A Framework for (Under)specifying Dependency Syntax without Overloading Annotators

Nathan Schneider[†*] **Brendan O'Connor**[†] **Naomi Saphra**[†] **David Bamman**[†]
Manaal Faruqui[†] **Noah A. Smith**[†] **Chris Dyer**[†] **Jason Baldridge**[‡]
[†]School of Computer Science, Carnegie Mellon University
[‡]Department of Linguistics, The University of Texas at Austin

Abstract

We introduce a framework for lightweight dependency syntax annotation. Our formalism builds upon the typical representation for unlabeled dependencies, permitting a simple notation and annotation workflow. Moreover, the formalism encourages annotators to underspecify parts of the syntax if doing so would streamline the annotation process. We demonstrate the efficacy of this annotation on three languages and develop algorithms to evaluate and compare underspecified annotations.

1 Introduction

Computational representations for natural language syntax are borne of competing design considerations. When designing such representations, there may be a tradeoff between parsimony and expressiveness. A range of *linguistic theories* attract support due to differing purposes and aesthetic principles (Chomsky, 1957; Tesnière, 1959; Hudson, 1984; Sgall et al., 1986; Mel'čuk, 1988, *inter alia*). Formalisms concerned with tractable computation may care chiefly about *learnability* or *parsing efficiency* (Shieber, 1992; Sleator and Temperly, 1993; Kuhlmann and Nivre, 2006). Further considerations may include *psychological* and *evolutionary plausibility* (Croft, 2001; Tomasello, 2003; Steels et al., 2011; Fossum and Levy, 2012), integration with other representations such as semantics (Steedman, 2000; Bergen and Chang, 2005), or suitability for particular applications (e.g., translation).

Here we elevate *ease of annotation* as a primary design concern for a syntactic annotation formalism. Currently, a lack of annotated data is a huge bottleneck for robust NLP, standing in the way of parsers for social media text (Foster et al., 2011) and many low-resourced languages (to name two examples). Traditional syntactic annotation projects like the Penn Treebank (Marcus et al., 1993) or Prague Dependency Treebank (Hajič, 1998) require highly trained annotators and huge amounts of effort. Lowering the cost of annotation, by making it easier and more accessible, could greatly facilitate robust NLP in new languages and genres.

To that end, we design and test new, lightweight methodologies for syntactic annotation. We propose a *formalism*, **Fragmentary Unlabeled Dependency Grammar** (FUDG) for unlabeled dependency syntax that addresses some of the most glaring deficiencies of basic unlabeled dependencies (§2), with little added burden on annotators. FUDG requires minimal theoretical commitments, and can be supplemented with a project-specific style guide (we provide a brief one for English). We contribute a simple ASCII markup language—**Graph Fragment Language** (GFL; §3)—that allows annotations to be authored using any text editor, along with tools for validating, normalizing, and visualizing GFL annotations.[1]

An important characteristic of our framework is annotator flexibility. The formalism supports this by allowing *underspecification* of structural portions that are unclear or unnecessary for the purposes of a project. Fully leveraging this power requires new algorithms for evaluation, e.g., of inter-annotator agreement, where annotations are partial; such algorithms are presented in §4.[2]

Finally, small-scale case studies (§5) apply our framework (formalism, notation, and evaluations) to syntactically annotate web text in English, news in Malagasy, and dialogues in Kinyarwanda.

2 A Dependency Grammar for Annotation

Although dependency-based approaches to syntax play a major role in computational linguistics, the nature of dependency representations is far from uniform. Exemplifying one end of the spectrum is the Prague Dependency Treebank, which articulates an elaborate dependency-based syntactic the-

[*] Corresponding author: nschneid@cs.cmu.edu

[1] https://github.com/brendano/gfl_syntax/

[2] Parsing algorithms are left for future work.

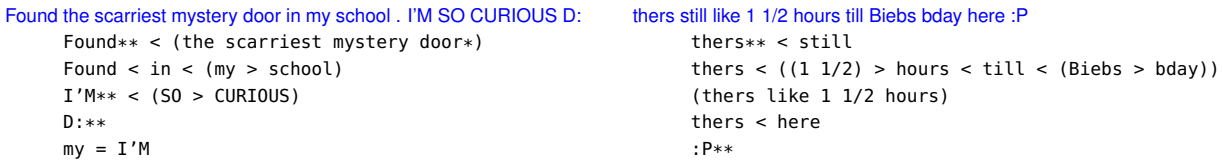


Figure 1: Two tweets with example GFL annotations. (The formalism and notation are described in §3.)

ory in a rich, multi-tiered formalism (Hajič, 1998; Böhmová et al., 2003). On the opposite end of the spectrum are the structures used in dependency parsing research which organize all the tokens of a sentence into a tree, sometimes with category labels on the edges (Kübler et al., 2009). Insofar as they reflect a theory of syntax, these **vanilla dependency grammars** provide a highly reductionist view of structure—indeed, parses used to train and evaluate dependency parses are often simplifications of Prague-style parses, or else converted from constituent treebanks.

