Quadratic-Time Dependency Parsing for Machine Translation

Gabriele Sarti, University of Trieste

Algorithmic Design

July 23, 2019

Table of Contents

What is a Context–Free Grammar?

Constituents → Group of words behaving as single units in sentence construction.

Context-Free Grammars → Formal system to model constituent structure, formed by:

- ❖ A set of **productions** that describes how symbols can be grouped together.
- ❖ A **lexicon** of words and symbols.

Used in linguistics and computer science for **language parsing**, difficult and costly to apply for many natural languages.

Lexicon:

Noun → *flight | Thursday*
Verb → *want | leave | do*
Pronoun → *I | me | you | it*
Prop Noun → *Los Angeles | Boston*
Determiner → *the | a | an | this | these*
Prep → *from | to | near | on*

Productions	**Examples**
S → NP VP	I + want a flight
NP → Pronoun	I
| (Prop) Noun	Los Angeles
| Det Noun	a + flight
VP → Verb	do
| Verb NP	want + a flight
| Verb PP	leave + on Thursday
| Verb NP PP	leave + Boston + on Thursday
PP → Prep NP	from + Los Angeles

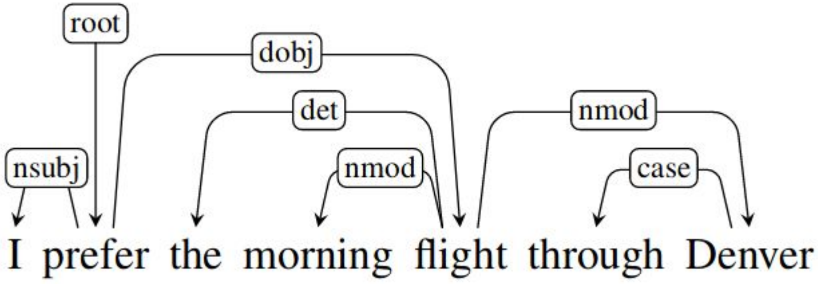
What is a Dependency Structure?

In **dependency grammars** we focus on **relations** among words instead of constituents.

Relations are **typed** and have a **head** and a **dependent.**

Abstracts away word order information, useful for **free word order** languages.

Models **semantic relations** between predicates and their arguments, used for many NLP tasks.



Relations	**Description**
NSUBJ	Nominal subject
DOBJ	Direct object
NMOD	Nominal modifier
DET	Determiner
CASE	Pre/postposition cases

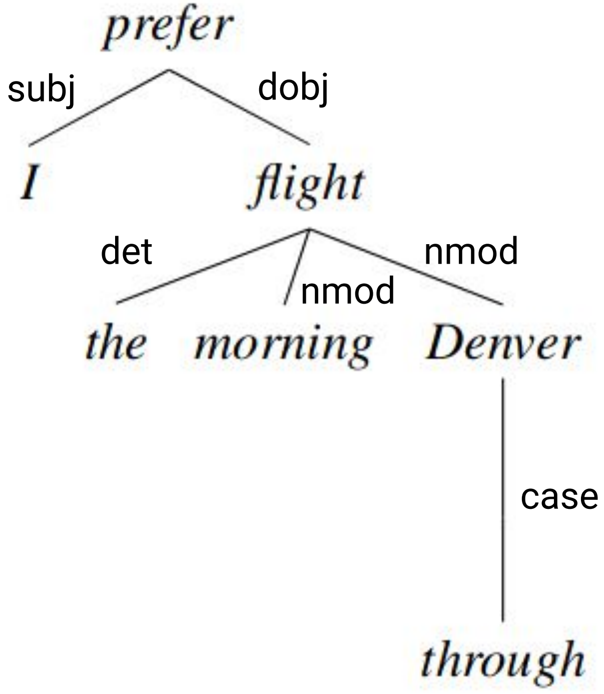
Dependency Structures as Trees

The dependency structures are **directed graphs** $G = (V, A)$, where V is a set of vertices corresponding to the words in the sentence and A is the set of arcs defining grammatical relations between words.

Computationally-motivated restrictions:

- **Single root** with no incoming arcs.
- **One incoming arc** per non-root vertex.
- **Unique path** from root to each vertex.

Thus, the structure becomes a **dependency tree** that can be used for efficient parsing.



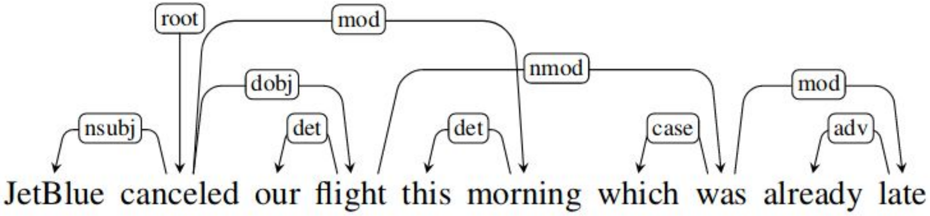
Projectivity

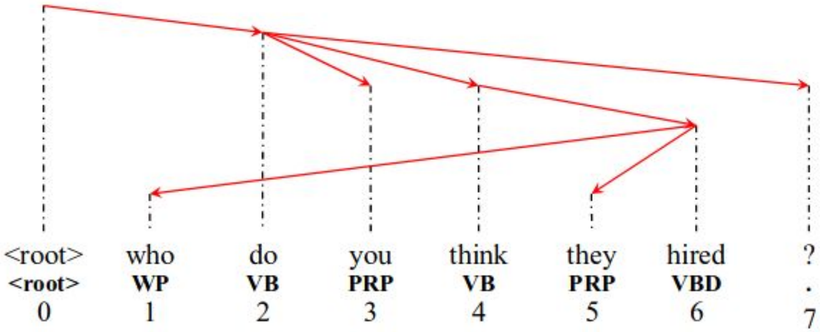
An arc A from a head h to a dependent d the is said to be **projective** if all words between h and d have a path that connects them to h.

