Learning and Inference for Clause Identification

Xavier Carreras **Lluís Màrquez**
Technical University of Catalonia (UPC)

Vasin Punyakanok **Dan Roth**
University of Illinois at Urbana-Champaign (UIUC)

ECML 2002

Outline

- Clause Identification.
- Inference Scheme.
- Learned Functions.
- Experimentation.
- Conclusions.

Goal

We want to identify **clauses** in a **sentence**.

- Clause = sequence of words with a subject (maybe implicit) and a predicate.

`( (When (you don't have any other option)), it's easy (to fight) .)`

Goal

We want to identify **clauses** in a **sentence**.

- Clause = sequence of words with a subject (maybe implicit) and a predicate.

```
( (When (you don't have any other option)),
        it's easy (to fight) .)
```

- Clauses in a sentence form a **hierarchical structure**.
- We do **not** consider clause types (main, relative, noun, adverbial, ...).

Embedded Bracketing

- **Input:** a sequence of words (with extra information).

Embedded Bracketing

- **Input:** a sequence of words (with extra information).
- **Output:** a bracketing codifying the hierarchical clause structure, in which:

Embedded Bracketing

- **Input:** a sequence of words (with extra information).
- **Output:** a bracketing codifying the hierarchical clause structure, in which:
  - ⋆ A clause is codified by its **boundaries**:

    `w w` **(** *words within the clause* **)** `w w w` $\surd$

Embedded Bracketing

- **Input:** a sequence of words (with extra information).
- **Output:** a bracketing codifying the hierarchical clause structure, in which:
  - ⋆ A clause is codified by its **boundaries**:

    w w (*words within the clause*) w w w √

  - ⋆ **Overlapping** of clauses is **not** permitted:

    w w (w w (w w w w) w w w) w ×

Embedded Bracketing

- **Input:** a sequence of words (with extra information).
- **Output:** a bracketing codifying the hierarchical clause structure, in which:
  - ⋆ A clause is codified by its **boundaries**:

    w w (*words within the clause*) w w w √
  - ⋆ **Overlapping** of clauses is **not** permitted:

    w w (w w (w w w w) w w w) w ×
  - ⋆ Clauses are possibly **embedded**.

    ((w w) w w (w w w (w w w))) √

Syntactic Parsing

Clause Identification $\in$ **Syntactic Parsing** $\in$ NLP

- Grammar-based methods:
  - ⋆ Grammars: manually constructed, inferred, . . .
  - ⋆ Parsing Schemes: CKY, Early, . . .
  - ⋆ PCFG's, parameter estimation.
- No-Explicit-Grammar Parsers:
  - ⋆ Usually, intensive use of **learning** techniques.
  - ⋆ Decision Trees, [Magerman 96]
  - ⋆ Maximum-Entropy parser, [Ratnaparkhi 98]
  - ⋆ Partial Parsing techniques, [Abney 91] [CoNLL tasks]

Learning and Inference for Partial Parsing

- **Local** classifiers: solve dependent partial decisions, e.g.:
  - ⋆ Whether a word **opens** and/or **closes** a constituent.
  - ⋆ Whether a word **starts** or **continues** a constituent.
- **Inference** is made on the outcome of local classifiers to produce a **global** solution, **coherent** wrt. the problem **constraints**. [Roth ECML'02]
- Much work in **chunking**, for plain structures (non-overlapping & non-embedding) [CoNLL'00]
- We propose an inference scheme for **clausing** (non-overlapping & embedding) [CoNLL'01]

Outline

- Clause Identification.
- Inference Scheme.
- Learned Functions.
- Experimentation.
- Conclusions.

Our Approach

- Learned Functions (classifiers):
  - ⋆ **S**tart of a clause: $spoint$
  - ⋆ **E**nd of a clause: $epoint$
  - ⋆ Score of a clause: $score$
- Algorithm: Recursively from the bottom-up . . .
  - ⋆ Generate clause candidates.
  - ⋆ Select best split of clauses.

