Learning and Inference in Phrase Recognition: A Filtering-Ranking Architecture using Perceptron

Ph.D. Dissertation by
Xavier Carreras Pérez

advised by
Lluís Màrquez Villodre
Introduction

Outline

• Introduction: Phrase Recognition
• Learning Methods for Text Analysis Tasks
• Filtering-Ranking Architecture
• Systems and Results on Syntactic-Semantic Parsing
• Conclusion and Future Research
Natural Language Learning for Text Analysis

- NLL: Learning as a central mechanism to process natural language
- Text Analysis: a fundamental task in NLP
  - Consists of recognizing the linguistic structures underlying text
  - Useful for applications dealing with language:
    - Intelligent Information Access (e.g., Question-Answering)
    - Machine Translation Systems
    - . . .
Phrase Recognition

- A family of text analysis tasks
- What is a phrase, in general?
  - a group of words performing a function as a unit
- Many problems in Natural Language consist of recognizing phrases in a sentence
- *a.k.a.* segmentation problems, tagging and parsing problems
Syntactic Parsing

- Phrase = constituent: a group of words performing a syntactic function

- Several levels/versions of the problem:
  - Full Parsing: recover the full syntactic tree
  - Partial Parsing: recover only some syntactic elements:
    - Chunking: recognize chunks, i.e., base non-recursive phrases
    - Noun-Phrase recognition: recognize the structure of NPs
    - Clause Identification: recover the clauses (usually in hierarchy)
    - . . .
Phrase Recognition in Partial Syntactic Analysis
Phrase Recognition in Partial Syntactic Analysis

The cat trapped with a hat that I had.

Diagram:

```
NP  VP  NP  PP  NP  NP  NP  VP
The  cat  trapped  the  rat  with  a  hat  that  I  had  .
```
Phrase Recognition in Partial Syntactic Analysis

The cat trapped with a hat that I had.
Introduction

Semantic Role Labeling

- Phrase $=$ Argument : a group of syntactic units playing a role with a predicate

- Example:
  (The cat)$_{AG}$ trapped (the rat)$_{PAC}$ (with a hat)$_{INS}$

  - For the predicate “trap”:
    - AG is the agent (the entity that traps)
    - PAC is the pacient (the thing trapped)
    - INS is the instrument
Phrase Recognition in Syntactic-Semantic Analysis

The cat trapped with a hat that I.

The diagram illustrates the syntactic and semantic analysis of the sentence, showing the constituents and their relationships.
Phrase Recognition: general

• Goal: find phrases in a sentence, of types in $\mathcal{K}$

• Solution: a set of phrases, each of the form $(s, e)_k$, satisfying that:
  ★ Phrases do not overlap (do not cross boundaries)
  ★ Sequential Structures: phrases do not embed
  ★ Hierarchical Structures: phrases may be embedded

• Evaluation: Precision/Recall/$F_1$ of recognized phrases
Sequential Phrase Structure: schematic view
Hierarchical Phrase Structure: schematic view
Observations (i): Huge Output Space

Output space is exponential: parsing strategy required
Observations (ii): Recursive Structures

Desirable to put learning in high-order level
This Thesis

• Proposes a general learning architecture for phrase recognition

• Presents state-of-the-art systems for several NL problems:
  ★ Syntactic Chunking
  ★ Clause Identification
  ★ Semantic Role Labeling
Outline

- Introduction: Phrase Recognition
- Learning Methods for Text Analysis Tasks
- Filtering-Ranking Architecture
- Systems and Results on Syntactic-Semantic Parsing
- Conclusion and Future Research
Supervised Machine Learning

