Deriving Multi-Headed Planar Dependency Parses from Link Grammar Parses[*]

Juneki Hong
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
juneki@cs.cmu.edu

Jason Eisner
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218, USA
jason@cs.jhu.edu

Abstract

Under multi-headed dependency grammar, a parse is a connected DAG rather than a tree. Such formalisms can be more syntactically and semantically expressive. However, it is hard to train, test, or improve multi-headed parsers because few multi-headed corpora exist, particularly for the projective or planar case. To help fill this gap, we observe that link grammar already produces *undirected* planar graphs, and so we wanted to determine whether these could be converted into directionalized dependency parses. We use Integer Linear Programming to assign consistent directions to the labeled links in a corpus of several thousand parses produced by the Link Grammar Parser, which has broad-coverage hand-written grammars of English as well as Russian and other languages. We find that such directions can indeed be consistently assigned in a way that yields valid multi-headed dependency parses. The resulting parses in English appear reasonably linguistically plausible, though differing in style from CoNLL-style parses of the same sentences; we discuss the differences.

1 Motivation

Link Grammar [29] is a syntactic formalism in which a parse of a sentence is an undirected, edge-labeled, planar graph. The labeled edges of the graph represent syntactic relationships among the words. The vertices of the graph are simply the words $1, 2, \ldots, n$ of the sentence, along with a distinguished "root" vertex 0.

The small Link Grammar community has invested effort in creating link grammars for several languages. In this short paper, we consider whether their *undirected* parses can be converted automatically to *directed* ones. We have three motivations:

[*]This material is based upon work supported by the National Science Foundation under Grant No. 1423276. The work was mainly conducted while the first author was at Johns Hopkins University.

1. We were curious about the relation between link grammar annotation and dependency grammar annotation. We suspected that the link grammar parses could be interpreted as multi-headed dependency grammar parses. Although the link grammar authors did not bother to specify directions for the different edge types, we suspected that they essentially had such directions in mind.

2. Our problem provides a good case study for how to automatically enrich a corpus. Hand-constructed grammars or corpora sometimes provide a lighter level of annotation than desired. In our setting, the edges lack directions; in other settings, the syntactic categories may be coarser than desired, or some relations may be omitted. One may want to automatically enrich the annotation in such cases, whether by doing some kind of learning [17, et seq.], or by exploiting implicit constraints. In this paper, we use Integer Linear Programming to exploit implicit constraints of consistency and acyclicity.

3. The resulting parses may be useful data for experimenting with new parsing *algorithms*. There has been a good deal of recent research on projective dependency parsing, variously using global optimization or sequential classification (see [16, 3, 5] for surveys). Some of these algorithms could be extended to the *multi-headed* case, which is of linguistic and computational interest for reasons discussed below. However, for training and testing such algorithms, one would need a plausible sample of multi-headed projective dependency parses of real sentences. Our method cheaply manufactures such a sample, to compensate for the current lack of gold-standard data of this form.

Our automatic method for reconstructing the latent directions also had an unexpected benefit. It revealed an inconsistency in the hand-written English link grammar, regarding the handling of embedded sentences with missing (PRO) subjects.

2 Multi-Headed Dependency Parsing

Dependency parsing maps a sentence to a directed graph whose vertices are the words $1, 2, \ldots, n$ of the sentence along with a distinguished "root" vertex 0. A labeled directed edge $u \xrightarrow{L} v$ or $v \xleftarrow{L} u$ indicates that the "child" v is some kind of argument or modifier of its "parent" u. The edge label L indicates the specific syntactic or semantic relationship between the two words.

In the special case $u = 0$, the edge designates v as playing some special top-level role in the sentence, e.g., as the main verb. We disallow $v = 0$.

As discussed by [13, 9], one might impose various requirements on the parse graph:

- SINGLE-HEAD: Each word has ≤ 1 parent.
- ACYCLIC: There are no directed cycles.

- CONNECTED: Each pair of words has a undirected path between them.
- REACHABLE: Each word can be reached from 0 by a directed path (which implies CONNECTED). Note that 0 may have multiple children.
- PLANAR: edges may not "cross." That is, if there are edges between i, j and between u, v, where $i < u < j$, then PLANAR requires $i \leq v \leq j$.