A dependency tree is projective if all its arcs are projective, aka **no crossing edges**.

Projectivity makes parsing more costly and it is not desirable for many languages.

Despite enlarging the tree search space, **non-projectivity** yields comparable performances in parsing even in English.



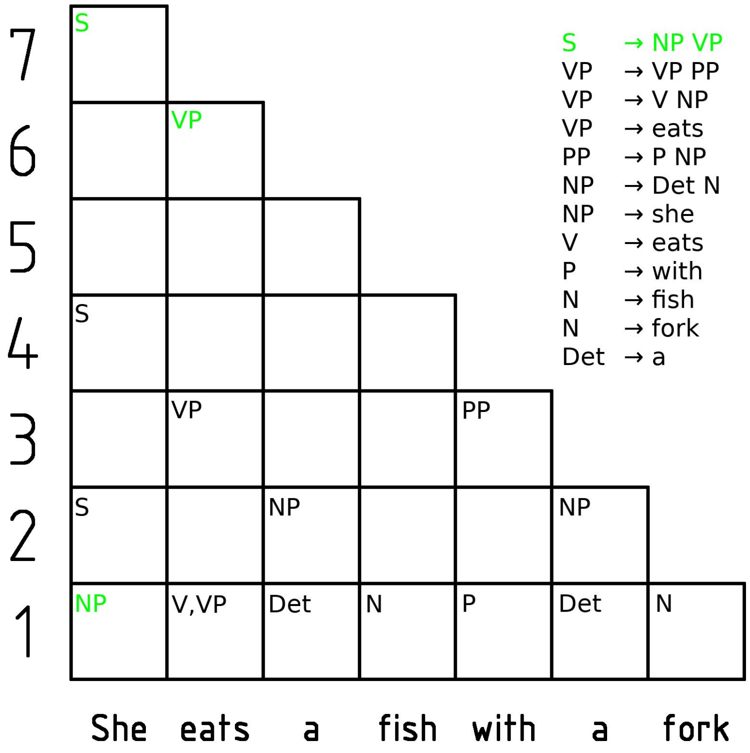


From Chart Parsing to Graph Dependency Parsing

Hierarchical approaches to **Machine Translation** (MT) employed the **Cocke-Younger-Kasami (CYK) algorithm** with high-order language models, very expensive computationally.

Complexity of CYK on a sentence of size n using m-grams is $O(n^{3+3(m-1)}) \simeq O(n^{3m})$ (Eisner and Satta, 1999).

Dependency parsing algorithms that run in time $O(n^3)$ (Eisner 1996) or even $O(n^2)$ (McDonald et al. 2005) can be used instead.



Dependency Parsing as a MST Problem

A **Maximum Spanning Tree (MST)** of a weighted graph G is an acyclic subgraph including all the vertices of G using the minimal possible number of edges (one entering edge per node) with the maximal possible total edge weight.

Given an input sentence, we build a **fully-connected, weighted directed graph** having words for vertices and all possible head-dependent relations as edges.

Weights represent the scores given by a classifier trained on treebanks (more later). A root node r connected to all vertices is added.

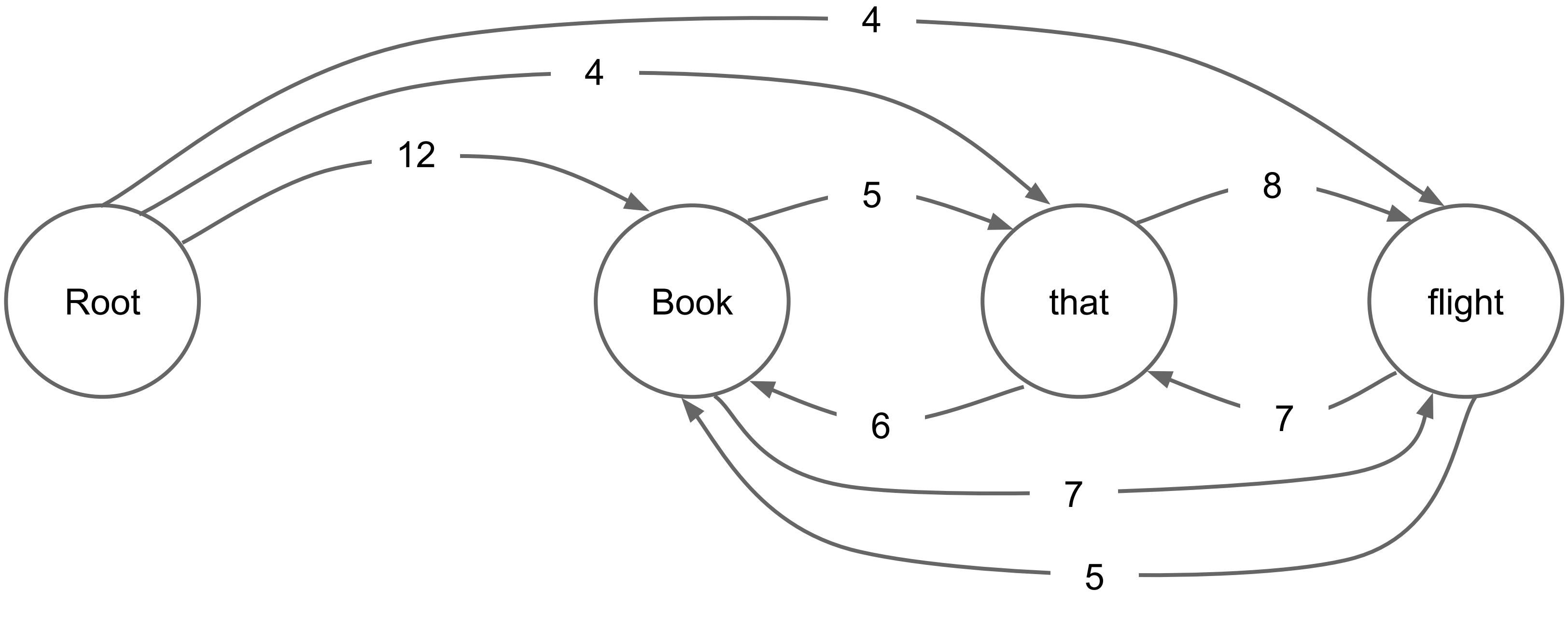
We use the **Chu-Liu Edmonds (CLE) algorithm** (Edmonds, 1967) to find the MST in a weighted directed graph. The MST rooted in r will be the optimal dependency parse selected by the algorithm.