- Given:
  - A training set, with examples \((x, y)\) where
    - \(x \in X\) could be sentences
    - \(y \in Y\) could be linguistic structures
    - We assume that the set was generated i.i.d. from an unknown distribution \(D\) over \(X \times Y\)
  - An error function, or loss:
    \[
    \text{error}(y, \hat{y}) = \text{cost of proposing } \hat{y} \text{ when the correct value was } y
    \]
- Goal: learn a hypothesis
  \[
  h : X \rightarrow Y
  \]
  that minimizes error on the entire distribution \(D\)
Scenarios in Machine Learning

A general form of learning hypothesis:

\[ h(x) = \arg \max_{\hat{y} \in \mathcal{Y}} \text{score}(x, \hat{y}) \]

Depending on the output space \( \mathcal{Y} \):

| Classes (\( \mathcal{Y} \)) | \( |\mathcal{Y}| \) | enumeration of \( \mathcal{Y} \) | error |
|-------------------------------|----------------|-------------------------------|-------|
| **Binary Classification**     | \{+, −\}       | 1                             | not needed | 0-1   |
| **Multiclass Classification** | A,B,C, . . .    | \( m \)                       | exhaustive | 0-1   |
| **Structure Learning**        | all structures | exponential                   | not tractable | prec/rec on nodes |
Structure Learning: Learning & Inference

- $\mathcal{Y}(x)$ is exponential on the size of $x$

- Not possible to exhaustively enumerate the output space

- Learning & Inference approach:
  - **Key Idea:** decompose a structure into fragments
  - **Model:** scores a structure by scoring its fragments
  - **Inference:** search in $\mathcal{Y}(x)$ for the best scored solution for $x$
    - Build incrementally, instead of explore exhaustively
    - Use automata, grammars, . . . to build the solution
    - Use constraints to discard regions of $\mathcal{Y}(x)$
Generative Learning (i): Models

- Probabilistic models that define a joint probability distribution of the data
- The model is associated to a stochastic generation mechanism of the data, such as an automaton or grammar
- Paradigmatic models to recognize structure:
  - Hidden Markov Models, e.g. [Rabiner 89]
  - Probabilistic Context-Free Grammars, e.g. [Collins 99]
Generative Learning (ii): Max-Likelihood Estimation

- Based on theory of probability and Bayesian learning:
  - Training: via Maximum Likelihood, i.e., counts on training
  - Inference Algorithms: e.g., Viterbi, CKY, etc.

- But:
  - Difficult to use arbitrary representations
    - Features are tied to the generation mechanism of the data
    - Otherwise, the training process becomes too complex
  - Asymptotic convergence wrt. the size of training data
Direct, Discriminative Learning

- ML methods that directly model the mapping between \( \mathcal{X} \) and \( \mathcal{Y} \)
- Allow arbitrary representations
- Not necessarily probabilistic
- Mostly designed for classification, mostly binary
- A wide range of methods appeared in the AI community during the 80’s and 90’s:
  - Maximum Entropy
  - Decision Trees (or Lists)
  - Memory-based
  - Transformation-based
  - Neural Nets, Perceptron
  - AdaBoost
  - Support Vector Machines
  - . . .
Learning and Inference: General Approach

• Transform the recognition problem into a chain of *simple* decisions:
  ★ Segmentation Decisions: e.g., Open-Close, Begin-Inside-Outside, Shift-Reduce, etc.
  ★ Labeling Decisions: made during segmentation or afterwards
  ★ Decisions might use the output of earlier steps in the chain

• Set up an inference strategy:
  ★ Decisions are applied in chain to build structure incrementally
  ★ Exploration might be at different levels of amplitude: e.g., greedy, dynamic programming, beam search, etc.