It is common to impose all of these requirements at once, leading to a *projective dependency parser* that produces projective trees rooted at 0. However, parsing algorithms can be devised that relax any or all of the requirements [9].

In this paper, we are interested in relaxing the SINGLE-HEAD requirement while preserving all the others. This is the setting of *multi-headed projective dependency parsing*. Just as in the single-headed case, the other requirements ensure that all edges are projective. (A projective edge is one where the parent is an ancestor of all words between the parent and the child [19].)

Relaxing SINGLE-HEAD means that the parse can have more than n edges, allowing it to express more relationships between words. In English, for example, here are some constructions that seem to call for a multi-headed analysis:

control In *"Jill likes to skip,"* the word *Jill* is the subject of two verbs. In *"Jill persuaded Jack to skip," Jack* is the object of one verb and the subject of another. Without recognizing this, our parser would miss the syntactic invariants that *skip* always has a subject and *persuaded* always has an object. It would also be unable to exploit the selectional preferences of both verbs to help disambiguate the parse. This is why we prefer to make the parser aware of multi-headedness, rather than using a single-headed parser and then extracting the additional semantic roles from its output.

relativization In *"The boy that Jill skipped with fell down,"* the word *boy* is the object of *with* as well as the subject of *fell*. Without recognizing this, we would miss the syntactic invariant that *with* always has an object.

conjunction In *"Jack and Jill went up the hill," Jack* and *Jill* serve as the two arguments to *and*, but they are also semantically subjects of *went*. Without recognizing this, we would have no (local) reason for expecting the arguments of *and* to be nouns.

In linguistics, it is common to analyze some of these structures using trees with "empty categories." The subject of *skip* is taken to be a silent morpheme *PRO*: *"Jill$_i$ likes PRO$_i$ to skip."* However, this is no longer a tree if one considers the implicit undirected edge between *Jill* and *PRO* (denoted by their shared index *i*). Our simpler representation contracts this coreference edge, eliminating *PRO* and creating a *Jill* $\leftarrow$ *skip* link.

An anonymous reviewer objects to research on projective parsing algorithms, since the PLANAR restriction is linguistically questionable even for single-headed parsing, and even more so for multi-headed parsing. However, efficient algorithms often exist in the projective case, and these projective algorithms—which are typically first developed on a projective corpus of the sort that we will construct—can be useful even when the true parses are not quite projective. Natural-language parses have a low rate of non-projective edges [19] and the non-projective parses tend to be "almost projective" [25]. Thus, one can apply a fast projective parser as an approximate method, or as one ingredient in a more complex model [15], or as the first step in a coarse-to-fine or stacking architecture [27, 16] in order to obtain preliminary edge scores that are then supplied to a non-projective parser. Another approach is to transform non-projective parses into a projective annotation style so that projective parsers can be used [22, 20].

3 Link Grammars

Graph representations of syntactic and semantic structure have been widely considered of late [7, 6, 10, 2, 23]. A few past NLP papers have explored multi-headed dependency parsing [4, 18, 28, 9]. They constructed their multi-headed dependency corpora by automatically converting from other formats such as HPSG. Currently there seem to be no corpora that were directly annotated in this form, other than the Danish Dependency Treebank [12].

The above work considered non-projective parses. It seems at first that no one has worked out annotation conventions for *projective* multi-headed dependency parsing. However, this is only half-true. Link Grammar [29] is a grammar-based formalism for projective dependency parsing with *undirected* links. It produces undirected connected planar graphs. Annotation conventions are implicit in the detailed lexicon for the Link Grammar Parser [30]. The 122 link types in the English lexicon are documented at `http://www.abisource.com/projects/link-grammar/dict/summarize-links.html`, which specifies for every word a constraint on the *sequence* of labeled leftward and rightward edges attached to it. As remarked by [8], this is analogous to dependency grammar's use of head automata to constrain a word's sequence of left and right children. For example, in *"The boy that Jill skipped with fell down,"* the word *with* uses a lexical entry that requires it to link to a governing verb to the left, an extracted object farther to the left, and nothing to the right. Each entry has a hand-assigned cost in {0,1,2}, and the parser finds the parse of minimum total cost [30, 31].