CLE Algorithm Intuitively

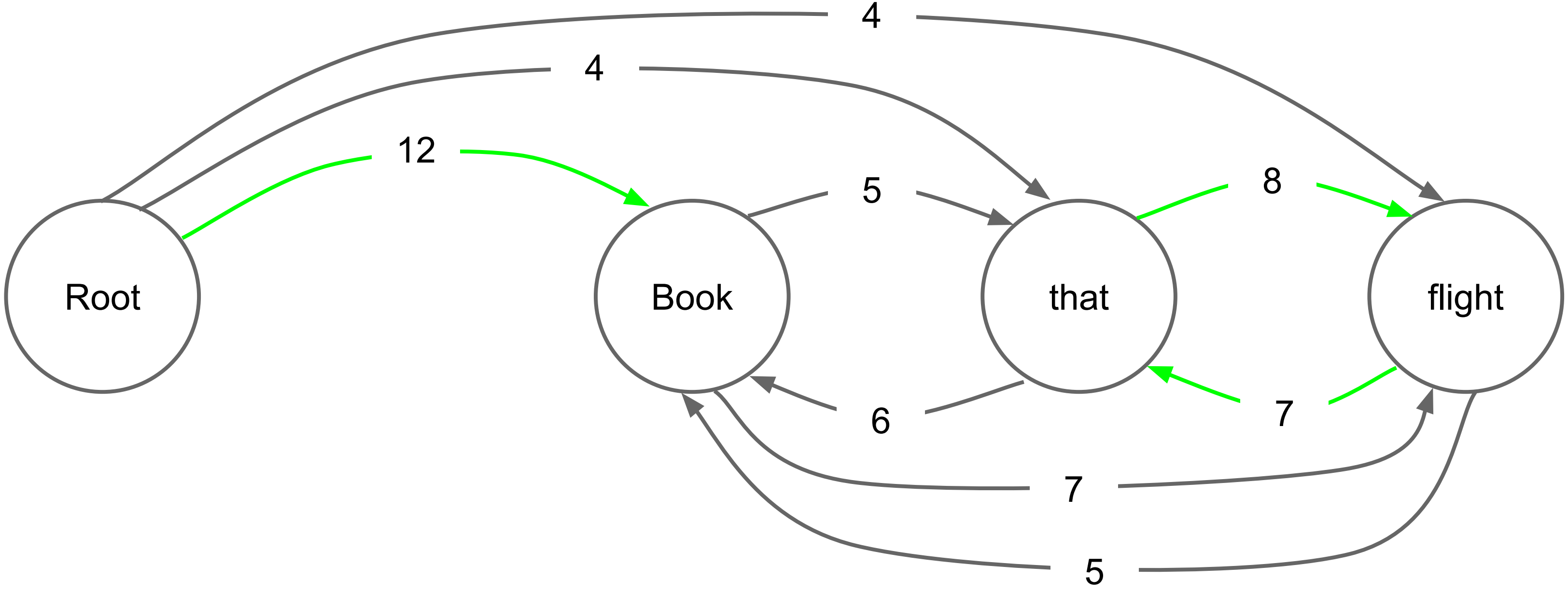
- For each vertex in the fully-connected weighted directed graph, greedily select the incoming edge with highest weight.
- If the result of the previous operation is a tree, it must be a MST. **Stop**.
- If not, there must be a cycle:
  - Identify and contract the cycle in a single vertex of a new contracted graph.
  - Recalculate edge weights coming in and out of the cycle
    - Only max outbound edge for each outer vertex is kept
    - Inbound edges are kept and scored as best spanning trees originating in outer vertices, including only inner vertices.
  - Apply CLE recursively.

It can be shown that the MST on the contracted graph resulting from CLE is equivalent to a MST on the original graph (Georgiadis, 2003).