• Learn a prediction function for each decision
Learning & Inference: Local vs. Global Training

- Local training: each local function is trained separately, as a classifier (binary or multiclass)
  - Good understanding on learning classifiers
  - but local accuracies do not guarantee global accuracy
  - that is, a local classification behavior might not be the optimal within inference
  - unless local classifications are perfect

- Global training: train the recognizer as a composed function
  - Local functions are trained dependently to optimize global accuracy
  - e.g., Linear models [Collins 02,04], CRFs [Lafferty et al. 01]
Learning Linear Separators (i)

• Most learning algorithms look for linear separators, under different criteria [Roth 98,99][Collins 02]

• Properties: simple, expressive, efficient

• Flexible at learning different prediction policies

• A linear separator has the following form:

\[ \text{score}(x, y) = \mathbf{w} \cdot \phi(x, y) \]

where:
★ \( \phi \) is a feature extraction function, given \textit{a priori}
★ \( \mathbf{w} \) is a weight vector, learned by the algorithm
Learning Linear Separators (ii): Separability

- Recent theoretical work concentrates on learning linear separators
- **Separability**: ability to separate between correct/incorrect instances
- **[Vapnik 95]**:
  - large separation on training $\implies$ low generalization error
  - A quantity called **margin** measures how much a hypothesis separates between correct/incorrect instances
  - Margin-based algorithms: look for linear separators that . . .
    - Perceptron: achieve positive margins
    - Support Vector Machines: achieve maximum margins
Learning Linear Separators (iii): Perceptron

- Online algorithm, with additive mistake-driven updates:
  - Promotion, when a prediction is too low (controls recall)
  - Demotion, when a prediction is too high (controls precision)
- With appropriate definitions of margin, can be used for:
  - Binary classifiers [Rosenblatt 58]
  - Multiclass [Crammer & Singer 03]
  - Ranking functions [Collins 02]
- Extensions: Voted Perceptron [Freund & Schapire 99]
  - Voting techniques to obtain larger margins
  - Kernel method: polynomial functions, structure kernels, . . .
Outline

- Introduction: Phrase Recognition
- Learning Methods for Text Analysis Tasks
- Filtering-Ranking Architecture
- Systems and Results on Syntactic-Semantic Parsing
- Conclusion and Future Research
Filtering-Ranking Architecture

- A general architecture to recognize phrase structures
- Two levels of learning:
  - Filter: decides which words start/end a phrase
  - Ranker: scores phrases
- On the top, dynamic programming inference builds the best-scored phrase structure
- We propose FR-Perceptron: a Perceptron learning algorithm tailored for the architecture
Filtering-Ranking Architecture: Decomposition

- A solution is decomposed at phrase level:

  \[ \text{score}(x, y) = \sum_{(s,e)_k \in y} \text{score}_p(x, y, (s, e)_k) \]

- Still, the number of phrases grows quadratically with the sentence length

- We reduce the space of phrases by filtering at word level. For a phrase \((s, e)_k\) to be in a solution:

  \[ \text{start}_w(x, s, k) > 0 \ \land \ \text{end}_w(x, e, k) > 0 \]
Filtering-Ranking Model

$\mathcal{Y}$: solution space, i.e. set of all phrase structures

$\mathcal{Y}_{SE}$: practical solution space, filtered at word level:

$$\mathcal{Y}_{SE} = \{y \in \mathcal{Y} \mid \forall (s, e)_k \in y \text{  start}_w(x, s, k) \land \text{end}_w(x, e, k)\}$$

The Filtering-Ranking architecture computes:

$$R(x) = \arg \max_{y \in \mathcal{Y}_{SE}} \sum_{(s, e)_k \in y} \text{score}_p(x, y, (s, e)_k)$$

using dynamic-programming.
Filtering-Ranking Architecture

Filtering-Ranking Strategy

(● (((● (● ● ●)) ●) ● ● (● (((● ● ● ● ●) (● ● (● (●)))))●) ●) ● (● (((● ● ● ● ●) (● ● (● (●)))))●) ●)
Filtering-Ranking Strategy
Filtering-Ranking Architecture

Filtering-Ranking Strategy

- Black +: Positive Scores
- Gray +: Positive Scores
- White +: Negative Score
Filtering-Ranking Architecture