Identifying Clause Candidates

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				

Identifying Clause Candidates

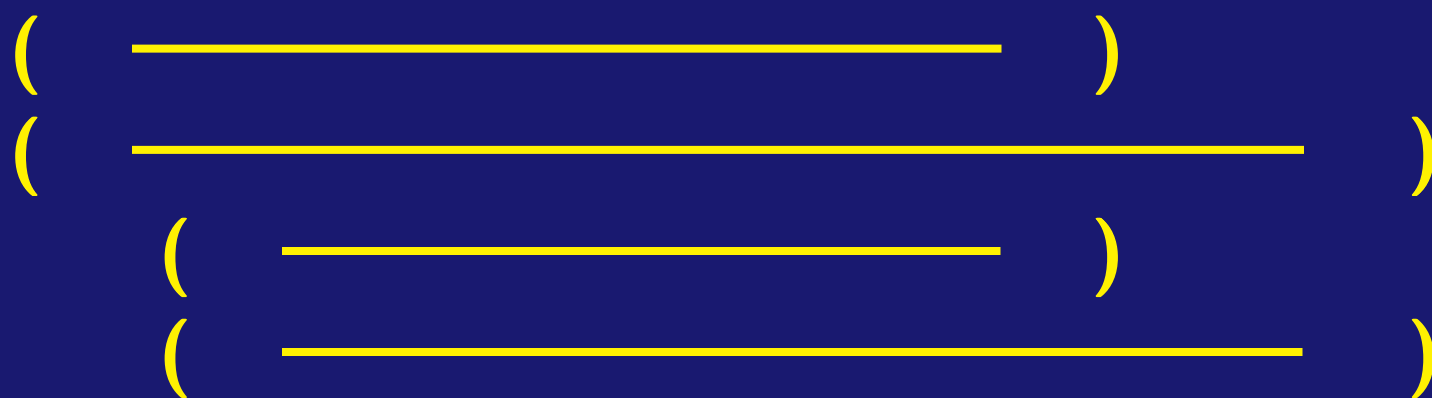
	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				
$epoints$								e_8		e_{10}

Identifying Clause Candidates

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				
$epoints$								e_8		e_{10}
	(	—	—	—	—	—	—	)		
	(	—	—	—	—	—	—	—	—	)

Identifying Clause Candidates

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				
$epoints$								e_8		e_{10}



Identifying Clause Candidates

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				
$epoints$								e_8		e_{10}

Possible Clause Splits

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				
$epoints$								e_8		e_{10}
	(	(				(		))		)

Possible Clause Splits

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				
$epoints$								e_8		e_{10}
	(	(				(		))		)
	(	(				(		)		))

Possible Clause Splits

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				
$epoints$								e_8		e_{10}
	(	(				(		))		)
	(	(				(		)		))
	(	(				((		)		)))

Possible Clause Splits

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				
$epoints$								e_8		e_{10}
	(	(				(		))		)
	(	(				(		)		))
	(	(				((		)		)))
	((							)		)

Possible Clause Splits

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
$spoints$	s_1	s_2				s_6				
$epoints$								e_8		e_{10}
	(	(				(		))		)
	(	(				(		)		))
	(	(				((		)		)))
	((							)		)
	(	((						)		))

. . .

Clause Score

Each clause candidate (s, e) is scored by a function:

$$score(s, e) \to \mathbb{R}$$

Given the score of (s, e):

- The **sign** tells whether (s, e) is a clause or not.
- The **magnitude** codifies the **confidence** of the decision.

Optimal Clause Split

Δ: set containing all possible splits.

S: a split, i.e. a coherent set of clauses, $\{(s_i, e_i)\}_{i=1}^{l}$.

$$S^* = \arg\max_{S \in \Delta} \sum_{(s,e) \in S} score(s,e)$$

Optimal Clause Split

Δ: set containing all possible splits.
S: a split, i.e. a coherent set of clauses, $\{(s_i, e_i)\}_{i=1}^{l}$.

$$S^* = \arg\max_{S \in \Delta} \sum_{(s,e) \in S} score(s, e)$$

The optimal clause split can be efficiently found . . .

- Using **dynamic programming** techniques.
- Exploring the sentence from the **bottom-up**.

Bottom-up Exploration

$$\ldots \quad w_s \quad w \quad w \quad w \quad w_e \quad \ldots$$

Bottom-up Exploration

$\ldots \quad w_s \quad w \quad w \quad w \quad w_e \quad \ldots$

internal split 1 $\quad \boxed{w_s} \; \boxed{w \quad w \quad w \quad w_e}$

Bottom-up Exploration

$$\ldots \quad w_s \quad w \quad w \quad w \quad w_e \quad \ldots$$

internal split 1 $\quad \boxed{w_s} \; \boxed{w \quad w \quad w \quad w_e}$

internal split 2 $\quad \boxed{w_s \quad w} \; \boxed{w \quad w \quad w_e}$

Bottom-up Exploration

$$\ldots \quad w_s \quad w \quad w \quad w \quad w_e \quad \ldots$$

internal split 1 $\quad [w_s] \; [w \; w \; w \; w_e]$

internal split 2 $\quad [w_s \; w] \; [w \; w \; w_e]$

internal split 3 $\quad [w_s \; w \; w] \; [w \; w_e]$

Bottom-up Exploration

$\ldots \quad w_s \quad w \quad w \quad w \quad w_e \quad \ldots$

internal split 1 [w_s] [w w w w_e]

internal split 2 [w_s w] [w w w_e]

internal split 3 [w_s w w] [w w_e]

internal split 4 [w_s w w w] [w_e]