In addition to the binary dependency links of vanilla dependency representations, we offer three devices to capture certain linguistic phenomena more straightforwardly:[3]

1. We make explicit the meaningful lexical units over which syntactic structure is represented. Our approach (a) allows punctuation and other extraneous tokens to be excluded so as not to distract from the essential structure; and (b) permits tokens to be grouped into shallow multiword lexical units.[4]

2. Coordination is problematic to represent with unlabeled dependencies due to its non-binary nature. A coordinating conjunction typically joins multiple expressions (conjuncts) with equal status, and other expressions may relate to the compound structure as a unit. There are several different conventions for forcing coordinate structures into a head-modifier straightjacket (Nivre, 2005; de Marneffe and Manning, 2008; Mareček et al., 2013). Conjuncts, coordinators, and shared dependents can be distinguished with edge labels; we equivalently use a special notation, permitting the coordinate structure to be automatically transformed with any of the existing conventions.[5]

3. Following Tesnière (1959), our formalism offers a simple facility to express anaphora-antecedent relations (a subset of semantic relationships) that are salient in particular syntactic phenomena such as relative clauses, appositives, and WH-expressions.

Underspecification. Our desire to facilitate lightweight annotation scenarios requires us to abandon the expectation that syntactic informants provide a complete parse for every sentence. On one hand, an annotator may be *uncertain* about the appropriate parse due to lack of expertise, insufficiently mature annotation conventions, or actual ambiguity in the sentence. On the other hand, annotators may be *indifferent* to certain phenomena. This can happen for a variety of reasons:

- Some projects may only need annotations of specific constructions. For example, building a semantic resource for events may require annotation of syntactic verb-argument relations, but not internal noun phrase structure.
- As a project matures, it may be more useful to annotate only infrequent lexical items.
- Semisupervised learning from partial annotations may be sufficient to learn complete parsers (Hwa, 1999; Clark and Curran, 2006).
- Beginning annotators may wish to focus on easily understood syntactic phenomena.
- Different members of a project may wish to specialize in different syntactic phenomena, reducing training cost and cognitive load.

Rather than treating annotations as invalid unless and until they are complete trees, we formally represent and reason about partial parse structures. Annotators produce **annotations**, which encode constraints on the (inferred) **analysis**, the parse structure, of a sentence. We say that a valid annotation **supports** (is compatible with) one or more **analyses**. Both annotations and analyses are represented as graphs (the graph representation is described below in §3.2). We require that the directed edges in an *analysis* graph must form a tree over all the lexical items in the sentence.[6] Less

[3] Some of this is inspired by the conventions of Reed-Kellogg *sentence diagramming*, a graphical dependency annotation system for English pedagogy (Reed and Kellogg, 1877; Kolln and Funk, 1994; Florey, 2006).

[4] The Stanford representation supports a limited notion of multiword expressions (de Marneffe and Manning, 2008). For simplicity, our formalism treats multiwords as unanalyzed (syntactically opaque) wholes, though some multiword expressions may have syntactic descriptions (Baldwin and Kim, 2010).

[5] Tesnière (1959) and Hudson (1984) similarly use special structures for coordination (Schneider, 1998; Sangati and Mazza, 2009).

[6] While some linguistic phenomena (e.g., relative clauses, control constructions) can be represented using non-tree

stringent well-formedness constraints on the *annotation* graph leave room for underspecification.

Briefly, an annotation can be underspecified in two ways: (a) an expression may not be attached to any parent, indicating it might depend on any non-descendant in a full analysis—this is useful for annotating sentences piece by piece; and (b) multiple expressions may be grouped together in a **fudge expression** (§3.3), a constraint that the elements form a connected subgraph in the full analysis while leaving the precise nature of that subgraph indeterminate—this is useful for marking relationships between chunks (possibly constituents).

A formalism, not a theory. Our framework for dependency grammar annotation is a syntactic *formalism*, but it is not sufficiently comprehensive to constitute a *theory* of syntax. Though it standardizes the basic treatment of a few basic phenomena, simplicity of the formalism requires us to be conservative about making such extensions. Therefore, just as with simpler formalisms, language- and project-specific conventions will have to be developed for specific linguistic phenomena. By embracing underspecified annotation, however, our formalism aims to encourage efficient corpus coverage in a nascent annotation project, without forcing annotators to make premature decisions.

3 Syntactic Formalism and GFL

In our framework, a syntactic **annotation** of a sentence follows an extended dependency formalism based on the desiderata enumerated in the previous section. We call our formalism **Fragmentary Unlabeled Dependency Grammar** (**FUDG**).

To make it simple to create FUDG annotations with a text editor, we provide a plain-text dependency notation called **Graph Fragment Language** (**GFL**). Fragments of the FUDG graph—nodes and dependencies linking them—are encoded in this language; taken together, these fragments describe the annotation in its entirety. The ordering of GFL fragments, and of tokens within each fragment, is of no formal consequence. Since the underlying FUDG representation is transparently related to GFL constructions, GFL notation will be introduced alongside the discussion of each kind of FUDG node.[7]

structures, we find that being able to alert annotators when they inadvertently violate the tree constraint is more useful than the expressive flexibility.

[7]In principle, FUDG annotations could be created with an alternative mechanism such as a GUI, as in Hajič et al. (2001).

3.1 Tokens

We expect a tokenized string, such as a sentence or short message. The provided tokenization is respected in the annotation. For human readability, GFL fragments refer to tokens as strings (rather than offsets), so all tokens that participate in an annotation must be unambiguous in the input.[8] A token may be referenced multiple times in the annotation.