Filtering-Ranking Strategy

- **Positive Scores**
- **Negative Score**

Diagram showing the filtering-ranking strategy with symbols for positive and negative scores.
Filtering-Ranking Strategy

Positive Scores
Negative Score
Filtering-Ranking Architecture

Filtering-Ranking Strategy

- Positive Scores
- Negative Score
Filtering-Ranking Strategy

- Positive Scores
- Negative Score
Filtering-Ranking Strategy

Positive Scores
Negative Score
Filtering-Ranking Strategy
Filtering-Ranking Strategy

- Positive Scores
- Negative Score
- Correct
- Over-predicted
- Missed
Filtering-Ranking Architecture

Filtering-Ranking Strategy

- Positive Scores
- Negative Score
- Correct
- Over-predicted
- Missed
Learning a Filtering-Ranking Model

- **Goal**: Learn the functions \((\text{start}_w, \text{end}_w, \text{score}_p)\) so as to maximize the \(F_1\) measure on the recognition of phrases

- **Desired behavior**:
  - **Start-End Filters**:
    - Do not block any correct phrase: very high recall
    - Block phrases that produce errors at the ranking stage
    - Block much incorrect phrases as possible
  - **Ranker**:
    - Separate between correct/incorrect structures
    - Forget about filtered phrases
Perceptron Learning at Global Level

- Following [Collins 02], we guide learning at global level:
  - Do not concentrate on individual errors of the learning functions
  - Instead, concentrate on errors at sentence level, after inference

- Key points:
  - Mistake-driven learning, a.k.a. Perceptron
  - Functions are learned together, visiting online training sentences
  - Errors are propagated from sentence-level, to phrase-level, to word-level
Filtering-Ranking Perceptron

• Configuration:
  - Feature extraction functions (given): \( \phi_w, \phi_p \)
  - Weight vectors (learned): \( w_S, w_E, w_p \)

• Algorithm: visit online sentence-structure pairs \((x, y)\):
  1. Infer the best phrase structure \( \hat{y} \) for \( x \)
  2. Identify errors and provide feedback to weight vectors.
     - We consider only errors at global level, comparing \( y \) and \( \hat{y} \):
       - Missed Phrases (those in \( y \setminus \hat{y} \))
       - Over-predicted Phrases (those in \( \hat{y} \setminus y \))
FR-Perceptron: Feedback on Missed phrases

If a phrase \((s, e)_k\) is missed, do promotion updates:

- if word \(s\) is not positive start for \(k\):
  \[ w_S = w_S + \phi_w(x, s, k) \]

- if word \(e\) is not positive end for \(k\):
  \[ w_E = w_E + \phi_w(x, e, k) \]

- if \((s, e)_k\) passes the filter (\(s/e\) are positive start/end for \(k\)):
  \[ w_p = w_p + \phi_p(x, y, (s, e)_k) \]
FR-Perceptron: Feedback on Over-Predicted phrases

If a phrase \((s, e)_k\) is over-predicted, do demotion updates:

- Give feedback to the ranker:
  \[
  w_p = w_p - \phi_p(x, y, (s, e)_k)
  \]

- If word \(s\) is not a correct start for \(k\):
  \[
  w_S = w_S - \phi_w(x, s, k)
  \]

- If word \(e\) is not a correct end for \(k\):
  \[
  w_E = w_E - \phi_w(x, e, k)
  \]
• Local predictions are corrected wrt. the global solution
Learning Feedback: Example

- Local predictions are **corrected** wrt. the global solution
Learning Feedback: Example

• Local predictions are **corrected** wrt. the global solution
Learning Feedback: Example

- Local predictions are **corrected** wrt. the global solution
Learning Feedback: Example

- Local predictions are **corrected** wrt. the global solution
Learning Feedback: Example

- Local predictions are **corrected** wrt. the global solution

- Local predictions that do not hurt globally are **not penalized**
Empirical validation of FR-Perceptron

