Data-Driven Deep Dependency Parsing

Weiwei Sun

Institute of Computer Science & Technology
Peking University

August 8, 2017

Outline

The Covert Helps Parse the Overt

Semantic Dependency Parsing

Question

HPSG	PET, Enju, ACE, ...
CCG	C&C, ...
LFG	XLE, ...

PCFG	Collins, Charniark&Johnson, Berkeley, ...
Data-driven	MST, Mate, Malt, SyntaxNet, Stanford, ZPar, RNNG, ...

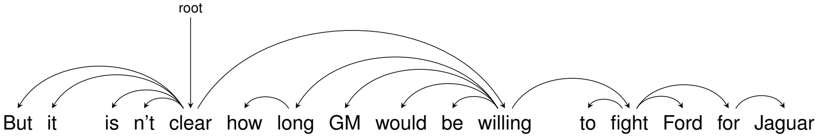
Question

HPSG	PET, Enju, ACE, ...
CCG	C&C, ...
LFG	XLE, ...

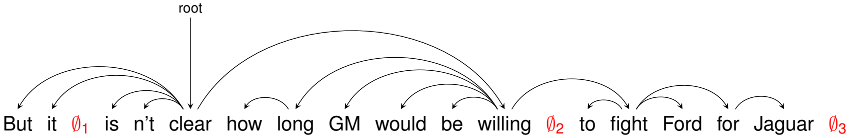
$\boxed{\text{Can deep syntactic information help surface parsing?}}$

PCFG	Collins, Charniark&Johnson, Berkeley, ...
Data-driven	MST, Mate, Malt, SyntaxNet, Stanford, ZPar, RNNG, ...

Syntactic analysis with empty categories

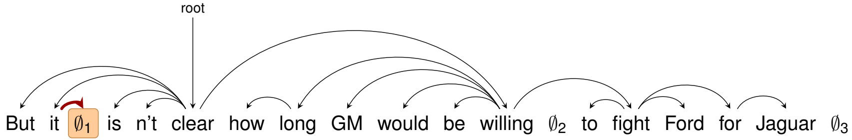


Syntactic analysis with empty categories



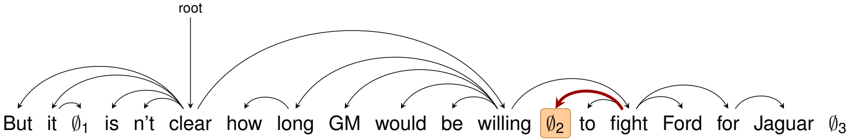
- $\emptyset_1$: expletive construction.
- $\emptyset_2$: the subject of *fight* is somehow missing because it is *controled* by the subject of *willing*.
- $\emptyset_3$: *wh*-movement in which an adjunct of *willing*, i.e. *how long* is moved to the front of the clause.

Syntactic analysis with empty categories



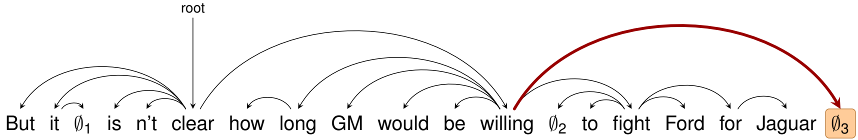
- $\emptyset_1$: expletive construction.
- $\emptyset_2$: the subject of *fight* is somehow missing because it is *controled* by the subject of *willing*.
- $\emptyset_3$: *wh*-movement in which an adjunct of *willing*, i.e. *how long* is moved to the front of the clause.

Syntactic analysis with empty categories



- $\emptyset_1$: expletive construction.
- $\emptyset_2$: the subject of *fight* is somehow missing because it is *controled* by the subject of *willing*.
- $\emptyset_3$: *wh*-movement in which an adjunct of *willing*, i.e. *how long* is moved to the front of the clause.

Syntactic analysis with empty categories



- $\emptyset_1$: expletive construction.
- $\emptyset_2$: the subject of *fight* is somehow missing because it is *controled* by the subject of *willing*.
- $\emptyset_3$: *wh*-movement in which an adjunct of *willing*, i.e. *how long* is moved to the front of the clause.