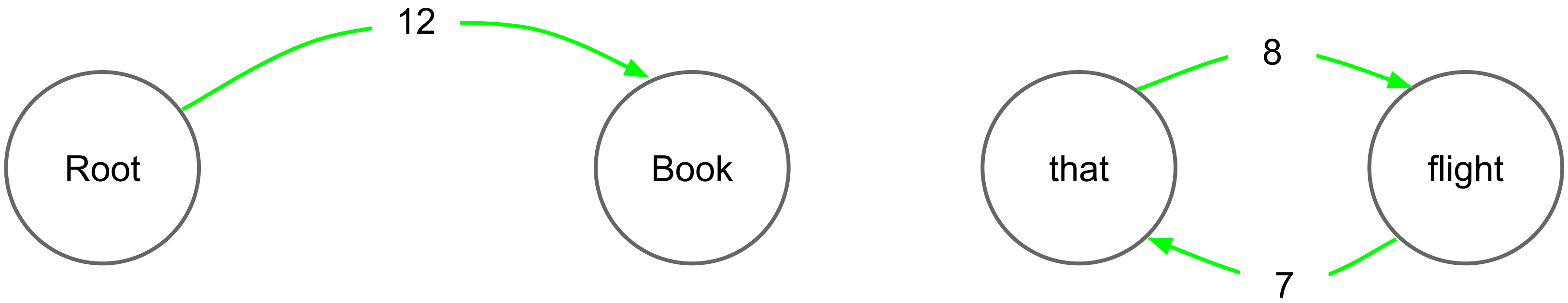
Visualizing the CLE Algorithm



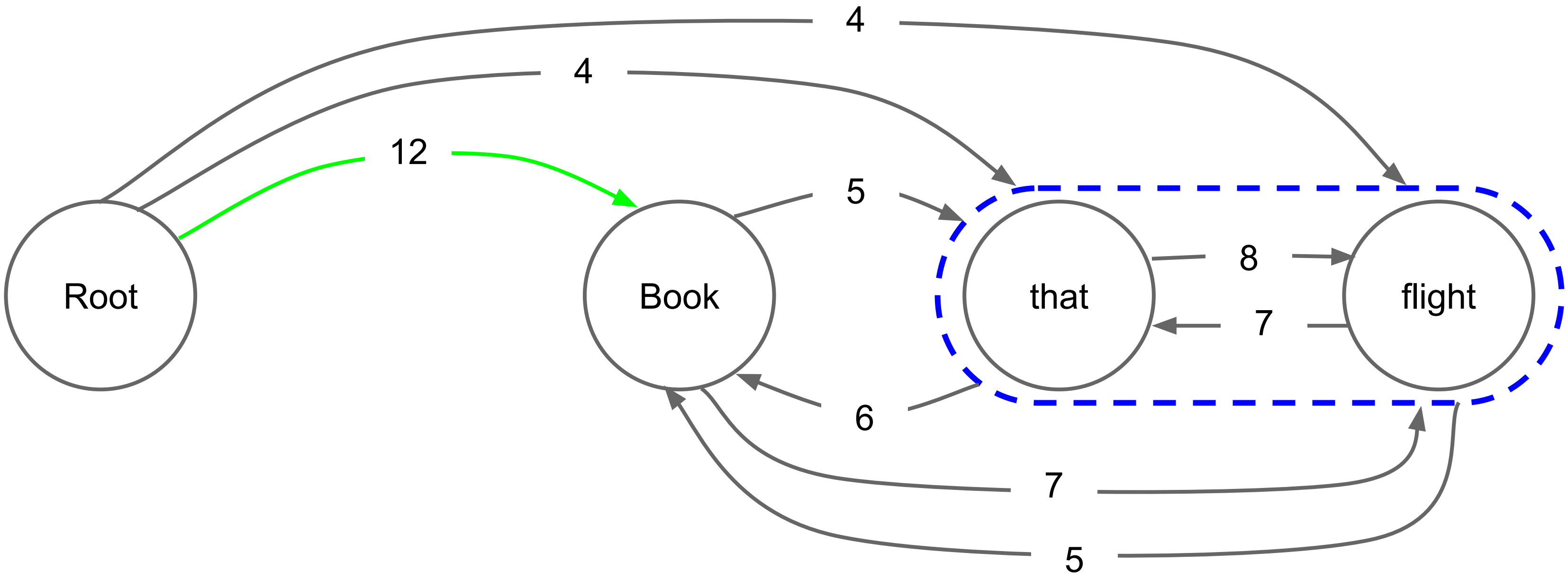
Greedy Selection of Maximum Inbound Edge