- We perform a number of experiments to validate the behavior of FR-Perceptron

- Problem: Clause Identification, following CoNLL-2001 Shared Task:
  - One type of phrases: clauses
  - Hierarchical Structure
  - Training: $\sim 9,000$ sentences, $\sim 25,000$ clauses
  - Test: $\sim 1,700$ sentences, $\sim 4,900$ clauses
Empirical validation of FR-Perceptron

We compare four training strategies for the Filtering-Ranking model:

<table>
<thead>
<tr>
<th></th>
<th>type</th>
<th>$w$’s trained</th>
<th>R on F</th>
<th>penalty wrt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>local-VP</td>
<td>VP</td>
<td>separately</td>
<td>no</td>
<td>binary sign</td>
</tr>
<tr>
<td>local-SVM</td>
<td>SVM</td>
<td>separately</td>
<td>no</td>
<td>binary sign</td>
</tr>
<tr>
<td>global-VP</td>
<td>VP</td>
<td>together</td>
<td>yes</td>
<td>binary sign</td>
</tr>
<tr>
<td>FR-Perceptron</td>
<td>VP</td>
<td>together</td>
<td>yes</td>
<td>arg max</td>
</tr>
</tbody>
</table>
Empirical validation of FR-Perceptron
Overall Results

- Global training strategies perform better than local strategies
- Feedback after inference trains more effectively the recognizer
Empirical validation of FR-Perceptron Behavior of the Start-End Filter

We look at the performance of Start-End functions:

- Precision/Recall of Start-End
- How much the phrase space is reduced?
- What is the maximum achievable $F_1$ after the Filter?
Empirical validation of FR-Perceptron
Recall/Precision on Start words

- FR-Perceptron favors recall, others favor precision
- On End words, the same behavior is observed
FR-Perceptron maintains a high upper-bound $F_1$ for the ranking layer (left), and reduces the space of explored phrases (right).

Other methods are not sensitive to F-R interactions.
Empirical validation of FR-Perceptron
Does the Filter help in performance?

• We train the architecture without the Filter (UB-$F_1 = 100\%$):

![Graph showing performance over epochs]

• Filtering favors not only efficiency, but also global accuracy
Outline

• Introduction: Phrase Recognition
• Learning Methods for Text Analysis Tasks
• Filtering-Ranking Architecture
• Systems and Results on Syntactic-Semantic Parsing
• Conclusion and Future Research
Phrase Recognition in Syntactic-Semantic Analysis

- We apply the Filtering-Ranking architecture to three NLP recognition tasks.
- We follow the CoNLL Shared Task settings.

|                         | edition | nature     | |K| | structure     |
|-------------------------|---------|------------|---|---|---------------|
| NP Chunking             | 2000    | syn.       | 1 |   | sequential    |
| Syntactic Chunking      | 2000    | syn.       | 11|   | sequential    |
| Clause Identification   | 2001    | syn.       | 1 |   | hierarchical  |
| Semantic Role Labeling  | 2004    | syn./sem.  | 20|   | seq./hier.    |
General Details about the Systems

• Averaged predictions: better convergence, better accuracy

• Feature Extraction functions:
  ★ $\phi_w$: window-based representations
  ★ $\phi_p$: patterns of the phrase candidate
  ★ Both make use of predictions on the explored space:
    ▶ Inference might lead to a sub-optimal, but accuracy is better

• Polynomial kernels of degree two:
  ★ Much better than default linear predictions
  ★ No improvement with higher degrees
Application to Syntactic Chunking

- Sequential structures:
  - Chunks do not overlap
  - Chunks do not admit embedding

- Inference: Viterbi-like dynamic programming

- Following the CoNLL-2000 Shared Task. Trained for:
  - NP-Chunking: a single type of chunk, i.e. NP
  - Syntactic Chunking: eleven types of chunks (NP, VP, PP, ...)