Given a link grammar parse, it would be straightforward to convert it to an acyclic dependency parse by orienting all edges rightward. However, the result may violate the REACHABLE constraint. Instead we could orient all edges by depth-first search from the root node, which yields a DAG satisfying all our constraints. However, this might result in inconsistent annotation conventions, with some `S`-labeled links pointing from subject to verb and others from verb to subject.

In the English link grammar, an `S` edge encodes a "subject-verb" relation *whose left word serves as the subject.* We would expect verbs to point to their subject arguments in dependency grammar, and so we surmise that all `S` links should be interpreted as pointing leftward (from verb to subject: *"Jack* $\xleftarrow{S}$ *is falling"*).

In general, we supposed that the link grammar lexicon designers actually had a *consistent* direction in mind for each *edge label*. This does not imply that English subjects must always appear to the left of the verb! The link grammar designers took care to use a distinct `SI` label in cases of subject-verb inversion, and we surmise that `SI` links are intended to point rightward (again from verb to subject: *"Is* $\xrightarrow{SI}$ *Jack falling?"*). Similarly, different edge labels are used for English "object-verb" relations according to whether it is the left or the right word that serves as the object. These labels are presumably intended to encode different edge directions.

Our goal in this paper is to recover these implicit directions by global optimization. We seek a fixed mapping from labels to directions such that link grammar parses become directed dependency parses that satisfy all of our constraints.

Our first thought was to seek a direction mapping such that no parsed word sequence allowed by the link grammar lexicon could possibly violate our constraints after directions were imposed. This is a well-defined constraint programming problem. For example, to prevent cyclicity, we would require (roughly speaking) that no word type in the lexicon could follow a sequence of directed rightward links through other word types and then a leftward link back to itself.

However, we feared that there would not be a feasible solution—because of errors in the lexicon or linguistically unnatural word sequences not anticipated by the grammar designers. In this case it would be unclear how to relax our constraints.

Thus, rather than considering all theoretically possible word sequences, we chose to use a sample of *naturally occurring* sentences parsed by the link grammar, and to seek a direction mapping so that *these* parses would not violate our constraints after directions were imposed. If no such mapping exists, then we are willing to orient a few edge tokens in the wrong direction to ensure that the parses are still well-formed—but we minimize the number of such violations. In this way, the empirical distribution of sentences guides our assignment of directions. We are releasing the resulting multi-headed directed corpus via our personal websites.

4 Data Sets

We used the English sentences from the CoNLL 2007 Shared Task [21]—a subset of the Penn Treebank for which *single*-headed reference parses are available. We also used a prefix of the Russian News Commentary data from the ACL 2013 Shared Task of Machine Translation,[1] which is unparsed.

We generated link parses using the AbiWord/CMU link grammar parser version 5.0.8 [24]. The parser's coverage is less than ideal: we obtained connected parses for only 10,960 (of 18,577) English sentences and only 4,913 (of 18,577) Russian sentences, discarding the other sentences.[2] These two languages have the

[1] `http://www.statmt.org/wmt13/training-monolingual-nc-v8.tgz`

[2] When the link parser fails, it outputs a disconnected graph representing its best effort parse within a time limit. We removed these sentences for fear that the parses would be unreliable.

most mature lexicons at present, although lexicons for 7 other languages are available.

On English, the link grammar parses have 8% more edges overall, indicating that their directed versions will have a few multi-headed constructions per sentence. They do differ in style from the single-headed CoNLL parses of the same English sentences. Only 52% of the links match CoNLL arcs, and only 57% of the CoNLL arcs match links.

