Combining Dynamic Programming and Integer Linear Programming for Dependency Parsing

Alexis Nasr - Ghasem Mirroshandel

Laboratoire d'Informatique Fondamentale de Marseille

Motivations

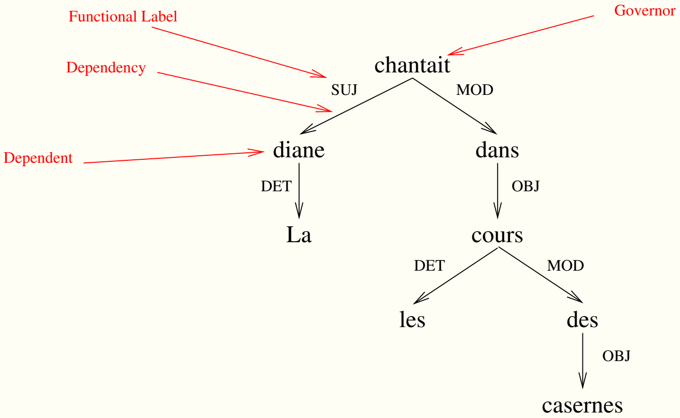
- Statistical parsers are trained on few thousand sentences manually annotated with syntax
- Some syntactico-lexical phenomena are incorrectly modeled, due to treebanks size
  - Jean regarde un homme avec un télescope
  - manger une glace à la fraise
  - commander une glace à la fraise
  - commander une glace à la serveuse
  - système de communication rapide
- We will never have enough annotated data for such phenomena !

Semi-supervised learning

- But : we have a lot of raw text
- General Idea :
  - Train a parser on a treebank
  - Parse a large amout of raw text
  - Select interesting sub-parses
  - Integrate this new data in the parser

Outline

Dependency Tree



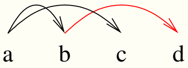
La diane chantait dans les cours des casernes
The trumpet blew in the yards of the barracks

CONLL Format

1	La	le	ART	2	DET
2	diane	diane	N	3	SUJ
3	chantait	chanter	V	0	ROOT
4	dans	dans	PREP	3	MOD
5	les	le	ART	6	DET
6	cour	cour	N	4	OBJ
7	des	de	PREP	6	MOD
8	casernes	caserne	N	7	OBJ

Projectivity

- Property defined on ordered trees
- A dependent cannot be separated from its governor in the string by a word that is not a descendent of the governor



- Linguistically reasonable
- Important reduction of the search space

nodes nb	3	4	5	6	7	8	9	10
trees	9	64	625	7776	117649	2.10^6	43.10^6	1.10^9
proj. trees	7	30	143	728	3876	21318	1200001	690690
ratio	1	2	4	11	30	98	358	1448

Graph-based parsing

- No rewriting grammar
- For a given sentence $S = m_1 \dots m_n$ and a functional labels tagset $\mathcal{F}$ any dependency tree T for S is a possible syntactic analysis of S.
- One structural constraint : projectivity
- Score of a tree :

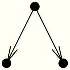
$$s(T) = \sum_{\psi \in \psi(T)} s(\psi)$$

- $\psi(T)$ is the set of all *relevant parts* of T
- $s(\psi)$ is the score of ψ

Decomposition

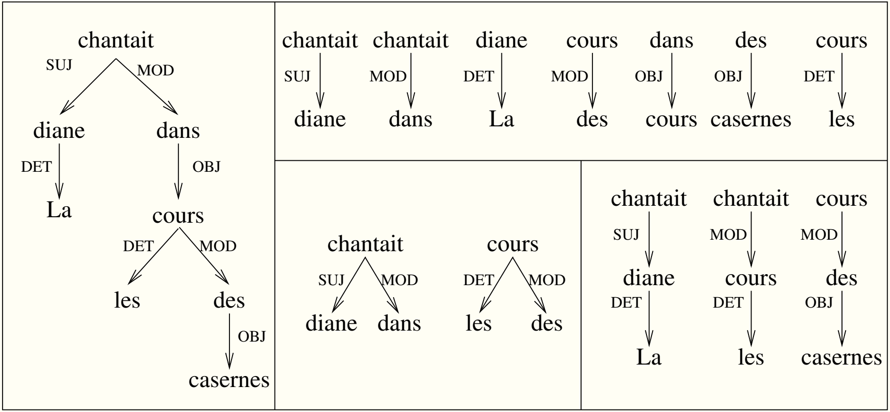
- First order models : a relevant part is just a dependency