- Many systems are evaluated on this benchmark data. All of them approach the problem as a tagging task.
## Syntactic Chunking - Results

<table>
<thead>
<tr>
<th>Reference</th>
<th>Technique</th>
<th>Precision</th>
<th>Recall</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Zhang 05]</td>
<td>SVD-ASO</td>
<td>94.57</td>
<td>94.20</td>
<td>94.39</td>
</tr>
<tr>
<td>[Zhang 02]</td>
<td>Winnow</td>
<td>94.28</td>
<td>94.07</td>
<td>94.17</td>
</tr>
<tr>
<td>[Kudo &amp; Matsumoto 01]</td>
<td>SVM voting</td>
<td>93.89</td>
<td>93.92</td>
<td>93.91</td>
</tr>
<tr>
<td>[Kudo &amp; Matsumoto 01]</td>
<td>SVM single</td>
<td>93.95</td>
<td>93.75</td>
<td>93.85</td>
</tr>
<tr>
<td>FRP-Chunker</td>
<td>FR-Perceptron</td>
<td>94.20</td>
<td>93.38</td>
<td>93.79</td>
</tr>
<tr>
<td>[Zhang 05]</td>
<td>SVD</td>
<td>93.83</td>
<td>93.37</td>
<td>93.60</td>
</tr>
<tr>
<td>[Zhang 02]</td>
<td>Winnow</td>
<td>93.54</td>
<td>93.60</td>
<td>93.57</td>
</tr>
<tr>
<td>[Kudo &amp; Matsumoto 00]</td>
<td>SVM</td>
<td>93.45</td>
<td>93.51</td>
<td>93.48</td>
</tr>
<tr>
<td>[van Halteren 00]</td>
<td>MBL&amp;WPD</td>
<td>93.13</td>
<td>93.51</td>
<td>93.32</td>
</tr>
<tr>
<td>[Tjong Kim Sang 00]</td>
<td>MBL voting</td>
<td>94.04</td>
<td>91.00</td>
<td>92.50</td>
</tr>
</tbody>
</table>

... + 8 shared task systems more
# NP Chunking - Results

<table>
<thead>
<tr>
<th>Reference</th>
<th>scope</th>
<th>Technique</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F&lt;sub&gt;1&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Zhang 05]</td>
<td>all</td>
<td>SVD-ASO</td>
<td>unav.</td>
<td>unav.</td>
<td>94.70</td>
</tr>
<tr>
<td>FRP-Chunker</td>
<td>all</td>
<td>FR-Perc.</td>
<td>94.55</td>
<td>94.37</td>
<td>94.46</td>
</tr>
<tr>
<td>[Kudo &amp; Matsumoto 01]</td>
<td>all</td>
<td>SVM voting</td>
<td>94.47</td>
<td>94.32</td>
<td>94.39</td>
</tr>
<tr>
<td>[Zhang 02]</td>
<td>all</td>
<td>Winnow</td>
<td>94.39</td>
<td>94.37</td>
<td>94.38</td>
</tr>
<tr>
<td>[Sha &amp; Pereira 03]</td>
<td>NP</td>
<td>CRF</td>
<td>unav.</td>
<td>unav.</td>
<td>94.38</td>
</tr>
<tr>
<td>FRP-Chunker</td>
<td>NP</td>
<td>FR-Perc.</td>
<td>94.69</td>
<td>93.98</td>
<td>94.33</td>
</tr>
<tr>
<td>[Kudo &amp; Matsumoto 01]</td>
<td>all</td>
<td>SVM single</td>
<td>94.54</td>
<td>94.09</td>
<td>94.32</td>
</tr>
<tr>
<td>[Sha &amp; Pereira 03]</td>
<td>NP</td>
<td>MM-VP</td>
<td>unav.</td>
<td>unav.</td>
<td>94.09</td>
</tr>
<tr>
<td>[Zhang 02]</td>
<td>all</td>
<td>Winnow</td>
<td>93.80</td>
<td>93.99</td>
<td>93.89</td>
</tr>
<tr>
<td>[Collins 02]</td>
<td>NP</td>
<td>MM-VP</td>
<td>unav.</td>
<td>unav.</td>
<td>93.53</td>
</tr>
<tr>
<td>. . .</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Application to Clause Identification