5 Integer Linear Programming Model

For each undirected labeled edge ij in the link corpus, where i, j denote tokens in the same sentence with $i < j$, we introduce nonnegative integer variables x_{ij} and x_{ji} with a constraint $x_{ij} + x_{ji} = 1$. We interpret $x_{ij} = 1$ or $x_{ji} = 1$ to mean that the link has direction $i \rightarrow j$ or $i \leftarrow j$, respectively.[3]

For each non-0 token v, we ensure that it has at least one parent by constraining[4]

$$\sum_u x_{uv} \geq 1 \tag{1}$$

where u ranges only over tokens such that the relevant variable exists. To prevent cycles,[5] for each token v we introduce a depth variable d_v in the range $[0, n_v]$ (not constrained to be integer), where n_v is the length of the sentence containing v. We require a child's depth to be at least 1 greater than each of its parents' depths—constraints that can be satisfied iff the sentence has no directed cycles:

$$(\forall u)\ d_v + (1 + n_v) \cdot (1 - x_{uv}) \geq 1 + d_u \tag{2}$$

The second summand ensures that (2) is trivially satisfied (hence has no effect) when u is *not* the parent of v.

Finally, we encourage all links with the same label to have the same direction. For each label L, we introduce binary variables r_L and ℓ_L, which say whether a link of type L is "allowed" to point right or left, respectively. For each undirected edge ij of label L, with $i < j$, we write

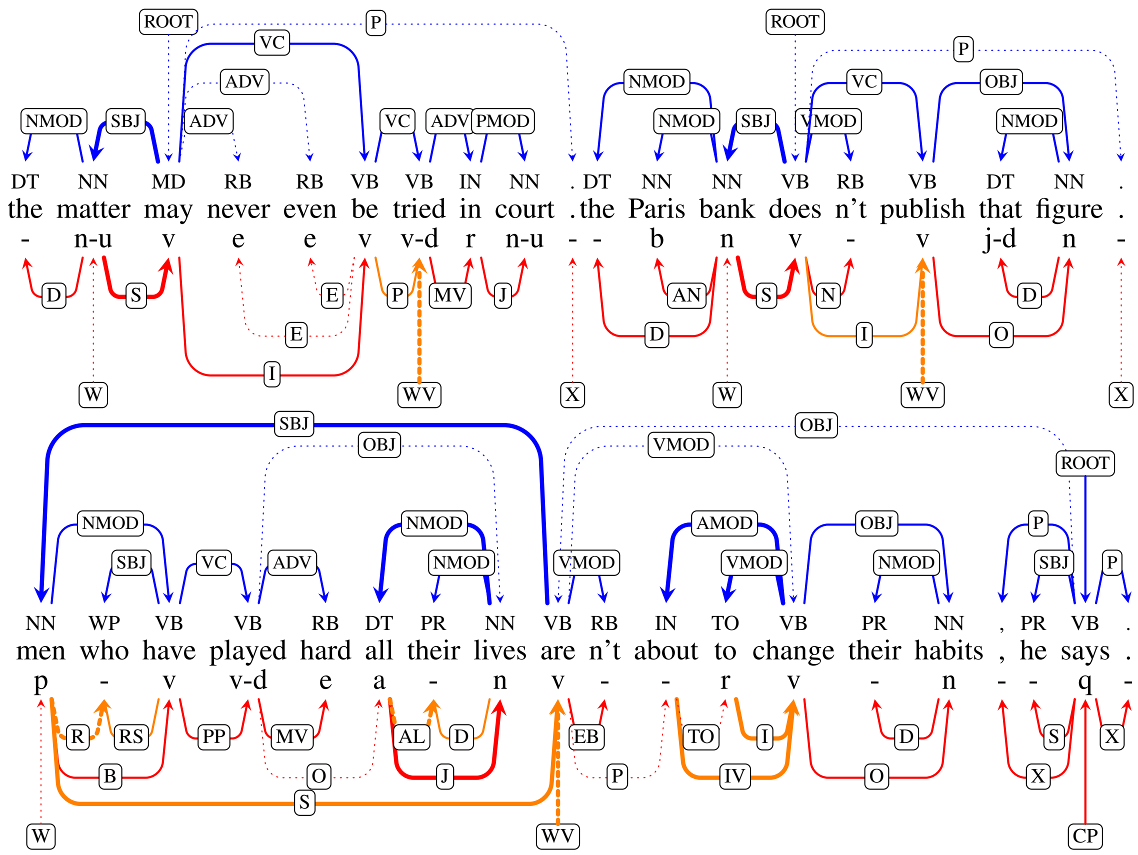
$$x_{ij} \leq r_L + s_{ij} \qquad\qquad x_{ji} \leq \ell_L + s_{ij} \tag{3}$$

where $s_{ij} \geq 0$ is a slack variable that allows an edge token to point in a disallowed direction if needed to ensure (1)–(2).