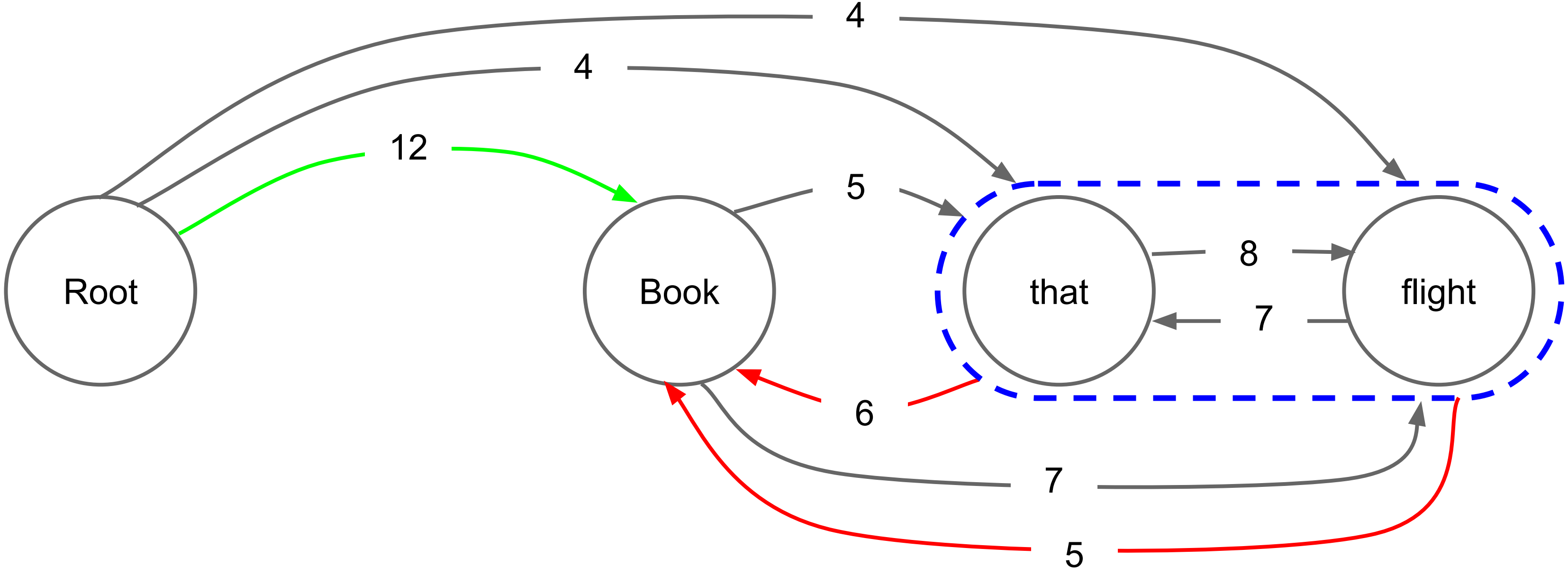
Not a Tree! Missing Connection and Cycle Presence



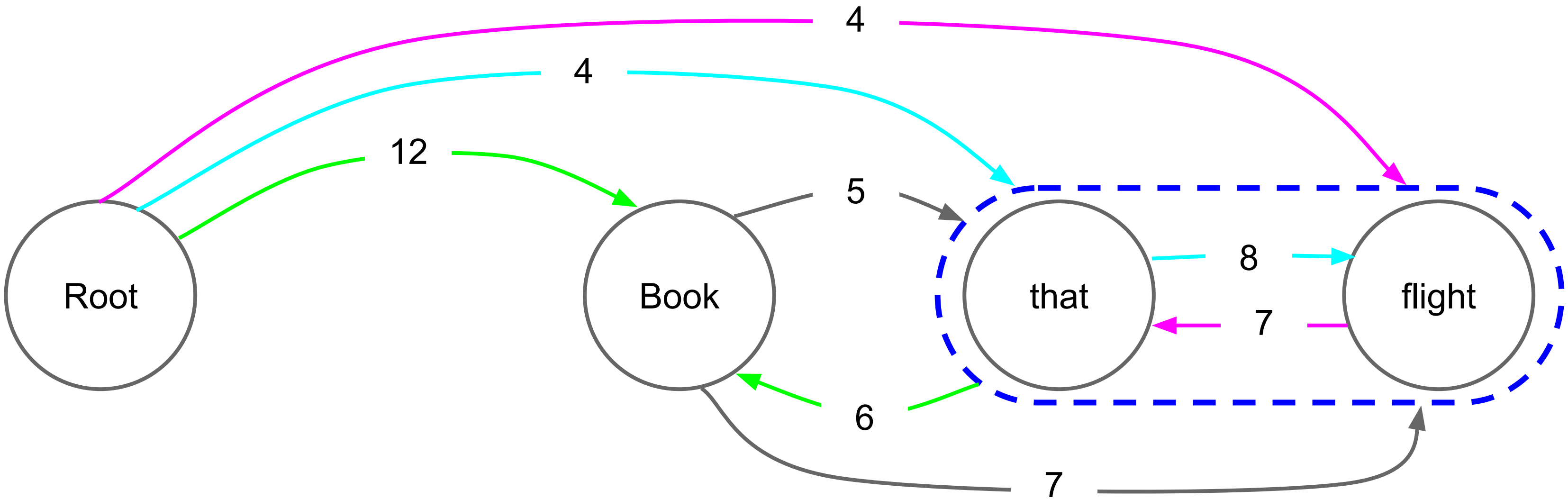
Contract the Cycle & Keep Unrelated Edges Weights