Parsing with empty elements

But it is n't clear how long GM would be willing to fight Ford for Jaguar

Task

- Predicting empty elements
- Predicting dependencies, including dependencies between normal and empty elements

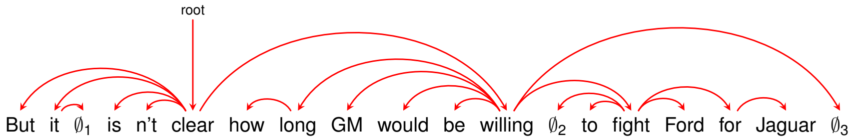
Parsing with empty elements

But it $\emptyset_1$ is n't clear how long GM would be willing $\emptyset_2$ to fight Ford for Jaguar $\emptyset_3$

Task

- Predicting empty elements
- Predicting dependencies, including dependencies between normal and empty elements

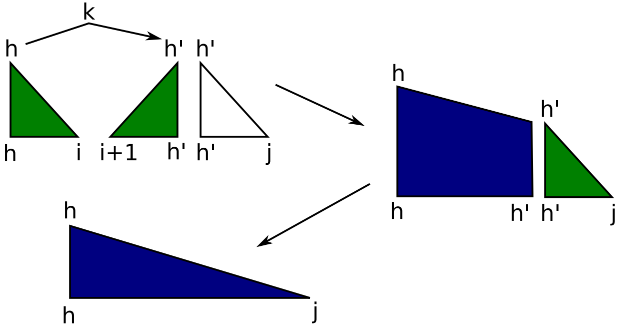
Parsing with empty elements



Task

- Predicting empty elements
- Predicting dependencies, including dependencies between normal and empty elements

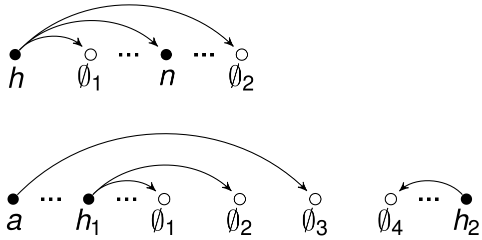
Parsing without empty elements



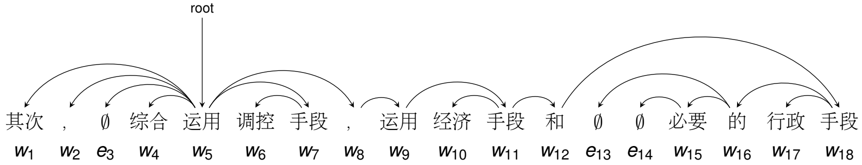
Prototypes of structures with empty categories

Assumption

Empty nodes can be only dependents.

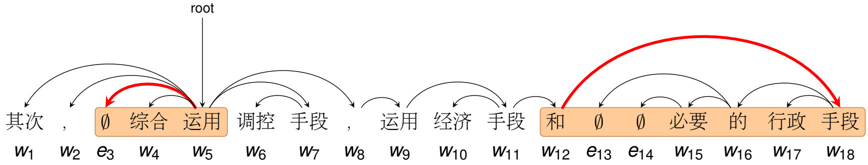


Parsing with empty elements

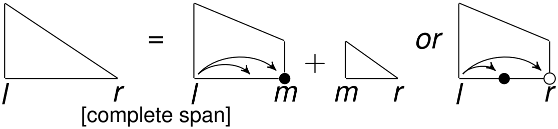


incomplete spans: a dependency and the region between the head and modifier.

Parsing with empty elements

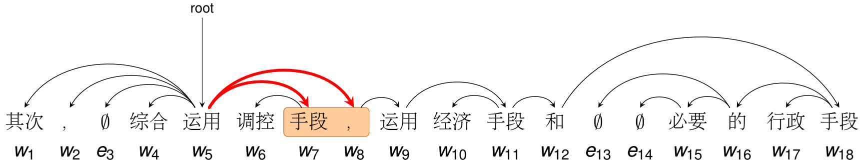


complete spans: a head-word and its descendents on one side

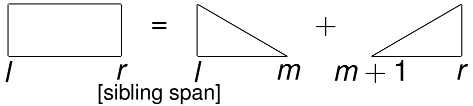


incomplete spans: a dependency and the region between the head and modifier.