- A single type of phrases: clauses
- Clauses form hierarchical structures in a sentence
- Inference: CKY-like dynamic programming
- Following the CoNLL-2001 Shared Task
# Clause Identification - Results

<table>
<thead>
<tr>
<th>Reference</th>
<th>Technique</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR-Clauser</td>
<td>FR-Perceptron</td>
<td>88.17</td>
<td>82.10</td>
<td>85.03</td>
</tr>
<tr>
<td>[Carreras et al. 02]</td>
<td>AdaBoost class.</td>
<td>90.18</td>
<td>78.11</td>
<td>83.71</td>
</tr>
<tr>
<td>[Carreras &amp; Màrquez 01]</td>
<td>AdaBoost class.</td>
<td>84.82</td>
<td>78.85</td>
<td>81.73</td>
</tr>
<tr>
<td>[Molina &amp; Pla 01]</td>
<td>HMM</td>
<td>70.85</td>
<td>70.51</td>
<td>70.68</td>
</tr>
<tr>
<td>[Tjong Kim Sang 01]</td>
<td>Memory-based</td>
<td>76.91</td>
<td>65.22</td>
<td>70.58</td>
</tr>
<tr>
<td>[Patrick &amp; Goyal 01]</td>
<td>AdaBoost</td>
<td>73.75</td>
<td>64.56</td>
<td>68.85</td>
</tr>
<tr>
<td>[Dejean 01]</td>
<td>Theory Ref.</td>
<td>72.56</td>
<td>58.69</td>
<td>64.89</td>
</tr>
<tr>
<td>[Hammerton 01]</td>
<td>LSTM-NNet</td>
<td>55.81</td>
<td>49.49</td>
<td>52.46</td>
</tr>
</tbody>
</table>
Application to Semantic Role Labeling

- We follow the CoNLL-2004 Shared Task: puts SRL after partial parsing analysis (chunks and clauses)

- The SRL strategy looks for a hierarchy of arguments in a sentence, where:
  - Arguments are formed by joining elements found within clauses: words, chunks and inner clauses
  - An argument is related to number of verbs. These relations are labelled with semantic roles

- Other systems in literature approach the problem as a chunking task, recognizing arguments of different predicates independently
## Semantic Role Labeling - Results

<table>
<thead>
<tr>
<th>Reference</th>
<th>Technique</th>
<th>Precision</th>
<th>Recall</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Hacioglu et al. 04]</td>
<td>SVM</td>
<td>72.43</td>
<td>66.77</td>
<td>69.49</td>
</tr>
<tr>
<td>[Punyakanok et al. 04]</td>
<td>Winnow</td>
<td>70.07</td>
<td>63.07</td>
<td>66.39</td>
</tr>
<tr>
<td>FR-SRLabeler</td>
<td>FR-Perceptron</td>
<td>71.81</td>
<td>61.11</td>
<td>66.03</td>
</tr>
<tr>
<td>[Lim et al. 04]</td>
<td>Max-Entropy</td>
<td>68.42</td>
<td>61.47</td>
<td>64.76</td>
</tr>
<tr>
<td>[Park et al. 04]</td>
<td>SVM</td>
<td>65.63</td>
<td>62.43</td>
<td>63.99</td>
</tr>
<tr>
<td>[Higgins 04]</td>
<td>TBL</td>
<td>64.17</td>
<td>57.52</td>
<td>60.66</td>
</tr>
<tr>
<td>[van den Bosch et al. 04]</td>
<td>Memory-Based</td>
<td>67.12</td>
<td>54.46</td>
<td>60.13</td>
</tr>
<tr>
<td>[Kouchnir 04]</td>
<td>Memory-Based</td>
<td>56.86</td>
<td>49.95</td>
<td>53.18</td>
</tr>
<tr>
<td>[Baldewein et al. 04]</td>
<td>Max-Entropy</td>
<td>65.73</td>
<td>42.60</td>
<td>51.70</td>
</tr>
<tr>
<td>[Williams et. al 04]</td>
<td>TBL</td>
<td>58.08</td>
<td>34.75</td>
<td>43.48</td>
</tr>
</tbody>
</table>
Conclusion and Future Research