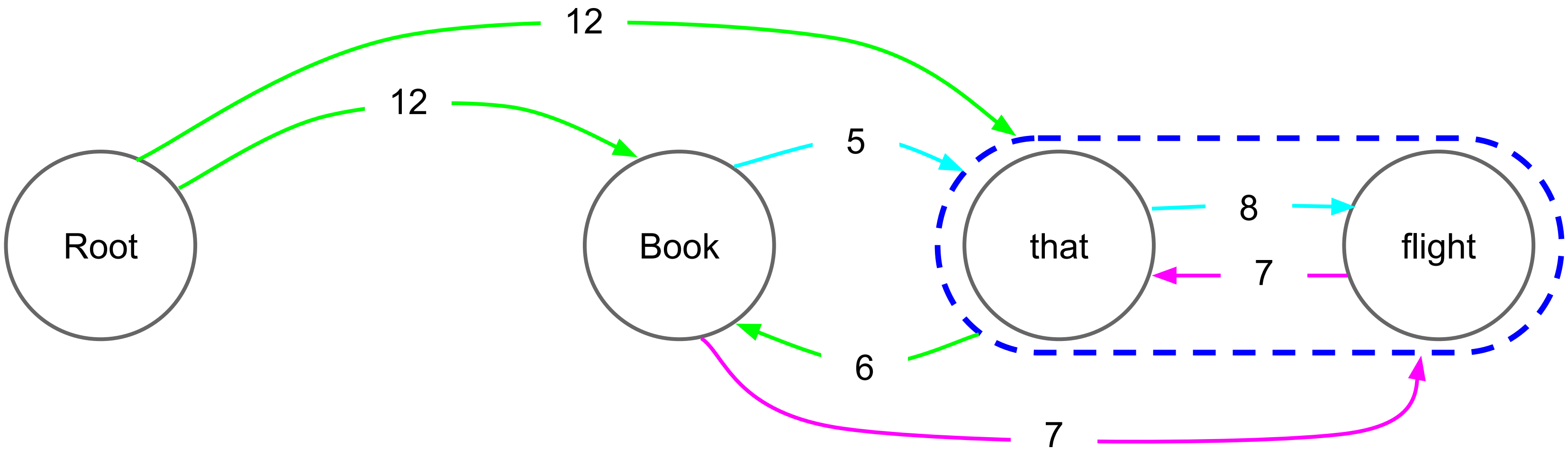
Keep Only Max Outbound Edge for each Vertex



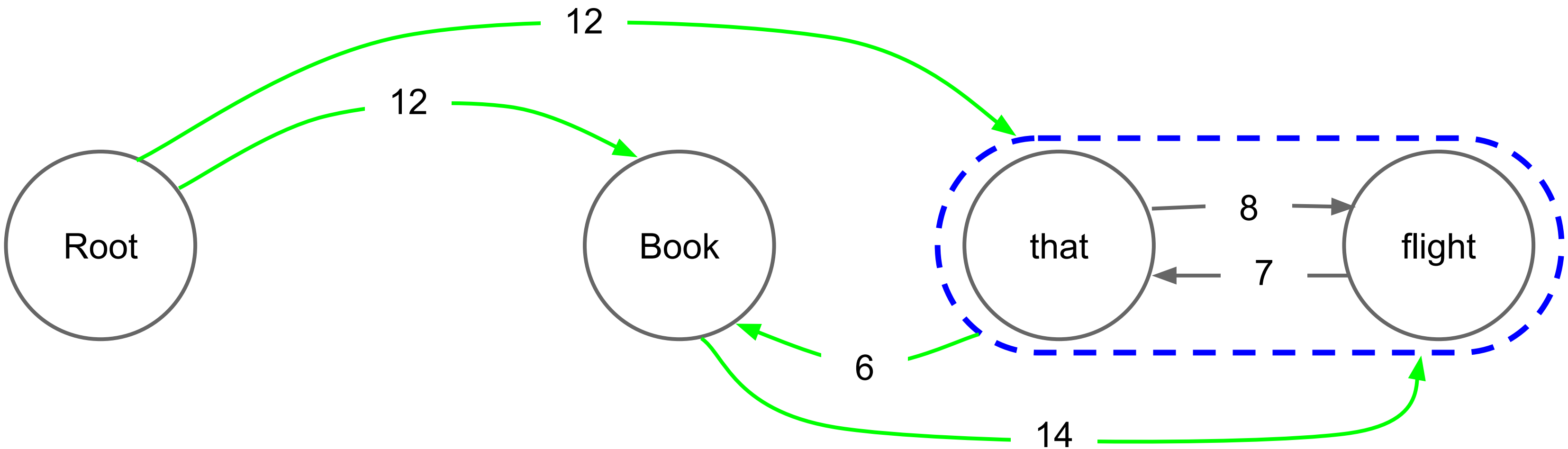
Rescoring and Pruning of Inbound Edges as MST



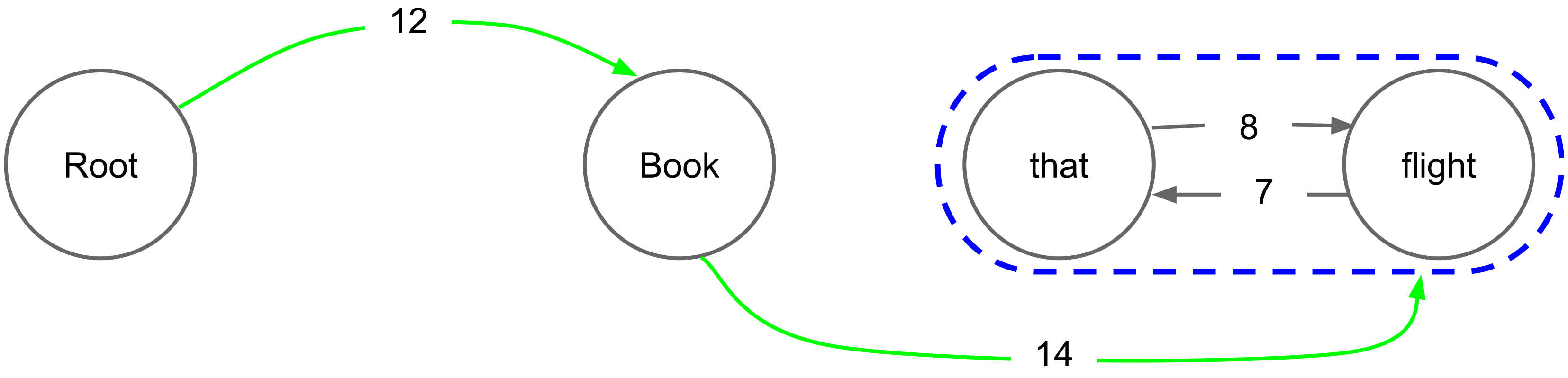
Rescoring and Pruning of Inbound Edges as MST