- Second order models : a relevant part is one of



or

Example


chantait
SUJ
MOD
diane
dans
DET
OBJ
La
cours
DET
MOD
les
des
OBJ
casernes
chantait SUJ diane
chantait MOD dans
diane DET La
cours MOD des
dans OBJ cours
des OBJ casernes
cours DET les
chantait SUJ diane MOD dans
cours DET les MOD des
chantait SUJ diane DET La
chantait MOD cours DET les
cours MOD des OBJ casernes

Score of a part

- A part is decomposed as a vector f of elementary features.

(diane, diane, N)	-DET->	(la, le, ART)
(X, X, N)	-DET->	(X ,X , ART)
(diane, X, N)	-DET->	(X ,X , ART)
(X, X, N)	-DET->	(la ,X , ART)
(X, diane, N)	-DET->	(X ,le , ART)
...		

- A score w is computed for every feature
- The score of the part is the dot product $\mathsf{f} \cdot w$
- Weights are computed with an on-line machine learning algorithm (perceptron, MIRA)

Decoding

$$\hat{T} = \arg\max_{T \in \mathcal{T}(S)} \sum_{X \in \psi(T)} s(X)$$

$\mathcal{T}(S)$ is the set of all projective trees for sentence S

- Enumerate all projective trees for a sentence
- Compute the score of each tree
- Select the tree with highest score
- Dynamic Programming : $O(n^3)$

Constrained Parsing

- Force the parser to produce a solution that contains dependency $\delta = (g, r, d)$
- Define a new weight function s_δ^+ based on s

$$s_\delta^+(g', r', d') = \begin{cases} -\infty & \text{if } d' = d \text{ and} \\ & (g' \neq g \text{ or } r' \neq r) \\ s(g', r', d') & \text{otherwise} \end{cases}$$

Constrained Parsing

Force the parser to produce a solution that contains dependency set Δ

$$s^{+}_{\Delta}(g, r, d) = \begin{cases} s^{+}_{\delta}(g, r, d) & \text{if } (\cdot, \cdot, d) \in \Delta \\ s(g, r, d) & \text{otherwise} \end{cases}$$

Simple Confidence Measure

- k best parses of a sentence
- subtree Δ present in at least one of the k best parses
- $\mathcal{C}(\Delta)$: number of occurrences of Δ in the k trees
- $\mathrm{CM}(\Delta)$: confidence measure associated to Δ

$$\mathrm{CM}(\Delta) = \frac{\mathcal{C}(\Delta)}{\mathrm{k}}$$

Results on French data

French Treebank

	sent. nb.	tokens nb.
TRAIN	9 881	278 083
DEV	1 239	36 508
TEST	1 235	36 340

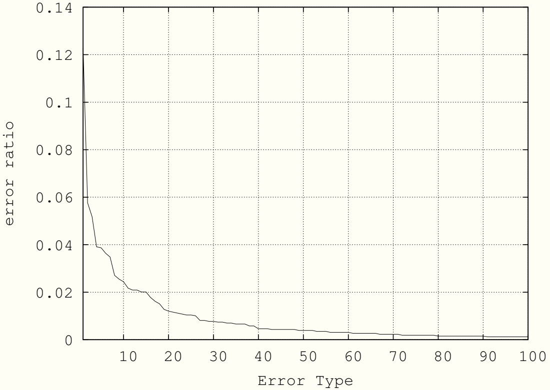
Accuracy

	TEST	DEV
LAS	88.88	88.53
UAS	90.71	90.37

UAS ratio of words with correct governor

LAS ratio of words with correct governor and correct syntactic function

Errors type distribution



Most frequent errors on DEV

dependency	freq.	acc.	contrib.	
N→N	1.50	72.23	2.91	
V → à	0.88	69.11	2.53	•
V—suj → N	3.43	93.03	2.53	•
N → CC	0.77	69.78	2.05	
N → de	3.70	92.07	2.05	
V → de	0.66	74.68	1.62	•
V—obj → N	2.74	90.43	1.60	•
V → en	0.66	81.20	1.24	
V → pour	0.46	67.78	1.10	
N → ADJ	6.18	96.60	0.96	
N → à	0.29	70.64	0.72	
N → pour	0.12	38.64	0.67	
N → en	0.15	47.69	0.57	