Bottom-up Exploration

$\ldots \quad w_s \quad w \quad w \quad w \quad w_e \quad \ldots$

internal split 1	$[w_s]$	$[w \; w \; w \; w_e]$
internal split 2	$[w_s \; w]$	$[w \; w \; w_e]$
internal split 3	$[w_s \; w \; w]$	$[w \; w_e]$
internal split 4	$[w_s \; w \; w \; w]$	$[w_e]$

(s,e) clause ? $\quad (\; w_s \quad w \quad w \quad w \quad w_e \;)$

General Algorithm

```
function optimal_clause_split (s, e)
  if (s ≠ e) then
    optimal_clause_split(s, e − 1)
    optimal_clause_split(s + 1, e)
  Δ := { BestSplit[s, r] ∪ BestSplit[r + 1, e] | s ≤ r < e }
  S* := arg max_{S∈Δ} Σ_{(k,l)∈S} Score[k, l]
  if (spoint(s) and epoint(e)) then
    Score[s, e] := score(s, e)
    if (Score[s, e] > 0) then
      S* := S* ∪ {(s, e)}
  BestSplit[s, e] := S*
end function
```

Outline

- Clause Identification.
- Inference Scheme.
- Learned Functions.
- Experimentation.
- Conclusions.

Spoints and Epoints

- Example to be classified: word.
- Decide whether a word **S**tarts and/or **E**nds a clause.
- Use of a sliding window to codify the local context with binary features:

					?				
form	w_{i-4}	w_{i-3}	w_{i-2}	w_{i-1}	w_i	w_{i+1}	w_{i+2}	w_{i+3}	w_{i+4}
PoS	p_{i-4}	p_{i-3}	p_{i-2}	p_{i-1}	p_i	p_{i+1}	p_{i+2}	p_{i+3}	p_{i+4}
chunk	c_{i-4}	c_{i-3}	c_{i-2}	c_{i-1}	c_i	c_{i+1}	c_{i+2}	c_{i+3}	c_{i+4}

Score Function

- Example: clause candidate (i.e. sequence of words)
- Clause candidates are represented by **patterns**:

Verb Phrases	Conjunctions	Adverbs
Punctuation	Relative Pronouns	. . .

- Example:

```
(( When ( you don't have any other option )) ,
      it's easy ( to fight ) .  )
```

```
When ~ VERB ~ , ~ VERB ~ VERB .
```

Score Function

- Example: clause candidate (i.e. sequence of words)
- Clause candidates are represented by **patterns**:

Verb Phrases	Conjunctions	Adverbs
Punctuation	Relative Pronouns	. . .

- Example:

`(( When ( you don't have any other option )) , it's easy ( to fight ) . )`

`When ∼ VERB ∼ , ∼ VERB ∼ VERB .`

- **Problem:** clauses can be very long.

Score: Subordinate Reduction

(Not everyone believes (that
(the good times are over for shippers)) .)

it.1 | ...that (the good times are over for shippers).

Score: Subordinate Reduction

(Not everyone believes (that
(the good times are over for shippers)) .)

it.1 | ...that (the good times are over for shippers).
$\Longrightarrow$ CLAUSE

Score: Subordinate Reduction

(Not everyone believes (that
(the good times are over for shippers)) .)

it.1	...that (the good times are over for shippers). $\Longrightarrow$ CLAUSE
it.2	...one believes (that the good ... shippers).

Score: Subordinate Reduction

(Not everyone believes (that
(the good times are over for shippers)) .)

it.1 | ...that (the good times are over for shippers).
$\Longrightarrow$ CLAUSE

it.2 | ...one believes (that the good ... shippers).
...one believes (that CLAUSE).

Score: Subordinate Reduction

(Not everyone believes (that
(the good times are over for shippers)) .)

it.1 | ...that (the good times are over for shippers).
$\Longrightarrow$ CLAUSE

it.2 | ...one believes (that the good ... shippers).
...one believes (that CLAUSE).
$\Longrightarrow$ CLAUSE

Score: Subordinate Reduction

(Not everyone believes (that (the good times are over for shippers)) .)

it.1 | ...that (the good times are over for shippers).
$\Longrightarrow$ CLAUSE

it.2 | ...one believes (that the good ... shippers).
...one believes (that CLAUSE).
$\Longrightarrow$ CLAUSE

it.3 | (Not everyone believes that ... shippers .)

Score: Subordinate Reduction

(Not everyone believes (that
(the good times are over for shippers)) .)

it.1	...that (the good times are over for shippers).
	$\Longrightarrow$ CLAUSE
it.2	...one believes (that the good ... shippers).
	...one believes (that CLAUSE).
	$\Longrightarrow$ CLAUSE
it.3	(Not everyone believes that ... shippers .)
	(Not everyone believes CLAUSE .)