Parsing with empty elements

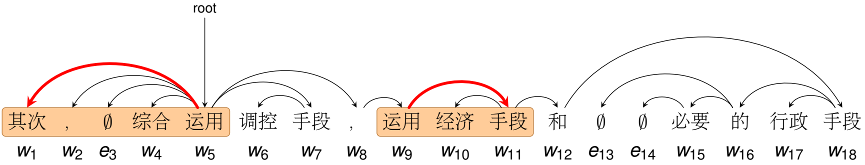


sibling span: the region between successive modifiers of same head.

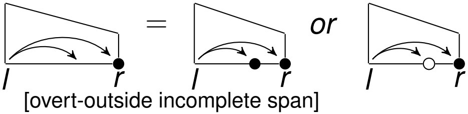


incomplete spans: a dependency and the region between the head and modifier.

Parsing with empty elements

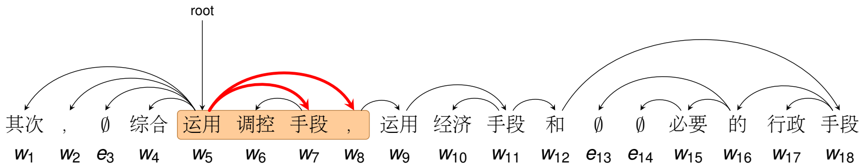


overt-outside incomplete span

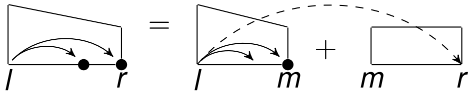


incomplete spans: a dependency and the region between the head and modifier.

Parsing with empty elements



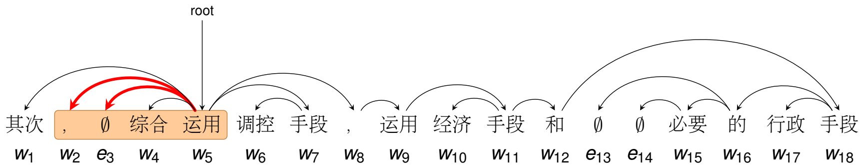
overt-both incomplete span



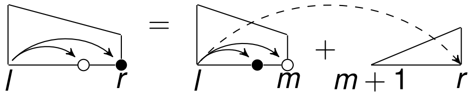
[overt-both incomplete span]

incomplete spans: a dependency and the region between the head and modifier.

Parsing with empty elements



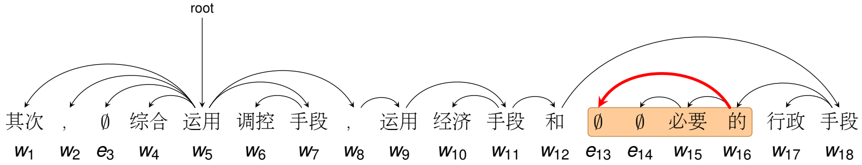
covert-inside incomplete span



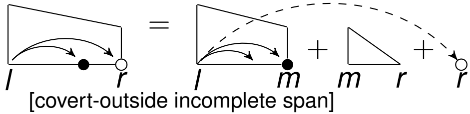
[covert-inside incomplete span]

incomplete spans: a dependency and the region between the head and modifier.

Parsing with empty elements



covert-ouside incomplete span



incomplete spans: a dependency and the region between the head and modifier.