Lexico-Syntactic Configurations (LSC)

- Pair (s, T)
  - s is a score
  - T is a dependency tree of arbitrary size
- Nodes of T are 5-tuples $\langle f, p, w, l, i \rangle$
  - f functional label
  - p part of speech tag
  - l lemma
  - w word
  - i index (position in a sentence)

Example

```
(0.028, [:V:offrir::](
          [SUJ:N:::],
          [OBJ:N:fleur:fleurs:],
          [AOBJ:PREP:a:a:](
            [OBJ:N:::])))
```

- fields can be unspecified in LSC
- indices are always unspecified in LSC

Instantiated Lexico-Syntactic Configurations (ILSC)

- result of instantiating an LSC on a sentence
- preceding LSC instantiates on sentence
  Jean offre des fleurs à Marie
  as

```
(0.028, [:V:offrir:offre:2](
           [SUJ:N:Jean:Jean:1],
           [OBJ:N:fleur:fleurs:4],
           [AOBJ:PREP:a:a:5](
             [OBJ:N:Marie:Marie:6])))
```

Patching = Selecting the Optimal Set of ILSC

- Given a sentence S, and a set $\mathcal{L}$ of LSC
- $\mathcal{I}$ is the set of all possible instantiations of elements of $\mathcal{L}$ on S.
- $\hat{\mathcal{I}} \subseteq \mathcal{I}$ is the set of *compatible* ILSC such that

$$\hat{\mathcal{I}} = \arg\max_{\mathcal{I}' \subseteq \mathcal{I}} \sum_{x \in \mathcal{I}'} s(x)$$

- $s(x)$ is the score of ILSC x
- Cannot be done by brute force
- Integer Linear Programming

Subcat Frames (SF)

- Special kind of LSC
  - root = predicate
  - leaves = arguments
- Example

```
(0.028, [:V:donner::] (
           [SUJ:N:::],
           [OBJ:N:::],
           [AOBJ:PREP:a:a:] (
             [OBJ:N:::])))}
```

- score $s_{\mathsf{SF}} = \mathsf{P}(\mathsf{T}|v)$

Selecting the Optimal Set of Subcategorization Frames

- *Notations*
  - $R(j)$ predicate of ISF j
  - $\mathcal{L}(j)$ arguments of ISF j
- *Definition of the variables*
  - $\alpha_i^j = 1$ if word i is the predicate of ISF number j, 0 otherwise
  - $\beta_i^j = 1$ if word i is an argument of ISF number j, 0 otherwise
- *Definition of the constraints*
  - a word is the predicate of at most one ISF :
    $\forall i \in \{1, \ldots, N\} \ \sum_{j \in \mathcal{I}} \alpha_i^j \leq 1$
  - a word cannot be an argument of more than one ISF :
    $\forall i \in \{1, \ldots, N\} \ \sum_{j \in \mathcal{I}} \beta_i^j \leq 1$
  - for an ISF to be selected, its pred. and all its args must be :
    $\forall j \in \{1, \ldots, |\mathcal{I}|\} \ |\mathcal{L}(j)| \alpha_{R(j)}^j - \sum_{l \in \mathcal{L}(j)} \beta_l^j = 0$
- *Definition of the objective function*

$$\max \sum_{j \in \mathcal{I}} \alpha_{R(j)}^j s_{SF}(j)$$

Example

S = *Jean rend le livre qu'il a emprunté à la bibliothèque.*
(*Jean returns the book that he has borrowed at the library.*

```
1 (0.2,[:V:rend:rendre:2](
      [SUJ:N:Jean:Jean:1],
      [OBJ:N:livre:livre:4]))

2 (0.4,[:V:rend:rendre:2](
      [SUJ:N:Jean:Jean:1],
      [OBJ:N:livre:livre:4],
      [AOBJ:PREP:a:a:9](
        [OBJ:N:biblio.:biblio.:11])))

3 (0.3,[:V:emprunte:emprunter:8] (
      [SUJ:N:il:il:6],
      [OBJ:N:qu':que:5]))

4 (0.6,[:V:emprunte:emprunter:8] (
      [SUJ:N:il:il:6],
      [OBJ:N:qu':que:5],
      [AOBJ:PREP:a:a:9] (
        [OBJ:N:biblio.:biblio.:11])))
```

$\hat{\mathcal{I}} = \{1, 4\}$.