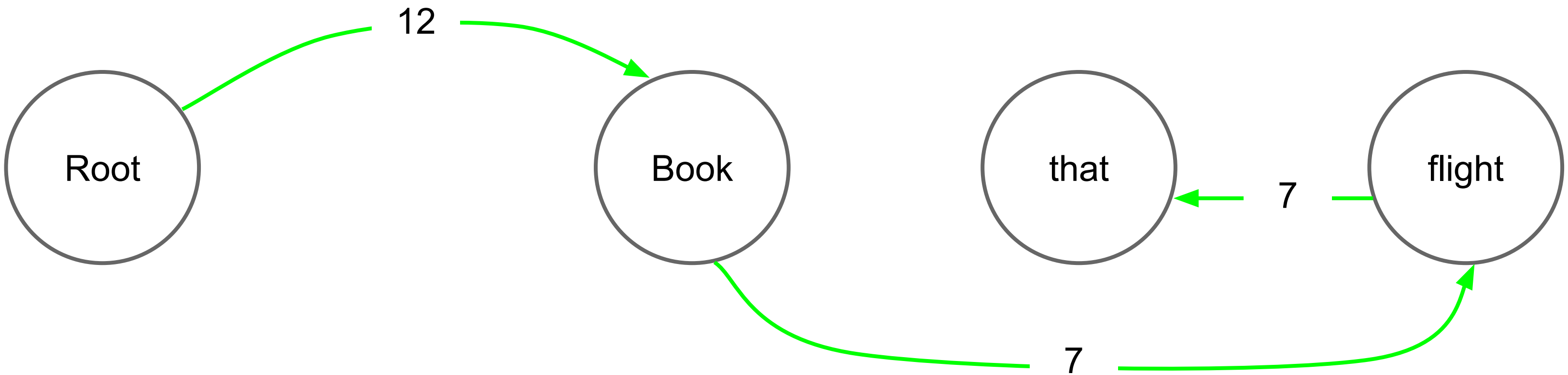
A New Contracted Graph is Created



Greedy Selection on Contracted Graph. It’s a MST!



Expand the Contracted Graph. We have our MST!



Chu Liu Edmonds Algorithm

- ❖ **`G(V,E)`** is the fully-connected weighted directed graph composed by vertices **`V`** and edges **`E`**.
- ❖ **`s`** is the adjacency matrix for the graph.
- ❖ Cycles are found using Tarjan SCC lowlink algorithm.
- ❖ Returns a MST for graph **`G`**

```
CLE(G=(V,E), root, s)

bestEdges ← []
newTree ← []
for each v in V do
     currBest ← ARGMAX(INCOMING_EDGES(v, s))
     INSERT(bestEdges,currBest)
if IS_SPANNING_TREE(T=(V,bestEdges)) then
     return T
else
     C ← FIND_CYCLE(bestEdges)
     newGraph ← CONTRACT(G,C,s)
     newTree ← CLE(newGraph,root,s)
     T ← EXPAND(newTree, C)
     return T
```

Algorithmic Complexity of CLE

CLE for dependency parsing deals with a fully connected graph with n vertices for words and $m = n^2$ edges.

Thus, naive complexity of CLE is $O(n^3)$

- ❖ At most n recursive calls, given at most n contractions.
- ❖ Each call takes $O(n^2)$ to find the highest incoming edge and contract the graph.

However, a modification adopted by (Tarjan, 1977) for dense graphs using ordered lists to store edges leads to a total complexity of $O(n^2)$.

Edge Scores: Features and Training

The score associated to an edge can be reduced to a weighted sum of features extracted from it:

$$score(S,e) = \sum_{i=1}^{N} w_i f_i(S,e)$$

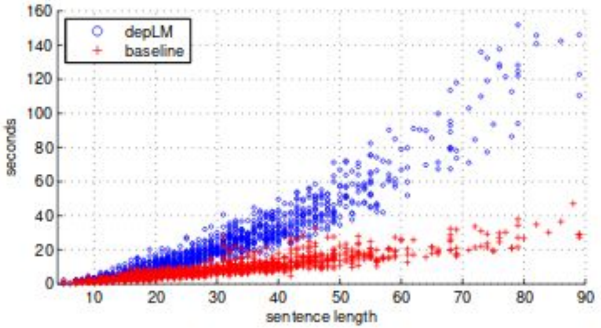
Features can go from POS tags to relation distances to word embeddings. In our case, we keep information concerning the **predicted POS tags** for each word and its adjacent words, conjoined to the **word itself**, a **dependency score** and a **translation score**. Recent methods features rely solely on embeddings (Zeman et al. 2017), making hand-crafted features obsolete.