[3]In practice we halve the number of variables by replacing x_{ji} with $1 - x_{ij}$ for $j > i$, but that obscures the exposition.

[4]To denote two linked tokens, we use variables i, j when i is to the left of j, or variables u, v when u is the parent of v.

[5]This also ensures REACHABLE, given (1).



ROOT
P
VC
ADV
NMOD
SBJ
ADV
VC
ADV
PMOD
DT NN MD RB RB VB VB IN NN .
the matter may never even be tried in court .
- n-u v e e v v-d r n-u -
D
S
E
E
I
P
MV
J
W
WV
X
ROOT
P
NMOD
VC
OBJ
NMOD
SBJ
VMOD
NMOD
DT NN NN VB RB VB DT NN .
the Paris bank does n't publish that figure .
- b n v - v j-d n -
AN
D
S
N
I
D
O
W
WV
X
SBJ
OBJ
OBJ
VMOD
ROOT
NMOD
NMOD
AMOD
OBJ
P
SBJ
VC
ADV
NMOD
VMOD
VMOD
NMOD
SBJ
P
NN WP VB VB RB DT PR NN VB RB IN TO VB PR NN , PR VB .
men who have played hard all their lives are n't about to change their habits , he says .
p - v v-d e a - n v - - r v - n - - q -
R
RS
PP
MV
AL
D
EB
TO
I
D
S
X
B
O
J
P
IV
O
S
W
WV
CP

Figure 1: The blue upper edges show CoNLL gold dependency parses; the red lower edges show our oriented version of the link grammar parses. Edges are shown as dotted lines if they appear only in one parse. Edges are highlighted in orange if the child has multiple parents. Edges that appear in both parses are solid lines, drawn thicker if the directions do not match. Vertical edges have parent 0. For 100 example parses, see Appendix B of the supplementary material.

Our objective tries to minimize the number of allowed directions (by link type—cf. [26]) and the total slack (by link token):

$$\min \left(\sum_L r_L + \ell_L \right) \cdot \frac{N_L}{4} + \sum_{ij} s_{ij} \tag{4}$$

where N_L is the number of link tokens with label L. Objective (4) is willing to tolerate up to 1/4 of those link tokens' using a disallowed direction before it prefers to allow *both* directions. One could experiment with adjusting the constant 1/4; we selected that value simply because it seemed like a reasonable threshold *a priori*.

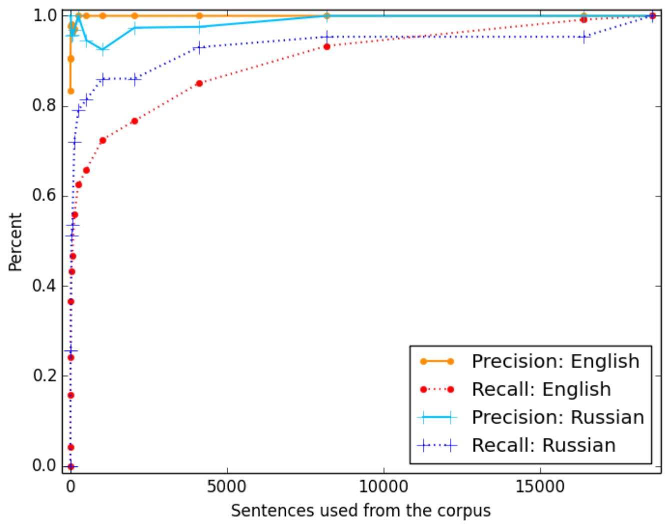


Figure 2: Rapid convergence to the direction mapping obtained on the largest dataset. The direction mappings obtained on small datasets have high *precision* relative to the one obtained on the largest dataset. Their *recall* grows as more link types are seen and directionalized.