3.2 Graph Encoding

Directed arcs. As in other dependency formalisms, **dependency arcs** are directed links indicating the syntactic headedness relationship between pairs of nodes. In GFL, directed arcs are indicated with angle brackets pointing from the dependent to its head, as in `black > cat` or (equivalently) `cat < black`. Multiple arcs can be chained together: `the > cat < black < jet` describes three arcs. Parentheses help group portions of a chain: `(the > cat < black < jet) > likes < fish` (the structure `black < jet > likes`, in which `jet` appears to have two heads, is disallowed). Note that another encoding for this structure would be to place the contents of the parentheses and the chain `cat > likes < fish` on separate lines. Curly braces can be used to list multiple dependents of the same head: `{cat fish} > likes`.

Anaphoric links. These undirected links join coreferent anaphora to each other and to their antecedent(s). In English this includes personal pronouns, relative pronouns (*who*, *which*, *that*), and anaphoric *do* and *so* (*Leo loves Ulla and **so does** Max*). This introduces a bit of semantics into our annotation, though at present we do not attempt to mark non-anaphoric coreference. It also allows a more satisfying treatment of appositives and relative clauses than would be possible from just the directed tree (the third example in figures 2 and 3).

Lexical nodes. Whereas in vanilla dependency grammar syntactic links are between pairs of **token nodes**, FUDG abstracts away from the individual tokens in the input. The lowest level of a FUDG annotation consists of **lexical nodes**, i.e.,

[8]If a word is repeated within the sentence, it must be indexed in the input string in order to be referred to from a fragment. In our notation, successive instances of the same word are suffixed with `~1`, `~2`, `~3`, etc. Punctuation and other tokens omitted from an annotation do not need to be indexed.

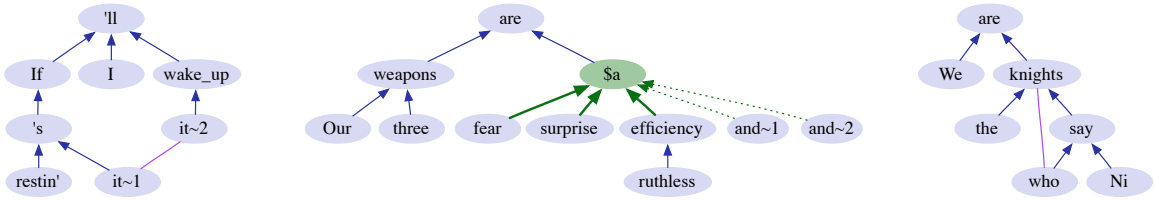


Figure 2: FUDG graphs corresponding to the examples in figure 3. The two special kinds of directed edges are for attaching conjuncts (bolded) and their coordinators (dotted) in a coordinate structure. Anaphoric links are undirected. The root node of each sentence is omitted.

```
If it~1 's restin' I 'll wake it~2 up .
  If < (it~1 > 's < restin')
  I > 'll < [wake up] < it~2
  If > 'll**
  it~1 = it~2
Our three weapons are fear and~1 surprise and~2
ruthless efficiency ...
  {Our three} > weapons > are < $a
  $a :: {fear surprise efficiency} :: {and~1 and~2}
  ruthless > efficiency
We are the knights who say ... Ni !
  We > are < knights < the
  knights < (who > say < Ni)
  who = knights
```

Figure 3: GFL for the FUDG graphs in figure 2.

lexical item occurrences. Every token node maps to 0 or 1 lexical nodes (punctuation, for instance, can be ignored).

A **multiword** is a lexical node incorporating more than one input token and is atomic (does not contain internal structure). A multiword node may group any subset of input tokens; this allows for multiword expressions which are not necessarily contiguous in the sentence (e.g., the verb-particle construction *make up* in *make the story up*). GFL notates multiwords with square brackets, e.g., `[break a leg]`.

Coordination nodes. Coordinate structures require at least two kinds of dependents: **coordinators** (i.e., lexical nodes for coordinating conjunctions—at least one per coordination node) and **conjuncts** (heads of the conjoined subgraphs—at least one per coordination node). The GFL annotation has three parts: a variable representing the node, a set of conjuncts, and a set of coordinator nodes. For instance, `$a :: {[peanut butter] honey} :: {and}` (*peanut butter and honey*) can be embedded within a phrase via the coordination node variable `$a`; *a [fresh [[peanut butter] and honey] sandwich] snack* would be formed with `{fresh $a} > sandwich > snack < a`. A graphical example of coordination can be seen in figure 2—note the bolded conjunct edges and the dotted coordinator edges. If the conjoined phrase as a whole takes modifiers, these are attached to the coordination node with regular directed arcs. For example, in *Sam really adores kittens and abhors puppies.*, the shared subject *Sam* and adverb *really* attach to the entire conjoined phrase. In GFL:

```
$a :: {adores abhors} :: {and}
Sam > $a < really
adores < kittens    abhors < puppies
```

Root node. This is a special top-level node used to indicate that a graph fragment constitutes a standalone utterance or a discourse connective. For an input with multiple utterances, the head of each should be designated with `**` to indicate that it attaches to the root.