Outline

- Introduction: Phrase Recognition
- Learning Methods for Text Analysis Tasks
- Filtering-Ranking Architecture
- Systems and Results on Syntactic-Semantic Parsing
- Conclusion and Future Research
Main Contributions (i):
A Framework for Phrase Recognition

• We have studied the problem of recognizing phrase structures in a sentence.
  ☆ Many problems in NLP analysis can be casted as phrase recognition tasks

• We have discussed architectures based on learning and inference:
  ☆ Models based on decompositions at word and phrase level
  ☆ Incremental inference procedures
  ☆ Learning algorithms at local and global contexts
Main Contributions (ii):
Filtering-Ranking Perceptron

- A novel architecture for general phrase recognition:
  - Puts learning at phrase level
  - Uses filtering to reduce the solution space

- FR-Perceptron:
  - Global online learning, with ultra-conservative feedback
  - Experiments show that FR-Perceptron trains the functions of the architecture as word filters and phrase rankers
  - Analysis of convergence (see thesis)
Main Contributions (iii): Systems for Syntactic-Semantic analysis

- The Filtering-Ranking architecture is general and flexible
- We have developed Filtering-Ranking systems for three CoNLL Shared Tasks
- In all cases, we obtain results among the top in the state-of-the-art
- On Clause Identification, our system obtains the best results
Future Lines (i)

• From Greedy to Exact Inference in Global Learned Models
  ★ We would like to test the influence of different inference strategies, in models that exploit increasing levels of dependencies

• Learning Issues for FR-Perceptron
  ★ Gain theoretical understanding on the filtering-ranking interactions

• On Natural Language Tasks
  ★ Joint analysis of several layers: e.g., PoS tagging + Chunking
  ★ Increasing levels of syntax, from shallow, to partial, to full
Future Lines (ii)

- Introducing Knowledge
  - Learn on the top of a grammar-based exploration
- On Representations and Kernels
  - Look for more efficient kernel-based representations
Selected Publications (i): Learning Architectures

- Xavier Carreras, Lluís Màrquez and Jorge Castro
  “Filtering-Ranking Perceptron Learning for Partial Parsing”

- Xavier Carreras and Lluís Màrquez
  “Online Learning via Global Feedback for Phrase Recognition”

- Xavier Carreras, Lluís Màrquez, Vasin Punyakanok and Dan Roth
  “Learning and Inference for Clause Identification”
Selected Publications (ii): Shared Task Systems

- Xavier Carreras, Lluís Màrquez and Grzegorz Chrupala
  “Hierarchical Recognition of Propositional Arguments with Perceptrons”, CoNLL-2004

- Xavier Carreras, Lluís Màrquez and Lluís Padró

- Xavier Carreras, Lluís Màrquez and Lluís Padró
  “Learning a Perceptron-Based Named Entity Chunker via Online Recognition Feedback”, CoNLL-2003

- Xavier Carreras, Lluís Màrquez and Lluís Padró
  “Named Entity Extraction using Adaboost”, CoNLL-2002

- Xavier Carreras and Lluís Màrquez
Selected Publications (iii): Shared Task Organization

- Xavier Carreras and Lluís Màrquez
  “Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling”

- Xavier Carreras and Lluís Màrquez
  “Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling”
Gràcies!