6 Experiments and Results

We solved our ILP problem using the SCIP Optimization Suite [1], encoding it using the ZIMPL language [11]. Our largest run took 1.5 hours. On English, only 7 of 113 link *types* allowed both directions, and only $\sum_{ij} s_{ij} = 4043$ of 195000 link *tokens* required a disallowed direction via slack. 72.09% of the English sentences but (alas) only 0.04% of the Russian ones had at least one multi-headed word. See Table 1 and Appendix A for detailed results.

6.1 Stability of Results

We worried that the direction mapping might be unstable and sensitive to the input corpus. Happily, Figure 2 shows otherwise (for both English and Russian). Using even a small prefix of the data got very high-precision results, in the sense that nearly all r_L or ℓ_L variables that were 1 in this lightly trained mapping were also 1 in our largest run. The only disadvantage to using small data is low recall relative to the large run—many of the labels L are not observed yet and so we do not yet allow either direction ($r_L = \ell_L = 0$).

We used only coarse link tags as our labels, keeping only the capital letters of a tag (and merging all `ID` tags). This is because other characters in a tag indicate fine-grained features such as plurality that generally do not affect link direction. However, when we tried using fine tags as our labels instead, we found that all refinements of the same coarse tag would almost always spontaneously agree on their preferred direction. This indicates that there is indeed a “natural” direction

Label	Rightward	Multiheaded	CoNLL Match	CoNLL Dir Match	CoNLL Label
A	0% (0/8501)	0% (0/8501)	84% (7148/8501)	98% (7002/7148)	NMOD 98% (7000/7148)
AA	0% (0/4)	0% (0/4)	-	-	-
AF	84% (16/19)	37% (7/19)	32% (6/19)	0% (0/6)	VMOD 83% (5/6)
AJ	50% (131/262)	0% (0/262)	86% (225/262)	99% (223/225)	COORD 97% (218/225)
AL	100% (71/71)	99% (70/71)	-	-	-
AM	0% (0/45)	0% (0/45)	51% (23/45)	65% (15/23)	AMOD 65% (15/23)
AN	0% (0/9401)	0% (0/9401)	83% (7825/9401)	98% (7639/7825)	NMOD 96% (7523/7825)
AZ	100% (2/2)	0% (0/2)	100% (2/2)	100% (2/2)	ADV 100% (2/2)
B	100% (1514/1515)	61% (919/1515)	53% (806/1515)	84% (678/806)	NMOD 75% (603/806)
BI	100% (34/34)	0% (0/34)	38% (13/34)	100% (13/13)	VMOD 77% (10/13)
BW	100% (1/1)	100% (1/1)	100% (1/1)	0% (0/1)	OBJ 100% (1/1)
C	100% (3272/3272)	0% (0/3272)	3% (85/3272)	53% (45/85)	NMOD 27% (23/85)
CC	100% (176/176)	4% (7/176)	9% (16/176)	0% (0/16)	PRN 56% (9/16)
CO	0% (0/2478)	1% (32/2478)	5% (114/2478)	68% (78/114)	NMOD 39% (44/114)
CP	100% (283/283)	13% (36/283)	88% (249/283)	100% (249/249)	ROOT 100% (249/249)
CQ	100% (7/7)	0% (0/7)	100% (7/7)	0% (0/7)	VMOD 57% (4/7)
CV	100% (3237/3237)	100% (3237/3237)	56% (1827/3237)	28% (512/1827)	VMOD 52% (956/1827)
CX	100% (6/6)	0% (0/6)	83% (5/6)	20% (1/5)	VMOD 60% (3/5)
D	0% (56/19535)	0% (71/19535)	85% (16656/19535)	100% (16608/16656)	NMOD 100% (16629/16656)
DD	0% (0/629)	0% (3/629)	26% (165/629)	99% (164/165)	NMOD 99% (163/165)
DG	0% (0/1051)	0% (0/1051)	90% (950/1051)	100% (950/950)	NMOD 100% (948/950)
DP	0% (0/13)	0% (0/13)	23% (3/13)	100% (3/3)	SBJ 100% (3/3)
DT	0% (0/509)	0% (0/509)	100% (508/509)	99% (505/508)	NMOD 99% (505/508)
E	0% (0/1897)	0% (2/1897)	67% (1279/1897)	99% (1263/1279)	ADV 84% (1079/1279)
EA	1% (6/473)	2% (11/473)	83% (394/473)	96% (377/394)	AMOD 95% (376/394)