3.3 Means of Underspecification

As discussed in §2, our framework distinguishes *annotations* from full syntactic *analyses*. With respect to dependency structure (directed edges), the former may underspecify the latter, allowing the annotator to commit only to a partial analysis.

For an annotation $\mathcal{A}$, we define *support*($\mathcal{A}$) to be the set of full analyses compatible with that annotation. A full analysis is required to be a directed rooted tree over all lexical nodes in the annotation. An annotation is *valid* if its support is non-empty.

The 2 mechanisms for dependency underspecification are unattached nodes and fudge nodes.

Unattached nodes. For any node in an annotation, the annotator is free to simply leave it not attached to any head. This is interpreted as allowing its head to be any other node (including the root node), subject to the tree constraint. We call a node's possible heads its **supported parents**. Formally, for an unattached node v in annotation $\mathcal{A}$, $suppParents_{\mathcal{A}}(v) = nodes(\mathcal{A}) \setminus (\{v\} \cup descendants(v))$.

Fudge nodes. Sometimes, however, it is desirable to represent a sort of skeletal structure without filling in all the details. A **fudge expression** (FE) asserts that a group of nodes (the expression's **members**) belong together in a *connected subgraph*, while leaving the internal structure of that subgraph unspecified.[9] The notation

[9]This underspecification semantics is, to the best of our knowledge, novel, though it has been proposed that connected dependency subgraphs (known as *catenae*) are of theoretical importance in syntax (Osborne et al., 2012).

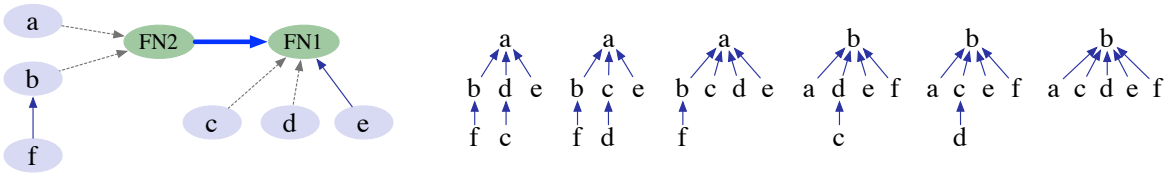


Figure 4: *Left:* An annotation graph with 2 fudge nodes and 6 lexical nodes; it can be encoded with GFL fragments `((a b)* c d) < e` and `b < f`. *Right:* All of its supported analyses: $prom(\mathcal{A}) = 6$. $com(\mathcal{A}) = 1 - \frac{\log 6}{\log 75} = .816$.

for this is a list of two or more nodes within parentheses: an annotation for *Few if any witches are friends with Maria.* might contain the FE `(Few if any)` so as to be compatible with the structures `Few < if < any`, `Few > if > any`, etc.—but *not*, for instance, `Few > witches < any`. In the FUDG graph, this is represented with a **fudge node** to which members are attached by special **member arcs**. Fudge nodes may be linked to other nodes: the GFL fragment `(Few if any) > witches` is compatible with `(Few < if < any) > witches`, `(Few < (if > any)) > witches`, and so forth.

Properties. Let **f** be a fudge expression. From the connected subgraph definition and the tree constraint on analyses, it follows that:

- Exactly 1 member of **f** must, in any compatible analysis, have a parent that is not a member of **f**. Call this node the **top** of the fudge expression, denoted f^*. f^* dominates all other members of **f**; it can be considered **f**'s "internal head."
- **f** does not necessarily form a full subtree. Any of its members may have dependents that are not themselves members of the fudge expression. (Such dependencies can be specified in additional GFL fragments.)

Top designation. A single member of a fudge expression may optionally be designated as its top (internal head). This is specified with an asterisk: `(Few* if any) > witches` indicates that *Few* must attach to *witches* and also dominate both *if* and *any*. In the FUDG graph, this is represented with a special **top arc** as depicted in bold in figure 4.

Nesting. One fudge expression may nest within another, e.g. `(Few (if any)) > witches`; the word analyzed as attaching to `witches` might be `Few` or whichever of `(if any)` heads the other. A nested fudge expression can be designated as top: `(Vanishingly few (if any)*)`.

Modifiers. An arc attaching a node to a fudge expression as a whole asserts that the external node should modify the *top* of the fudge expression (whether or not that top is designated in the annotation). For instance, two of the interpretations of *British left waffles on Falklands* would be preserved by specifying `British > left` and `(left waffles) < on < Falklands`. Analyses `British > left < waffles < on < Falklands` and `(British > left < on < Falklands) > waffles` would be excluded because the preposition does not attach to the head of `(left waffles)`.[10]

Multiple membership. A node may be a member of multiple fudge expressions, or a member of an FE while attached to some other node via an explicit arc. Each connected component of the FUDG graph is therefore a polytree (not necessarily a tree). The annotation graph minus all member edges of fudge nodes and all (undirected) anaphoric links must be a directed tree or forest.

Enumerating supported parents. Fudge expressions complicate the procedure for listing a node's supported parents (see above). Consider an FE **f** having some member v. v might be the top of **f** (unless some other node is so designated), in which case anything the fudge node can attach to is a potential parent of v. If some node other than v might be the top of **f**, then v's head could be any member of **f**. Below (§4.1) we develop an algorithm for enumerating supported parents for any annotation graph node.