Selectional Constraints (SC)

- Special kind of LSC
  - one lexical root
  - one lexical leaf
- tendency of the root and the leaf to co-occur in a specific syntactic configuration.
- four configurations :

```
([:V:::]([SUJ:N:::]))
([:V:::]([OBJ:N:::]))
([:V:::]([DOBJ:P:de::]([OBJ:N:::])))
([:V:::]([AOBJ:P:a::]([OBJ:N:::])))
```

SC Scores

- The score of a SC reflects the tendency of the root r and the leaf l to appear together in configuration C.
- It is maximal if :
  - whenever r occurs as the root of configuration C, the leaf position is occupied by l
  - and, symmetrically, if whenever l occurs as the leaf of configuration C, the root position is occupied by r.

$$s_{SC}(C, r, l) = \frac{1}{2}\left(\frac{\mathcal{C}(C, r, l)}{\mathcal{C}(C, r, *)} + \frac{\mathcal{C}(C, r, l)}{\mathcal{C}(C, *, l)}\right)$$

- $\mathcal{C}(C, l, *)$: occurrences of conf. C with l as a root
- $\mathcal{C}(C, *, l)$: occurrences of conf. C with l as a leaf
- $\mathcal{C}(C, r, l)$: occurrences of conf. C with r as a root and l as a leaf.

Selecting the Optimal Set of Selectional Constraints

- *Definition of the variables*
  - $\gamma_i^j = 1$ if word i is the root of ISC number j, 0 otherwise
  - $\delta_i^j = 1$ if word i is the leaf of ISC number j, 0 otherwise
- *Definition of the constraints*
  - a word cannot be the leaf of more than one ISC
    $\forall i \in \{1, \dots N\} \; \sum_{j \in \mathcal{I}'} \delta_i^j \leq 1$
  - for ISC j to be selected, both its root and its leaf must be
    $\forall j \in \{1, \dots, |\mathcal{I}'|\} \; \gamma_{R(j)}^j - \delta_{d \in \mathcal{L}(j)}^j = 0$
- *Definition of the objective function*

$$\max \sum_{j \in \mathcal{I}'} \gamma_{R(j)}^j s_{SC}(j)$$

Example

S =*Jean rend le livre qu'il a emprunté à la bibliothèque.*

```
5 (0.2,[:V:rend:rendre:2]
       ([SUJ:N:Jean:Jean:1]))
6 (0.2,[:V:rend:rendre:2]
       ([OBJ:N:livre:livre:4]))
7 (0.4,[:V:rend:rendre:2]
       ([AOBJ:PREP:a:a:9]
          ([OBJ:N:biblio.:biblio.:11])))
8 (0.6,[:V:emprunte:emprunter:8]
       ([AOBJ:PREP:a:a:9]
          ([OBJ:N:biblio.:biblio.:11])))
```

$\hat{\mathcal{I}} = \{5, 6, 8\}$.

Combining Subcategorization Frames and Selectional Constraints

- *Definition of the variables*
  $\alpha_i^j, \beta_i^j, \gamma_i^j, \delta_i^j$
- *Definition of the constraints*
  - All the constraints of ISF selection and ISC satisfaction
  - Incompatible ISF and ISC cannot be selected together
    An ISF j and an ISC j' are not compatible if they share a common leaf but have different roots.

$$\forall i, i', j, j' \; \alpha_i^j + 2\beta_{i'}^j + \gamma_i^{j'} + 2\delta_{i'}^{j'} \neq 5$$

- *Definition of the objective function*

$$\max \left(\sum_{j \in \mathcal{I}'} \delta_{R(j)}^j s_{SC}(j) + \sum_{j \in \mathcal{I}} \alpha_{R(j)}^j s_{SF}(j) \right)$$

Two Processes

- Dynamic Programming : Parsing

$$\hat{T} = \arg\max_{T \in \mathcal{T}(S)} \sum_{X \in \psi(T)} s(X)$$

- Linear Programming : Patching

$$\hat{\mathcal{C}} = \arg\max_{E \subseteq \mathcal{C}(S)} \sum_{X \in E} s'(X)$$

- How to combine them ?