Evaluation

Disambiguation: Global linear model

$$f(\boldsymbol{y}) = \sum_{p \in \boldsymbol{y}} \mathbf{w}_f^\top \phi_f(s, p)$$

Results: unlabeled attachment score for all overt words

Model	English	Chinese
second-order	91.73	89.16
$+\emptyset$ (partial)	91.70 (-0.03)	89.20 ($+0.04$)
$+\emptyset$ (full)	91.72 (-0.01)	89.28 ($+0.12$)
third-order	92.23	90.00
$+\emptyset$ (partial)	92.41 ($+0.18$)	89.82 (-0.18)

Evaluation

Disambiguation: Global linear model

$$f(\boldsymbol{y}) = \sum_{p \in \boldsymbol{y}} \mathbf{w}_f^\top \phi_f(s, p)$$

Results: unlabeled attachment score for all overt words

Model	English	Chinese
second-order	91.73	89.16
$+\emptyset$ (partial)	91.70 (-0.03)	89.20 ($+0.04$)
$+\emptyset$ (full)	91.72 (-0.01)	89.28 ($+0.12$)
third-order	92.23	90.00
$+\emptyset$ (partial)	92.41 ($+0.18$)	89.82 (-0.18)

Analysis

Two types of errors

- Approximation error
- Estimation error

Information about empty categories is helpful for reducing the approximation error, but brings new challenge for estimation.

Structure Regularization with joint decoding

$$\begin{array}{ll} \text{max.} & \lambda f(\boldsymbol{y}) + (1-\lambda) g(\boldsymbol{z}) \\ \text{s.t.} & \boldsymbol{y} \in \mathcal{Y}, \boldsymbol{z} \in \mathcal{Z} \\ & y(i,j) = z(i,j), \ \forall (i,j) \in \mathcal{I} \end{array}$$

Results

	Algo	English	Chinese
CM	1+3	91.94 (+0.21)	89.53 (+0.37)
	1+4	91.88 (+0.15)	89.44 (+0.28)
DD	1+3	91.96 (+0.23)	89.53 (+0.37)
	1+4	91.94 (+0.21)	89.53 (+0.37)
CM	2+5	92.60 (+0.37)	90.35 (+0.35)
DD	2+5	92.71 (+0.48)	90.38 (+0.38)

Two joint decoders

- **CM** Chart merging
- **DD** Dual decomposition

Results

	Algo	English	Chinese
CM	1+3	91.94 (+0.21)	89.53 (+0.37)
	1+4	91.88 (+0.15)	89.44 (+0.28)
DD	1+3	91.96 (+0.23)	89.53 (+0.37)
	1+4	91.94 (+0.21)	89.53 (+0.37)
CM	2+5	92.60 (+0.37)	90.35 (+0.35)
DD	2+5	92.71 (+0.48)	90.38 (+0.38)

Two joint decoders

CM Chart merging

DD Dual decomposition

Outline

The Covert Helps Parse the Overt

Semantic Dependency Parsing

Yesterday

$\boxed{\text{They use our semantics, but don't use our grammar.}}$

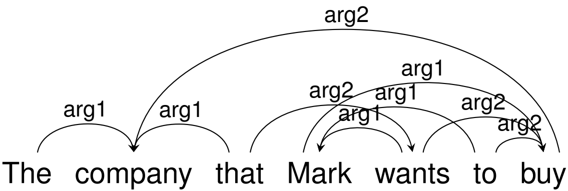
Yesterday

$\boxed{\text{They use our semantics, but don't use our grammar.}}$

$\boxed{\text{Grammar as an annotator.}}$

Semantic dependency parsing

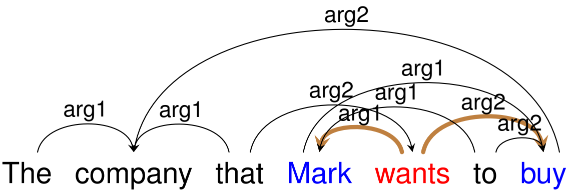
Example



- Predicate–argument analysis, bi-lexical relations
- Long-distance dependencies
- Graph-structured representations, many crossing arcs
- Not a tree: single-headed (✗), cycle-free (✗)

Semantic dependency parsing

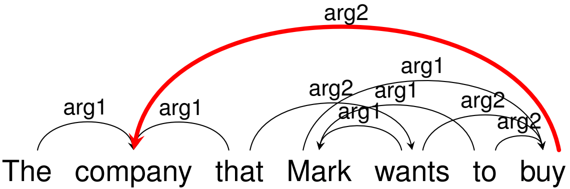
Example



- Predicate–argument analysis, bi-lexical relations
- Long-distance dependencies
- Graph-structured representations, many crossing arcs
- Not a tree: single-headed (✗), cycle-free (✗)