4 Annotation Evaluation Measures

For an annotation task which allows for a great deal of latitude—as in our case, where a syntactic annotation may be full or partial—quantitative evaluation of data quality becomes a challenge. In the context of our formalism, we propose measures that address:

- **Annotation efficiency**, quantified in terms of annotator productivity (tokens per hour).
- The **amount of information** in an underspecified annotation. Intuitively, an annotation that flirts with many full analyses conveys less syntactic information than one which supports few analyses. We define an annotation's **promiscuity** to be the number of full analyses it supports, and develop an algorithm to compute it (§4.1).

[10] Not all attachment ambiguities can be precisely encoded in FUDG. For instance, there is no way to forbid an attachment to a word that lies along the path between the possible heads. The best that can be done given a sentence like *They conspired to defenestrate themselves on Tuesday.* is `They > conspired < to < defenestrate < themselves` and `(conspired* to defenestrate (on < Tuesday))`.

- **Inter-annotator agreement** between two partial annotations. Our measures for dependency structure agreement (§4.2) incorporate the notion of promiscuity.

We test these evaluations on our pilot annotation data in the case studies (§5).

4.1 Promiscuity vs. Commitment

Given a FUDG annotation of a sentence, we quantify the extent to which it underspecifies the full structure by counting the number of analyses that are compatible with the constraints in the annotation. We call this number the **promiscuity** of the annotation. Each analysis tree is rooted with the root node and must span all lexical nodes.[11]

A naïve algorithm for computing promiscuity would be to enumerate all directed spanning trees over the lexical nodes, and then check each of them for compatibility with the annotation. But this quickly becomes intractable: for n nodes, one of which is designated as the root, there are n^{n-2} spanning trees. However, we can filter out edges that are known to be incompatible with the annotation *before* searching for spanning trees. Our "upward-downward" method for constructing a graph of **supported edges** first enumerates a set of candidate top nodes for every fudge expression, then uses that information to infer a set of supported parents for every node.[12] The supported edge graph then consists of vertices $\mathit{lexnodes}(\mathcal{A}) \cup \{\textsc{root}\}$ and edges $\bigcup_{v \in \mathit{lexnodes}(\mathcal{A})} \{(v \rightarrow v') \; \forall \; v' \in \mathit{suppParents}_{\mathcal{A}}(v)\}$. From this graph we can count all directed spanning trees in cubic time using Kirchhoff's matrix tree theorem (Chaiken and Kleitman, 1978; Smith and Smith, 2007; Margoliash, 2010).[13] If some lexical node has no supported parents, this reflects conflicting constraints in the annotation, and no spanning tree will be found.

Promiscuity will tend to be higher for longer sentences. To control for this, we define a second quantity, the annotation's **commitment quotient** (commitment being the opposite of promiscuity), which normalizes for the number of possible spanning trees given the sentence length. The commitment quotient for an annotation of a sentence with $n-1$ lexical nodes and one root node is given by:

$$\mathit{com}(\mathcal{A}) = 1 - \frac{\log \mathit{prom}(\mathcal{A})}{\log n^{n-2}}$$

(the logs are to attenuate the dominance of the exponential term). This will be 1 if only a single tree is supported by the annotation, and 0 if the annotation does not constrain the structure at all. (If the constraints in the annotation are internally inconsistent, then promiscuity will be 0 and commitment undefined.) In practice, there is a tradeoff between efficiency and commitment: more detailed annotations require more time. The value of minimizing promiscuity will therefore depend on the resources and goals of the annotation project.

4.2 Inter-Annotator Agreement

FUDG can encode flat groupings and coreference at the lexical level, as well as syntactic structure over lexical items. Inter-annotator agreement can be measured separately for each of these facets. Pilot annotator feedback indicated that our initial lexical-level guidelines were inadequate, so we focus here on measuring structural agreement pending further clarification of the lexical conventions.

Attachment accuracy, a standard measure for evaluating dependency parsers, cannot be computed between two FUDG annotations if either of them underspecifies any part of the dependency structure. One solution is to consider the intersection of supported full trees, in the spirit of our promiscuity measure. For annotations $\mathcal{A}_1$ and $\mathcal{A}_2$ of sentence $\mathbf{s}$, one annotation's supported analyses can be enumerated and then filtered subject to the constraints of the other annotation. The tradeoff between inter-annotator compatibility and commitment can be accounted for by taking their product, i.e. $\mathit{comPrec}(\mathcal{A}_1 \mid \mathcal{A}_2) = \mathit{com}(\mathcal{A}_1)\frac{|\mathit{supp}(\mathcal{A}_1) \cap \mathit{supp}(\mathcal{A}_2)|}{|\mathit{supp}(\mathcal{A}_1)|}$.