Score: Subordinate Reduction

(Not everyone believes (that
(the good times are over for shippers)) .)

it.1 | ...that (the good times are over for shippers).
$\Longrightarrow$ CLAUSE

it.2 | ...one believes (that the good ... shippers).
...one believes (that CLAUSE).
$\Longrightarrow$ CLAUSE

it.3 | (Not everyone believes that ... shippers .)
(Not everyone believes CLAUSE .)
$\Longrightarrow$ CLAUSE

Score: Coordinate Reduction

((Sapporo gained 80 to 1,050)
and
(Kirin added 60 to 2,000) .)

it.1 | (Sapporo gained 80 to 1,050) and Kirin ...

Score: Coordinate Reduction

((Sapporo gained 80 to 1,050)
and
(Kirin added 60 to 2,000) .)

it.1 | (Sapporo gained 80 to 1,050) and Kirin ...
$\Longrightarrow$ CLAUSE

Score: Coordinate Reduction

((Sapporo gained 80 to 1,050)
and
(Kirin added 60 to 2,000) .)

it.1 | (Sapporo gained 80 to 1,050) and Kirin ...
$\Longrightarrow$ CLAUSE
it.1 | ...1,050 and (Kirin added 60 to 2,000) .

Score: Coordinate Reduction

((Sapporo gained 80 to 1,050)
and
(Kirin added 60 to 2,000) .)

it.1 | (Sapporo gained 80 to 1,050) and Kirin ...
$\Longrightarrow$ CLAUSE

it.1 | ...1,050 and (Kirin added 60 to 2,000) .
$\Longrightarrow$ CLAUSE

Score: Coordinate Reduction

((Sapporo gained 80 to 1,050)
and
(Kirin added 60 to 2,000) .)

it.1	(Sapporo gained 80 to 1,050) and Kirin ...
	$\Longrightarrow$ CLAUSE
it.1	...1,050 and (Kirin added 60 to 2,000) .
	$\Longrightarrow$ CLAUSE
it.2	(Sapporo gained ... and ... to 2,000 .)

Score: Coordinate Reduction

((Sapporo gained 80 to 1,050)
and
(Kirin added 60 to 2,000) .)

it.1	(Sapporo gained 80 to 1,050) and Kirin . . .
	$\Longrightarrow$ CLAUSE
it.1	. . .1,050 and (Kirin added 60 to 2,000) .
	$\Longrightarrow$ CLAUSE
it.2	(Sapporo gained . . . and . . . to 2,000 .)
	(CLAUSE and CLAUSE .)
	$\Longrightarrow$ CLAUSE

Scoring Functions

- **Plain Scoring**: No reduction of previously identified clauses. Consists of one classifier.

- **Structured Scoring**: Reduction of the optimal split identified inside the current candidate. The function is a composition of three specialized classifiers:
  - ⋆ Base clauses.
  - ⋆ Recursive clauses, assuming complete split.
  - ⋆ Recursive clauses, assuming partial split.

Learning Algorithm: AdaBoost

- real AdaBoost with confidence-rated predictions.
  [Schapire & Singer '99]
- $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$
  - ⋆ The sign codifies the predicted class.
  - ⋆ The magnitude is a confidence score of the prediction.
- Weak Rules (h_t): Decision Trees of small depth (3-4).
- Good performance in NLP domains.

Outline

- Clause Identification.
- Inference Scheme.
- Learned Functions.
- Experimentation.
- Conclusions.

CoNLL 2001 Setting

- Data Set:
  - ⋆ Penn Treebank: Wall Street Journal
  - ⋆ Words, POS tags, chunks.
  - ⋆ Training Set: sections 15-18 (8,936 sentences).
  - ⋆ Development Set: section 20 (2,012 sentences).
  - ⋆ Test Set: section 21 (1,671 sentences).
- Evaluation: precision, recall, $F_{\beta=1}$

Results on the Development Set

	prec.	rec.	$F_{\beta=1}$
Plain Scoring	88.33%	**83.92%**	86.07%
Structured Sco.	**92.53%**	82.48%	**87.22%**

CoNLL'01 CM	87.18%	82.48%	84.77%

Results on the Test Set

	prec.	rec.	$F_{\beta=1}$
Plain Scoring	85.25%	**74.53%**	79.53%
Structured Sco.	**90.18%**	72.59%	**80.44%**

CM01	84.82%	73.28%	78.63%
MP01	70.89%	65.57%	68.12%
TKS01	76.91%	60.61%	67.79%
PG01	73.75%	60.00%	66.17%
Dej01	72.56%	54.55%	62.77%
Ham01	55.81%	45.99%	50.42%

Conclusions

- We have presented an **inference scheme** for recognizing **hierarchical structure**.
- All the decisions involved in the process are solved with learning techniques.
- Local decisions take advantage of the partial solution.
- On Clause Identification, our approach improves top-performing methods . . .

  . . . but there is still room for improvement.