Semantic dependency parsing

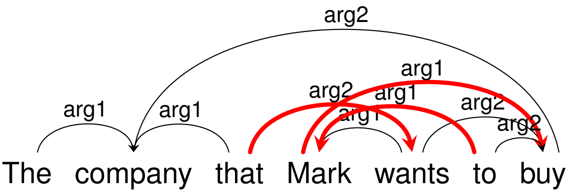
Example



- Predicate–argument analysis, bi-lexical relations
- Long-distance dependencies
- Graph-structured representations, many crossing arcs
- Not a tree: single-headed (✗), cycle-free (✗)

Semantic dependency parsing

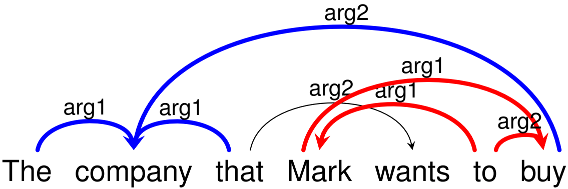
Example



- Predicate–argument analysis, bi-lexical relations
- Long-distance dependencies
- Graph-structured representations, many crossing arcs
- Not a tree: single-headed (✗), cycle-free (✗)

Semantic dependency parsing

Example



- Predicate–argument analysis, bi-lexical relations
- Long-distance dependencies
- Graph-structured representations, many crossing arcs
- Not a tree: single-headed (✗), cycle-free (✗)

Parsing approaches

Approaches

- Maximum Subgraph Parsing
- Transition-based Parsing
- Graph Merging

Maximum Subgraph

Input A directed graph $G = (V, A)$

Output Subgraph $G' = (V, A' \subseteq A)$ with maximum total weight such that G' belongs to $\mathcal{G}$

$$G'(s) = \arg\max_{H \in \mathcal{G}(s,G)} \sum_{p \in H} \textsc{ScorePart}(s, p)$$

Example When $\mathcal{G}$ is tree,

Maximum Subgraph = Maximum Spanning Tree

Complexity $\mathcal{G}$ and the order of $\textsc{ScorePart}$ determine the complexity of inference.

Complexity

$\mathcal{G}$	O	Algo	
Arbitrary	1	$O(n^2)$	
Arbitrary	2	NP-hard	ACL15
Acyclic	1	NP-hard	Kuhlmann & Jonsson
Noncrossing	1	$O(n^3)$	Kuhlmann & Jonsson
Noncrossing	2	$O(n^4)$	ACL17a
1-endpoint-crossing	1	$O(n^5)$	Ongoing work
1-endpoint-crossing pagenumber-2	1	$O(n^5)$	ACL17b
1-endpoint-crossing pagenumber-2, C-free	1	$O(n^4)$	ACL17b
1-endpoint-crossing pagenumber-2, C-free	2	$O(n^4)$	EMNLP17

Transition-based parsing

- Psycholinguistically motivated: Left-to-right, word-by-word
- Partially parsed results (parsing states) constrain parsing of subsequent words
- Usually, perform greedy search to get a *good* parse.

New transition systems

A naive idea

$\textsc{Parse}(x = (w_1, \ldots, w_n))$
1 **for** $j = 1..n$
2 **for** $k = j - 1..1$
3 $\texttt{Link}(j, k)$

New transition systems

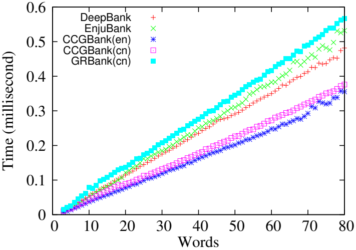
$\mathcal{G}$	System
Arbitrary graphs	Two-stack-based
Arbitrary graphs	Non-incremental online reordering
Supersets of noncrossing graphs	Incremental K-permutation