Table 1: Our solution, i.e., our reconstruction of the “intended” direction for each link type in the English Link Grammar. We also indicate the extent to which each of these link types (1) has a single dominant direction, (2) participates in multi-headed constructions, and (3) corresponds to CoNLL links of a predictable direction and type. For space reasons, we show only the start of this table—the full table can be found in Appendix A of the supplementary material.

for the coarse tag and that we can find it.

6.2 Linguistic Analysis

The resulting English corpus uses a syntactic annotation scheme that is somewhat different from the CoNLL annotations. Differences are tabulated in Appendix A of the supplementary material, while the actual parses are contrasted in Appendix B. Fragments of these appendices are shown in Table 1 and Figure 1.

The link grammar results in multi-headed treatments of infinitivals, compound determiners, relative clauses, and embedding. The other annotation differences are generally reasonable, e.g., letting *'s* be the head of a possessive, and different handling of punctuation and lists. One could easily modify the ILP to explicitly encourage agreement with the CoNLL link directions (for word pairs that are linked in CoNLL). Of course, a few of the differences are due to parser attachment errors.

The main vexation is the handling of subject-verb links. Under the English link grammar, the verb (or 0) that governs a clause will link to both the clause's subject and its last (main) verb. This would permit our desired treatment of *"Jill persuaded him to skip"*, in which *"him"* has two parents. But the ILP solution generally treats subjects as *parents* of verbs (thus we get *him* $\rightarrow$ *to*). The reason for this is an inconsistency in the link grammar itself.[6] Fixing the link grammar would presumably correct the link direction. As noted in section 1, it is arguably a positive result that our method was able to detect a problem in the grammar engineering.

7 Conclusions

We have presented an automatic ILP-based method to "orient" link grammar parses in multiple languages, turning them into rooted connected DAGs. This improves their linguistic interpretability and provides new corpora for experimenting with multi-headed dependency parsers.

ILP may *in general* be a valuable technology for enriching existing annotated corpora. For example, the Penn Treebank project [14] deliberately omitted types of annotations that plausibly could be added automatically. ILP can help by leveraging unsupervised corpus-wide information [26], enforcing annotations that are simultaneously well-formed per sentence and consistent across sentences.

References

[1] Tobias Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the Linguistic Annotation Workshop and Interoperability with Discourse*, 2013.

[6] Specifically, the link grammar is inconsistent about how it handles clauses with missing subjects (*"Jill wanted to skip"*). In this case, the governing verb links to the clause's first (tensed) verb in lieu of its subject. The problem is that it no longer also links to the clause's last (main) verb as it would if the subject were present. Hence our method concludes that a VP is always headed by its first verb (*to* $\rightarrow$ *skip*). That is a respectable convention on its own, but recall that unfortunately, the governing verb does *not* link to this first verb when the subject is present. As a result, the only remaining possible parent for the first verb is the subject (*him* $\rightarrow$ *to* $\rightarrow$ *skip*). This leads to the odd solution where subjects are treated as parents of verbs in general.

Note that subjects are not *always* parents, due to another inconsistency in the link grammar. In the construction *"It was impossible, they said"*, the subject (*"they"*) is inconsistently not linked to 0 but only to the verb (*"said"*) and so must be the verb's child, no other parent being available.

[3] Bernd Bohnet. Comparing advanced graph-based and transition-based dependency parsers. In *Proceedings of the First International Conference on Dependency Linguistics*, pages 282–289, 2011.