Combining

- **Stong integration**
  - Parsing as an Integer Program
  - Patching as a Dynamic Programming problem
- **Weak integration**
  - Run the processes separately
  - Combine the outputs

Weak integration

1. Candidate generation
   - Produce k best parses of sentence $\mathcal{S}$.
   - Build set $\mathcal{I}$ of ISFs and set $\mathcal{I}'$ of ISCs.
2. Candidate selection
   - Patch $\mathcal{S}$ with $\mathcal{I}$ and $\mathcal{I}'$ to get set $\hat{\mathcal{I}}''$ of ISF and ISC.
3. Constrained Parsing
   - Compute new scoring function $s^{+}_{\hat{\mathcal{I}}''}$
   - Produces a parse tree $\hat{\mathsf{T}}$ that preserves the ISF and ISC computed in step 2.

Bias

- Patching always win !
- integrating confidence measure in the patching process :

$$\hat{s}_{SF}(\Delta) = (1 - \mu_{SF})s_{SF}(\Delta) + \mu_{SF}CM(\Delta)$$

$$\hat{s}_{SC}(\Delta) = (1 - \mu_{SC})s_{SC}(\Delta) + \mu_{SC}CM(\Delta)$$

Raw Data

CORPUS	Sent. nb.	Tokens nb.
AFP	2 041 146	59 914 238
EST REP	2 998 261	53 913 288
WIKI	1 592 035	33 821 460
TOTAL	5 198 642	147 648 986

SF extraction

- Linguistic constraints
  - category of the root $\in$ {V, VINF, VPP, VPR}
  - functional labels $\in$ {SUJ, OBJ, A_OBJ, DE_OBJ}
  - category of the pre leaves $\in$ {P, CS}
  - category of the leaves $\in$ {ADJ, N, V, VINF, VPP, VPR}
- Abstraction
  - abstract over linear order
  - group together proper nouns, common nouns and pronouns into a single category N

Some statistics

	A_0	A_5	A_{10}
nb of verbs	23 915	4 871	3 923
nb of diff SF	12 122	2 064	1 355
avg. nb of SF	14.26	16.16	13.45

Coverage

		A_0	A_5	A_{10}	T
Lex. cov.	types	99.52	98.56	98.08	89.56
	tokens	99.85	99.62	99.50	96.98
Synt. cov.	types	95.78	91.08	88.84	62.24
	tokens	97.13	93.96	92.39	73.54

Statistics

Conf.	A_0	A_5	A_{10}
OBJ	422 756	58 495	26 847
SBJ	433 196	55 768	25 291
VdeN	116 519	11 676	4 779
VaN	185 127	23 674	10 729
TOTAL	1 157 598	149 613	67 646

Coverage

Conf.		A_0	A_5	A_{10}	T
OBJ	types	68.31	58.50	53.24	17.94
	tokens	69.71	61.51	56.12	21.31
SBJ	types	66.26	52.87	47.82	23.11
	tokens	68.18	57.24	52.94	24.48
VdeN	types	46.28	34.93	31.66	11.57
	tokens	56.61	49.57	46.73	15.77
VaN	types	62.93	48.89	42.68	20.06
	tokens	64.69	52.42	47.56	23.34
TOTAL	types	65.17	53.10	47.92	19.96
	tokens	67.54	57.15	52.78	22.24

Accuracy results

	LAS	UAS	SCAS	SFAS
Baseline	88.88	90.71	87.81	80.84
SF	89.54	91.37	88.44	84.57
SC	89.73	91.60	92.32	82.14
SF + SC	89.85	91.72	93.08	85.26

- **Labeled Accuracy Score (LAS)**
  ratio of correct labeled dependencies in $\hat{T}$.
- **Unlabeled Accuracy Score (UAS)**
  ratio of correct unlabeled dependencies in $\hat{T}$.
- **Subcategorization Frame Accuracy Score (SFAS)**
  ratio of verbs in $\hat{T}$ that have been assigned their correct SF.
- **Selectional Constraint Accuracy Score (SCAS)**
  ratio of correct occurrences of SC patterns in $\hat{T}$.

Future Work

- Better candidate generation : use of parse forests
- Patching with Framenet Frames
- Patching with Discourse Patterns