Real running time

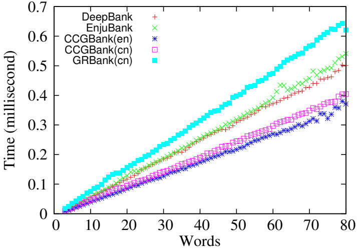
Naive spanning runs in time of $\Theta(n^2)$

$$
\begin{array}{l}
\textbf{for } j = 1..n \\
\quad \textbf{for } k = j-1..1 \\
\quad\quad \texttt{Link}(j,k)
\end{array}
$$

New systems



Online re-ordering



Two stack based

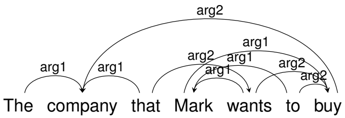
A new framework

Challenge of graph parsing

- Complex graphs are difficult to construct for its complex structure.
- Simple graphs can be solved more easily, but the coverage is not satisfactory.

Graph merging

Constructing a complex structure via constructing simple partial structures.



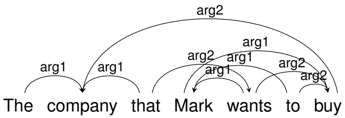
A new framework

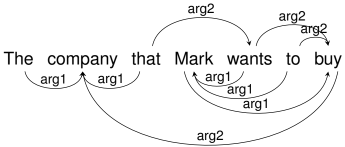
Challenge of graph parsing

- Complex graphs are difficult to construct for its complex structure.
- Simple graphs can be solved more easily, but the coverage is not satisfactory.

Graph merging

Constructing a complex structure via constructing simple partial structures.





Workflow

Architecture

	Annotated graphs		**Output**
	↓		↑
Graph Transformation	**Graph-to-Subgraphs** (Graph Decomposition)		**Subgraphs-to-Graph** (Graph Merging)
	↓		↑
Parser	**Parameter Estimation**	→	**Disambiguation/Decoding**
			↑
			Input

Workflow

Architecture

	Annotated graphs ↓		**Output** ↑
Graph Transformation	**Graph-to-Subgraphs**		**Subgraphs-to-Graph**
	(Graph Decomposition) ↓		(Graph Merging) ↑
Parser	**Parameter Estimation**	→	**Disambiguation/Decoding**
			↑ **Input**

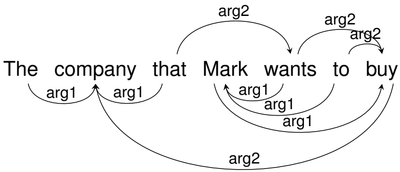
Three problems

Training How to decompose a complex graph into noncrossing graphs?

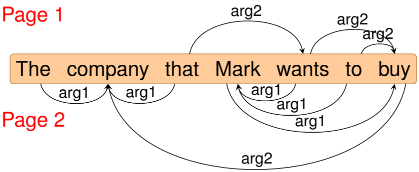
Parsing How to construct simple graphs?

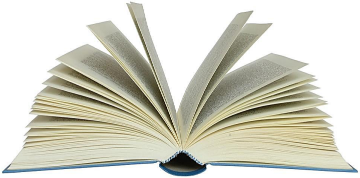
Parsing How to merge subgraphs into a coherent complex graph?

Book embedding



Book embedding



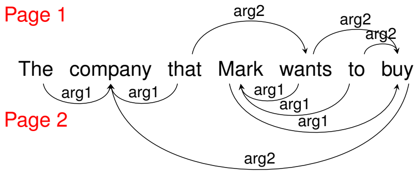


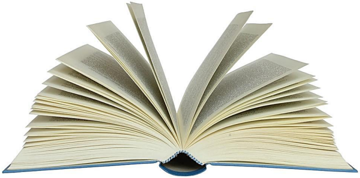
book embedding

A book embedding B of G satisfies the following conditions.

1. Every vertex of G is depicted as a point on the spine of B.
2. Every edge of G is depicted as a curve that lies within a single page of B.
3. Every page of B does not have any edge crossings.