A limitation of this support-intersection approach is that if the two annotations are not compatible, the intersection will be empty. A more fine-grained approach is to decompose the comparison by lexical node: we generalize attachment accuracy with $\mathit{softComPrec}(\mathcal{A}_1 \mid \mathcal{A}_2) = \mathit{com}(\mathcal{A}_1)\frac{\sum_{\ell \in \mathbf{s}} \bigcap_{i \in \{1,2\}} \mathit{suppParents}_{\mathcal{A}_i}(\ell)}{\sum_{\ell \in \mathbf{s}} \mathit{suppParents}_{\mathcal{A}_1}(\ell)}$, computing $\mathit{com}(\cdot)$ and $\mathit{suppParents}(\cdot)$ as in the previous section. As lexical nodes may differ between the two annotations, a reconciliation step is required

[11] This measure assumes a fixed lexical analysis (set of lexical nodes) and does not consider anaphoric links. Coordinate structures are simplified into ordinary dependencies, with coordinate phrases headed by the coordinator's lexical node. If a coordination node has multiple coordinators, one is arbitrarily chosen as the head and the others as its dependents.

[12] Python code for these algorithms appears in Schneider et al. (2013) and the accompanying software release.

[13] Due to a technicality with non-member attachments to fudge nodes, for some annotations this is only an upper bound on promiscuity; see Schneider et al. (2013).

Language	Tokens	Rate (tokens/hr)
English Tweets (partial)	667	430
English Tweets (full)	388	250
Malagasy	4,184	47
Kinyarwanda	8,036	80

Table 1: Productivity estimates from pilot annotation project. All annotators were native speakers of English.

to compare the structures: multiwords proposed in only one of the two annotations are converted to fudge expressions. Tokens annotated by neither annotator are ignored. Like with the promiscuity measure, we simplify coordinate structures to ordinary dependencies (see footnote 11).

5 Case Studies

5.1 Annotation Time

To estimate annotation efficiency, we performed a pilot annotation project consisting of annotating several hundred English tweets, about 1,000 sentences in Malagasy, and a further 1,000 sentences in Kinyarwanda.[14] Table 1 summarizes the number of tokens annotated and the effort required. For the two Twitter cases, the same annotator was first permitted to do partial annotation of 100 tweets, and then spend the same amount of time doing a complete annotation of all tokens. Although this is a very small study, the results clearly suggest she was able to make much more rapid progress when partial annotation was an option.[15]

This pilot study helped us to identify linguistic phenomena warranting specific conventions: these include WH-expressions, comparatives, vocatives, discourse connectives, null copula constructions, and many others. We documented these cases in a 20-page style guide for English,[16] which informed the subsequent pilot studies discussed below.

5.2 Underspecification and Agreement

We annotated 2 small English data samples in order to study annotators' use of underspecification. The first is drawn from Owoputi et al.'s 2013 Twitter part-of-speech corpus; the second is from the REVIEWS portion of the English Web Treebank (EWTB) (Bies et al., 2012). (Our annotators only saw the tokenized text.) Both datasets are informal and conversational in nature, and are dominated by short messages/sentences. In spite of their brevity, many of the items were deemed to contain multiple "utterances," which we define to include discourse connectives and emoticons (at best marginal parts of the syntax); utterance heads are marked with $**$ in figure 1.

	Omit.				*prom* Hist.				Mean
	1Ws	MWs	Tkns	FEs	1	>1	≥10	≥10^2	*com*
TWEETS	*60 messages, 957 tokens*								
A	597	56	304	23	43	17	11	5	.96
B	644	47	266	28	37	23	12	6	.95
REVIEWS	*55 sentences, 778 tokens*								
A	609	33	136	2	53	2	2	1	1.00
C ∩ D	643	19	116	114	11	44	38	21	.82
T	704	—	74	—	55	0	0	0	1

Table 2: Measures of our annotation samples. Note that annotator "D" specialized in noun phrase–internal structure, while annotator "C" specialized in verb phrase/clausal phenomena; C ∩ D denotes the combination of their annotation fragments. "T" denotes our dependency conversion of the English Web Treebank parses. (The value 1.00 was rounded up from .9994.)

Table 2 indicates the sizes of the two data samples, and gives statistics over the output of each annotator: total counts of single-word and multiword lexical nodes, tokens not represented by any lexical node, and fudge nodes; as well as a histogram of promiscuity counts and the average of commitment quotients (see §4.1). For instance, the two sets of annotations obtained for the TWEETS sample used underspecification in 17/60 and 23/60 tweets, respectively, though the promiscuity rarely exceeded 100 compatible trees per annotation. Examples can be seen in figure 1, where annotator "A" marked only the noun phrase head for *the scarriest mystery door*, opted not to choose a head within the quantity *1 1/2*, and left ambiguous the attachment of the hedge *like*. The strong but not utter commitment to the dependency structure is reflected in the mean commitment quotients for this dataset, both of which exceed 0.95.

Inter-annotator agreement (IAA) is quantified in table 3. The row marked A ∼ B, for instance, considers the agreement between annotator "A" and annotator "B". Measuring IAA on the dependency structure requires a common set of lexical nodes, so a **lexical reconciliation** step ensures that (a) any token used by either annotation is present in both, and (b) no multiword node is present in only one annotation—solved by relaxing incompatible multiwords to FEs (which increases promiscuity). For TWEETS, lexical reconciliation

[14] Malagasy is a VOS Austronesian language spoken by 15 million people, mostly in Madagascar. Kinyarwanda is an SVO Bantu language spoken by 12 million people mostly in Rwanda. All annotations were done by native speakers of English. The Kinyarwanda and Malagasy annotators had basic proficiency in these languages.