[4] Matthias Buch-Kromann. *Discontinuous Grammar. A Model of Human Parsing and Language*. Dr.ling.merc. dissertation, Copenhagen Business School, 2006.

[5] Wenliang Chen, Zhenghua Li, and Min Zhang. Dependency parsing: Past, present, and future. In *Proceedings of COLING 2014: Tutorial Abstracts*, pages 14–16, Dublin, August 2014. Tutorial slides available online.

[6] Marie-Catherine de Marneffe, Miriam Connor, Natalia Silveira, Samuel R. Bowman, Timothy Dozat, and Christopher D. Manning. More constructions, more genres: Extending Stanford dependencies. In *Proceedings of the Second International Conference on Dependency Linguistics*, pages 187–196, 2013.

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

[8] Jason Eisner. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, October 2000.

[9] Carlos Gómez-Rodríguez and Joakim Nivre. Divisible transition systems and multiplanar dependency parsing. *Computational Linguistics*, 39(4):799–845, 2013.

[10] Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the 6th Linguistic Annotation Workshop*, pages 2–11, Jeju, Korea, 2012.

[11] Thorsten Koch. *Rapid Mathematical Programming*. PhD thesis, Technische Universität Berlin, 2004. ZIB-Report 04-58.

[12] Matthias T. Kromann. The Danish Dependency Treebank and the underlying linguistic theory. In Joakim Nivre and Erhard Hinrichs, editors, *Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, 2003.

[13] Marco Kuhlmann and Joakim Nivre. Mildly non-projective dependency structures. In *Proceedings of COLING-ACL*, pages 507–514, Sydney, 2006.

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

[15] Andre Martins, Miguel Almeida, and Noah A. Smith. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of ACL*, pages 617–622, Sofia, Bulgaria, August 2013.

[16] André F. T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. Stacking dependency parsers. In *Proceedings of EMNLP*, pages 157–166, Honolulu, October 2008.

[17] Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic CFG with latent annotations. In *Proc. of ACL*, pages 75–82, Ann Arbor, Michigan, June 2005.

[18] Ryan McDonald and Fernando Pereira. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88, 2006.

[19] Joakim Nivre. Beyond MaltParser: Advances in transition-based dependency parsing. Available at `http://stp.lingfil.uu.se/~nivre/docs/BeyondMaltParser.pdf`. Slides from invited talks.

[20] Joakim Nivre. Non-projective dependency parsing in expected linear time. In *Proceedings of ACL-IJCNLP*, pages 351–359, Singapore, August 2009.

[21] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, 2007.

[22] Joakim Nivre and Jens Nilsson. Pseudo-projective dependency parsing. In *Proceedings of ACL*, pages 99–106, Ann Arbor, June 2005.

[23] Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 63–72, Dublin, August 2014.

[24] OpenCog Project. Link grammar parser. Available from `http://www.abisource.com/projects/link-grammar/`, April 2014.

[25] Emily Pitler, Sampath Kannan, and Mitchell Marcus. Finding optimal 1-endpoint-crossing trees. *Transactions of the Association for Computational Linguistics*, 1:13–24, 2013.

[26] Sujith Ravi and Kevin Knight. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of ACL-IJCNLP*, pages 504–512. Association for Computational Linguistics, 2009.

[27] Alexander Rush and Slav Petrov. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of NAACL*, pages 498–507, Montréal, Canada, June 2012.

[28] Kenji Sagae and Jun'ichi Tsujii. Shift-reduce dependency DAG parsing. In *Proceedings of COLING*, pages 753–760, Manchester, UK, August 2008.

[29] Daniel Sleator and Davy Temperley. Parsing English with a link grammar. Computer Science Technical Report CMU-CS-91-196, Carnegie Mellon University, October 1991.

[30] Davy Temperley. An introduction to the link grammar parser. Available at `http://www.link.cs.cmu.edu/link/dict/introduction.html`, March 1999.

[31] Linus Vepstas. Re: Warning: Combinatorial explosion. Message to the `link-grammar` Google Group. Available at `https://groups.google.com/forum/#!msg/link-grammar/eeJw1Ofgc9U/diqPYSwuFfoJ`, February 2014.