Weights are inferred from a treebank training set by applying a learning algorithm. Our paper uses the **Margin Infused Relaxed Algorithm (MIRA)** (Crammer, Singer 2003)

Machine Translation Experiments

- The **Moses** phrase-based decoder was used with standard features to test **Chinese-to-English** translation.
- 28 million English words & 23.3 million Chinese words from news parallel corpora, plus a smoothed 5-gram LM using Gigaword corpus ~ **700 million words**.
- Used NIST MT evaluation data for Chinese, tested on BLUE and TER.
- Tested with and without the **dependency language model score** as additional feature.

	BLEU[%]			
DEP. LM	newswire	web	speech	all
no	32.86	21.75	36.88	32.29
yes	33.19 (+0.33)	22.64 (+0.89)	37.51 (+0.63)	32.74 (+0.45)
	TER[%]			
DEP. LM	newswire	web	speech	all
no	57.73	62.64	55.16	58.02
yes	56.73 (−1)	61.97 (−0.67)	54.26 (−0.9)	57.10 (−0.92)
	newswire	web	speech	all
Sentences	4006	1149	1451	6606



Results and Discussion

- Without CLE, performance is similar (not relevant for LM)
- Performance is only slightly worse than SOTA implementation.
- Performance loss worth the scaling gain for larger LM.
- For MT, outperforms 5-gram LM with significant scores, though a bit slower.

ALGORITHM	TIME	SETUP	TRAINING	TESTING	ACCURACY
Projective	$O(n^3)$	Parsing	WSJ(02-21)	WSJ(23)	90.60
Chu-Liu-Edmonds	$O(n^3)$	Parsing	WSJ(02-21)	WSJ(23)	89.64
Chu-Liu-Edmonds	$O(n^2)$	Parsing	WSJ(02-21)	WSJ(23)	89.32
Local classifier	$O(n^2)$	Parsing	WSJ(02-21)	WSJ(23)	89.15
Projective	$O(n^3)$	MT	CTB(050-325)	CTB(001-049)	86.33
Chu-Liu-Edmonds	$O(n^3)$	MT	CTB(050-325)	CTB(001-049)	85.68
Chu-Liu-Edmonds	$O(n^2)$	MT	CTB(050-325)	CTB(001-049)	85.43
Local classifier	$O(n^2)$	MT	CTB(050-325)	CTB(001-049)	85.22
Projective	$O(n^3)$	MT	CTB(050-325), WSJ(02-21), ATB, OntoNotes	CTB(001-049)	87.40(**)
Chu-Liu-Edmonds	$O(n^3)$	MT	CTB(050-325), WSJ(02-21), ATB, OntoNotes	CTB(001-049)	86.79
Chu-Liu-Edmonds	$O(n^2)$	MT	CTB(050-325), WSJ(02-21), ATB, OntoNotes	CTB(001-049)	86.45(*)
Local classifier	$O(n^2)$	MT	CTB(050-325), WSJ(02-21), ATB, OntoNotes	CTB(001-049)	86.29

	BLEU[%]					
DEP. LM	MT05 (tune)	MT02	MT03	MT04	MT06	MT08
no	33.42	33.38	33.13	36.21	32.16	24.83
yes	34.19 (+.77**)	33.85 (+.47)	33.73 (+.6*)	36.67 (+.46*)	32.84 (+.68**)	24.91 (+.08)
	TER[%]					
DEP. LM	MT05 (tune)	MT02	MT03	MT04	MT06	MT08
no	57.41	58.07	57.32	56.09	57.24	61.96
yes	56.27 (−1.14**)	57.15 (−.92**)	56.09 (−1.23**)	55.30 (−.79**)	56.05 (−1.19**)	61.41 (−.55*)
	MT05 (tune)	MT02	MT03	MT04	MT06	MT08
Sentences	1082	878	919	1788	1664	1357

Thanks for the attention!

Gabriele Sarti, University of Trieste

gabriele.sarti996@gmail.com

DATA SCIENCE &
SCIENTIFIC COMPUTING

References

Main paper

- Galley & Manning "Quadratic-Time Dependency Parsing for Machine Translation", Proc. of ACL 2009.

Other sources, in appearance order:

- Primer on linguistics formalisms was inspired by Jurafsky & Martin "Speech and Language Processing" 3rd edition draft 2018, Chapters 10-13. Examples were mainly taken and adapted from there.
- Chapter 2 of Nguyen Bach "Dependency Structures for Statistical Machine Translation" PhD Thesis @ CMU, 2012 for an overview of the field of SMT, with a focus on dependency structures.
- Moses website and its phrase-based tutorial.

References

Other papers consulted and referred inside the presentation, in appearance order:

- Eisner & Satta "Efficient parsing for bilexical context-free grammars and head-automaton grammars", Proc. of ACL 1999.
- Eisner "Three new probabilistic models for dependency parsing: An exploration", Proc. of COLING 1996.
- McDonald & al. "Online large-margin training of dependency parsers", Proc. of ACL 2005a.
- McDonald & al. "Non-projective dependency parsing using spanning tree algorithms", Proc. of EMNLP 2005b.
- Edmonds "Optimum branchings", Research of the National Bureau of Standards, 1967.
- Crammer & Singer "Ultraconservative online algorithms for multiclass problems", JMLR 2003
- Georgiadis "Arborescence optimization problems solved by Edmonds' algorithm", Theoretical CS, 2003.
- Tarjan "Finding Optimum Branchings", Networks, 1977.
- Zeman et al. "CoNLL shared task: Multilingual parsing from raw text to universal dependencies", Proc. of CoNLL 2017
- Papineni et al. "BLEU: a method for automatic evaluation of machine translation"