[15] As a point of comparison, during the Penn Treebank project, annotators corrected the syntactic bracketings produced by a high-quality hand-written parser (Fidditch) and achieved a rate of only 375 tokens/hour using a specialized GUI interface (Marcus et al., 1993).

[16] Included with the data and software release (footnote 1).

thus reduces the commitment averages for each annotation—to a greater extent for annotator "A" (.96 in table 2 vs. .82 in table 3) because "A" marked more multiwords. An analysis fully compatible with both annotations exists for only 27/60 sentences; the finer-grained *softComPrec* measure (§4.2), however, offers insight into the balance between commitment and agreement.

Qualitatively, we observe three leading causes of incompatibilities (disagreements): obvious annotator mistakes (such as *the* marked as a head); inconsistent handling of verbal auxiliaries; and uncertainty whether to attach expressions to a verb or the root node, as with *here* in figure 1.[17] Annotators noticed occasional ambiguous cases and attempted to encode the ambiguity with fudge expressions: *again* in the tweet *maybe put it off until you feel like ~ talking again ?* is one example. More often, fudge expressions proved useful for syntactically difficult constructions, such as those shown in figure 1 as well as: *2 shy of breaking it*, *asked what tribe I was from*, *a $ 13 / day charge*, *you two*, and *the most awkward thing ever*.

5.3 Annotator Specialization

As an experiment in using underspecification for labor division, two of the annotators of Reviews data were assigned specific linguistic phenomena to focus on. Annotator "D" was tasked with the internal structure of base noun phrases, including resolving the antecedents of personal pronouns. "C" was asked to mark the remaining phenomena—i.e., utterance/clause/verb phrase structure—but to mark base noun phrases as fudge expressions, leaving their internal structure unspecified. Both annotators provided a full lexical analysis. For comparison, a third individual, "A," annotated the same data in full. The three annotators worked completely independently.

Of the results in tables 2 and 3, the most notable difference between full and specialized annotation is that the combination of independent specialized annotations (C ∩ D) produces somewhat higher promiscuity/lower commitment. This is unsurprising because annotators sometimes overlook relationships that fall under their specialty.[18] Still, annotators reported that specialization made the task less burdensome, and the specialized annotations did prove complementary to each other.[19]

	com			*softComPrec*		
IAA	1	2	$N_{\lvert\cap\rvert>0}$	1|2	2|1	F_1
Tweets (N=60)						
A ~ B	.82	.91	27	.57	.72	.63
Reviews (N=55)						
A ~ (C ∩ D)	.95	.76	30	.64	.40	.50
A ~ T	.92	1	26	.48	.91	.63
(C ∩ D) ~ T	.73	1.00	28	.33	.93	.49

Table 3: Measures of inter-annotator agreement. Annotator labels are as in table 2. Per-annotator *com* (with lexical reconciliation) and inter-annotator *softComPrec* are aggregated over sentences by arithmetic mean.

5.4 Treebank Comparison

Though the annotators in our study were native speakers well acquainted with representations of English syntax, we sought to quantify their agreement with the expert treebankers who created the EWTB (the source of the Reviews sentences). We converted the EWTB's constituent parses to dependencies via the PennConverter tool (Johansson and Nugues, 2007),[20] then removed punctuation.

Agreement with the converted treebank parses appears in the bottom two rows of table 3. Because the EWTB commits to a single analysis, precision scores are quite lopsided. Most of its attachments are consistent with our annotations (*softComPrec* > 0.9), but these allow many additional analyses (hence the scores below 0.5).

6 Conclusion

We have presented a framework for simple dependency annotation that overcomes some of the representational limitations of unlabeled dependency grammar and embraces the practical realities of resource-building efforts. Pilot studies (in multiple languages and domains, supported by a human-readable notation and a suite of open-source tools) showed this approach lends itself to rapid annotation with minimal training.

The next step will be to develop algorithms exploiting these representations for learning parsers. Other future extensions might include additional expressive mechanisms (e.g., multi-headedness, labels), crowdsourcing of FUDG annotations (Snow et al., 2008), or even a semantic counterpart to the syntactic representation.

[17] Another example: Some uses of conjunctions like *and* and *so* can be interpreted as either phrasal coordinators or discourse connectives (cf. The PDTB Research Group, 2007).

[18] A more practical and less error-prone approach might be for specialists to work sequentially or collaboratively (rather than independently) on each sentence.

[19] In fact, for only 2 sentences did "C" and "D" have incompatible annotations, and both were due to simple mistakes that were then fixed in the combination.

[20] We ran PennConverter with options chosen to emulate our annotation conventions; see Schneider et al. (2013).

Acknowledgments

We thank Lukas Biewald, Yoav Goldberg, Kyle Jerro, Vijay John, Lori Levin, André Martins, and several anonymous reviewers for their insights. This research was supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533 and by NSF grant IIS-1054319.

References

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, Boca Raton, FL.

Benjamin K. Bergen and Nancy Chang. 2005. Embodied Construction Grammar in simulation-based language understanding. In Jan-Ola Östman and Mirjam Fried, editors, *Construction grammars: cognitive grounding and theoretical extensions*, pages 147–190. John Benjamins, Amsterdam.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. Technical Report LDC2012T13, Linguistic Data Consortium, Philadelphia, PA.