Book embedding



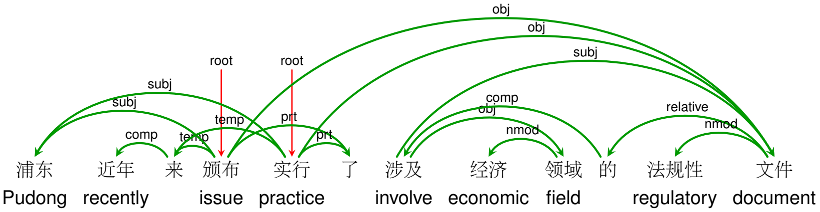


book embedding

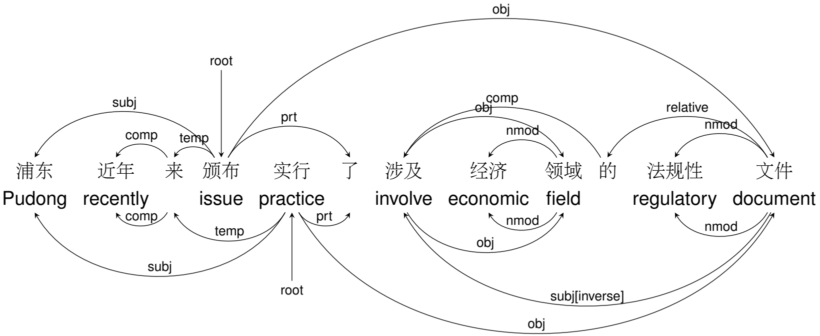
A book embedding B of G satisfies the following conditions.

1. Every vertex of G is depicted as a point on the spine of B.
2. Every edge of G is depicted as a curve that lies within a single page of B.
3. Every page of B does not have any edge crossings.

Tree + Tree + ...


obj
obj
subj
root
root
subj
subj
comp
obj
relative
nmod
temp
prt
comp
temp
prt
nmod
浦东 Pudong
近年 recently
来
颁布 issue
实行 practice
了
涉及 involve
经济 economic
领域 field
的
法规性 regulatory
文件 document

Tree + Tree + ...


obj
root
subj
comp
temp
prt
comp
obj
nmod
relative
nmod
浦东 近年 来 颁布 实行 了 涉及 经济 领域 的 法规性 文件
Pudong recently issue practice involve economic field regulatory document
comp
temp
prt
nmod
nmod
obj
subj
root
subj[inverse]
obj

Decomposing and combining subgraphs

Decomposition as Optimization

$$
\begin{array}{ll}
\text{max.} & \sum_k s_k(\boldsymbol{y}_k) \\
\text{s.t.} & \boldsymbol{y}_k \text{ belongs to } \mathcal{G}_k \\
 & \sum_k \boldsymbol{y}_k(i,j) \geq \boldsymbol{y}(i,j), \forall i,j
\end{array}
$$

Combination as Optimization

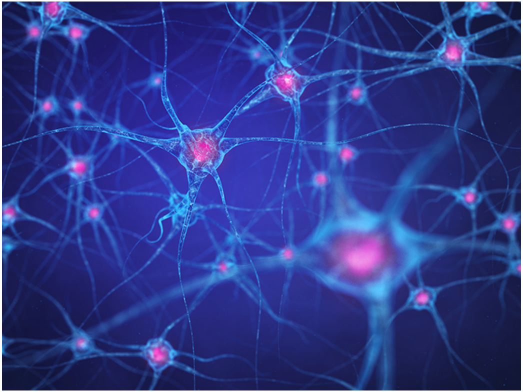
$$
\begin{array}{ll}
\text{min.} & -f_A(\mathbf{g}_A) - f_B(\mathbf{g}_B) \\
\text{s.t.} & \mathbf{g}_A \text{ belongs to } \mathcal{G}_A, \mathbf{g}_B \text{ belongs to } \mathcal{G}_B \\
 & A\mathbf{g}_A + B\mathbf{g}_B \leq 0
\end{array}
$$

Usually, we can employ Lagrangian Relaxation for solutions.

Lessons learned

Data-driven models can produce
high-quality deep dependency analysis.

Another experience



Game Over