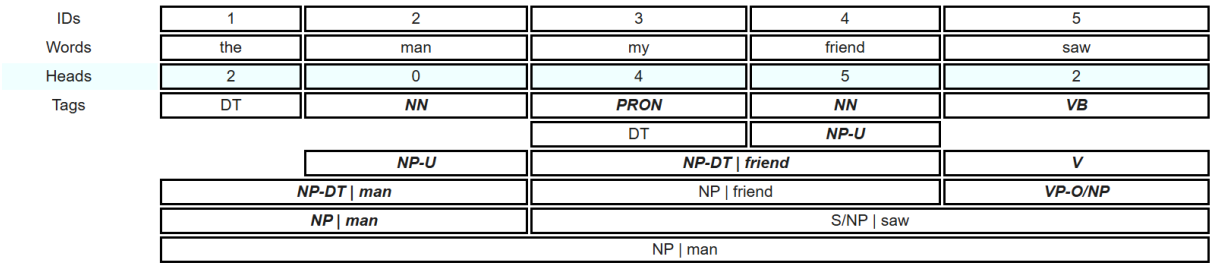

Figure 4: Slash representation derivation for VP with missing NP

Phrases missing a noun phrase and WH-questions.

Conclusions

In this study, we revisited grammar-based parsing, and examined whether it can provide new insights into the modern language processing landscape. For this purpose, we explored the possibility of using an inventory of explicit syntactic rules, as well as listings of lexical categories and features for closed-set words, as the basis for a bottom-up parsing approach. The pipeline built through the syntactic rule inventory and the bottom-up parsing algorithm was shown to be capable of parsing real-world data, with initially promising performance. It is important to note that the goal of this approach, at this stage, is not competitive parsing. However, we report its parsing outcomes, measured by UAS metrics as the baseline for future implementations.

Another important goal of this study is to leverage some of the knowledge acquired over decades in the field of syntax and other areas of theoretical linguistics. It is important to be able to frame this knowledge into computational frameworks, usable for NLP applications. We started addressing this goal as part of our qualitative evaluation, by feeding sentences exhibiting specific syntactic phenomena to the parser and evaluating its output accordingly. As an example, we attempted to represent the syntactic gap construction among the syntactic rules processed by the DPH parser, using GPSG slash notation. The qualitative results indicated that the parser can indeed process such constructions with the expected derivation. However, more detailed quantitative investigation is needed for how accurately the parser captures these constructions in real-world data.

This parsing approach shows the full grammatical derivation used to achieve the final structure. In fact, it produces multiple parse hypotheses, both for the sentence as a whole, and for each chunk of words. This will allow automatic detection of syntactic ambiguity and where there can be multiple meanings. It will also allow re-ranking these hypotheses to identify the most likely parse. In addition, this approach produces a richer combination of dependency and constituency structures, leveraging the strengths of both paradigms. It is compatible with modern parsing tools and evaluation schemes and metrics.

With this new approach, it is possible to expand to any new language. What is needed is the following: 1) An inventory of possible POS tags, lexical categories and features for this language; 2) An inventory of GPSG/HPSG rules for this language; 3) A framework for predicting POS tags for this language; and 4) For languages with rich morphology, also a framework for predicting the likely sub-tokenization of prefixes, suffixes and subwords

This work thus contributes to ongoing efforts to integrate linguistic formalisms with empirically testable computational modeling. Further, it allows more explainable parsing behavior, including identifying syntactic ambiguities and allowing closer inspection of both underlying constituency and dependency structures. Finally, it allows building language-specific rule inventory, to leverage theoretical syntactic knowledge possibly into low-resource and understudied languages.

Limitations

The results discussed in this paper are only related to English. The rules and features used in the system are hand crafted only as a sample and do not represent a comprehensive listing of English syntactic rules. The Parts of Speech tagger included was trained on training data from Universal Dependencies. All fine tuning of rules and algorithms was according to the UD training data. Results only pertain to corpora included in UD dev/test datasets.

Acknowledgments

References

Ulrich Callmeier. 2000. PET–a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–107.

Noam Chomsky. 1957. *Syntactic structures*. Walter de Gruyter.

Berthold Crysmann and Woodley Packard. 2012. Towards efficient HPSG generation for German, a non-configurational language. In *Proceedings of COLING 2012*, pages 695–710.

Anthony J. Fisher. 1989. Practical parsing of generalized phrase structure grammars. *Computational Linguistics*, 15(3):139–148.

Gerald Gazdar. 1985. *Generalized phrase structure grammar*. Harvard University Press.

Henk Harkema. 2000. A recognizer for minimalist grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 111–122.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142. ISBN: 2307-387X.

Daniel Jurafsky and James H. Martin. 2026. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*, 3rd edition.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686.

Zuchao Li, Junru Zhou, Hai Zhao, and Kevin Parnow. 2021. Head-driven Phrase Structure Parsing in O ($ n^ 3$) Time Complexity. *arXiv preprint arXiv:2105.09835*.

Thomas Mainguy. 2010. A probabilistic top-down parser for minimalist grammars. *arXiv preprint arXiv:1010.1826*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 523–530.

Duc-Vu Nguyen, Thang Chau Phan, Quoc-Nam Nguyen, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. 2024. An Attempt to Develop a Neural Parser based on Simplified Head-Driven Phrase Structure Grammar on Vietnamese. In *International Symposium on Information and Communication Technology*, pages 313–328. Springer.

Joakim Nivre. 2006. *Inductive dependency parsing*. Springer.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Jan Hajic, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the twelfth language resources and evaluation conference*, pages 4034–4043.

Carl Pollard and Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Eva Portelance, Amelia Bruno, Daniel Harasim, Leon Bergen, and Timothy J. O'Donnell. 2017. Grammar Induction for Minimalist Grammars using Variational Bayesian Inference: A Technical Report. *arXiv preprint arXiv:1710.11350*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th annual meeting of the association for computational linguistics: system demonstrations*, pages 101–108.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827.

Kentaro Torisawa, Kenji Nishida, Yusuke Miyao, and Jun-Ichi Tsujii. 2000. An HPSG parser with CFG filtering. *Natural Language Engineering*, 6(1):63–80. ISBN: 1469-8110.

John Torr, Miloš Stanojević, Mark Steedman, and Shay B. Cohen. 2019. Wide-coverage neural A* parsing for minimalist grammars. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 2486–2505.

Fei Xia and Martha Palmer. 2001. Converting dependency structures to phrase structures.

Yoshida Yoshida, Kazuhiro. 2005. Corpus-oriented development of japanese HPSG parsers. In *In Proceedings of the ACL Student Research Workshop, pp. 139-144.*

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.

Olga Zamaraeva and Carlos Gómez-Rodríguez. 2024. Revisiting Supertagging for faster HPSG parsing. In *In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pp. 11359-11374.*

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. *arXiv preprint arXiv:1907.02684*.

A Appendix