Alena Böhmová, Jan Hajič, Eva Hajičová, Barbora Hladká, and Anne Abeillé. 2003. The Prague Dependency Treebank: a three-level annotation scenario. In *Treebanks: building and using parsed corpora*, pages 103–127. Springer.

Seth Chaiken and Daniel J. Kleitman. 1978. Matrix Tree Theorems. *Journal of Combinatorial Theory, Series A*, 24(3):377–381.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton, La Haye.

Stephen Clark and James Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL 2006)*, pages 144–151. Association for Computational Linguistics, New York City, USA.

William Croft. 2001. *Radical Construction Grammar: Syntactic Theory in Typological Perspective*. Oxford University Press, Oxford.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual. `http://nlp.stanford.edu/downloads/dependencies_manual.pdf`.

Kitty Burns Florey. 2006. *Sister Bernadette's Barking Dog: The quirky history and lost art of diagramming sentences*. Melville House, New York.

Victoria Fossum and Roger Levy. 2012. Sequential vs. hierarchical syntactic models of human incremental sentence processing. In *Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2012)*, pages 61–69. Association for Computational Linguistics, Montréal, Canada.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. #hardtoparse: POS Tagging and Parsing the Twitterverse. In *Proceedings of the 2011 AAAI Workshop on Analyzing Microtext*, pages 20–25. AAAI Press, San Francisco, CA.

The PDTB Research Group. 2007. The Penn Discourse Treebank 2.0 annotation manual. Technical Report IRCS-08-01, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA.

Jan Hajič. 1998. Building a syntactically annotated corpus: the Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press, Prague.

Jan Hajič, Barbora Vidová Hladká, and Petr Pajas. 2001. The Prague Dependency Treebank: annotation structure and support. In *Proceedings of the IRCS Workshop on Linguistic Databases*, pages 105–114. University of Pennsylvania, Philadelphia, USA.

Richard A. Hudson. 1984. *Word Grammar*. Blackwell, Oxford.

Rebecca Hwa. 1999. Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pages 73–79. Association for Computational Linguistics, College Park, Maryland, USA.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek, and Mare Koit, editors, *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA-2007)*, pages 105–112. Tartu, Estonia.

Martha Kolln and Robert Funk. 1994. *Understanding English Grammar*. Macmillan, New York.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Number 2 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool, San Rafael, CA.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 507–514. Association for Computational Linguistics, Sydney, Australia.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Daniel Zeman, Zdeněk Žabokrtský, and Jan Hajič. 2013. Cross-language study on influence of coordination style on dependency parsing performance. Technial Report 49, ÚFAL MFF UK.

Jonathan Margoliash. 2010. Matrix-Tree Theorem for directed graphs. `http://www.math.uchicago.edu/~may/VIGRE/VIGRE2010/REUPapers/Margoliash.pdf`.

Igor Aleksandrovič Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. SUNY Press, Albany, NY.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. Technical Report MSI report 05133, Växjö University School of Mathematics and Systems Engineering, Växjö, Sweden.

Timothy Osborne, Michael Putnam, and Thomas Groß. 2012. Catenae: introducing a novel unit of syntactic analysis. *Syntax*, 15(4):354–396.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390. Association for Computational Linguistics, Atlanta, Georgia, USA.

Alonzo Reed and Brainerd Kellogg. 1877. *Work on English grammar & composition*. Clark & Maynard.

Federico Sangati and Chiara Mazza. 2009. An English dependency treebank à la Tesnière. In Marco Passarotti, Adam Przepiórkowski, Savina Raynaud, and Frank Van Eynde, editors, *Proceedings of the Eigth International Workshop on Treebanks and Linguistic Theories*, pages 173–184. EDUCatt, Milan, Italy.

Gerold Schneider. 1998. *A linguistic comparison of constituency, dependency and link grammar*. Master's thesis, University of Zurich.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A. Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under)specifying dependency syntax without overloading annotators. arXiv:1306.2091 [cs.CL]. `arxiv.org/pdf/1306.2091`.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in its Semantic and Pragmatic Aspects*. Reidel, Dordrecht and Academia, Prague.

Stuart M. Shieber. 1992. *Constraint-Based Grammar Formalisms*. MIT Press, Cambridge, MA.

Daniel Sleator and Davy Temperly. 1993. Parsing English with a link grammar. In *Proceedings of the Third International Workshop on Parsing Technology (IWPT'93)*, pages 277–292. Tilburg, Netherlands.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 132–140. Association for Computational Linguistics, Prague, Czech Republic.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast — but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 254–263. Association for Computational Linguistics, Honolulu, Hawaii.

Mark Steedman. 2000. *The Syntatic Process*. MIT Press, Cambridge, MA.

Luc Steels, Jan-Ola Östman, and Kyoko Ohara, editors. 2011. *Design patterns in Fluid Construction Grammar*. Number 11 in Constructional Approaches to Language. John Benjamins, Amsterdam.

Lucien Tesnière. 1959. *Eléments de Syntaxe Structurale*. Klincksieck, Paris.

Michael Tomasello. 2003. *Constructing a Language: A Usage-Based Theory of Language Acquisition*. Harvard University Press, Cambridge, MA.