Recognizing Phrases with Filtering-Ranking Perceptron

Xavier Carreras and Lluís Màrquez

Centre de Tecnologies i Aplicacions del Llenguatge i la Parla
UNIVERSITAT POLITÈCNICA DE CATALUNYA
Outline

- Phrase Recognition
- Filtering-Ranking Strategy
- FR-Perceptron
- Experiments
Phrase Recognition Problems

- Following CoNLL Shared Tasks:
  - 2000: Chunking of Syntactic Base Phrases
  - 2001: Identification of Syntactic Clauses
  - 2004: Recognition of Semantic Roles

- Also, Full Syntactic Parsing
### Phrase Recognition in CoNLL: Example

<table>
<thead>
<tr>
<th>WORDS</th>
<th>PoS</th>
<th>CHUNKS</th>
<th>CLAUSES</th>
<th>NE</th>
<th>&lt;-- SEMANTIC ROLES ---------------&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>DT</td>
<td>B-NP</td>
<td>(S*</td>
<td>0</td>
<td>(A0*</td>
</tr>
<tr>
<td>San</td>
<td>NNP</td>
<td>I-NP</td>
<td>*</td>
<td>B-ORG</td>
<td>*</td>
</tr>
<tr>
<td>Francisco</td>
<td>NNP</td>
<td>I-NP</td>
<td>*</td>
<td>I-ORG</td>
<td>*</td>
</tr>
<tr>
<td>Examiner</td>
<td>NNP</td>
<td>I-NP</td>
<td>*</td>
<td>I-ORG</td>
<td>*A0)</td>
</tr>
<tr>
<td>issued</td>
<td>VBD</td>
<td>B-VP</td>
<td>*</td>
<td>0</td>
<td>(V:ISSUE *V)</td>
</tr>
<tr>
<td>a</td>
<td>DT</td>
<td>B-NP</td>
<td>*</td>
<td>0</td>
<td>(A1*</td>
</tr>
<tr>
<td>special</td>
<td>JJ</td>
<td>I-NP</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>edition</td>
<td>NN</td>
<td>I-NP</td>
<td>*</td>
<td>0</td>
<td>*A1)</td>
</tr>
<tr>
<td>around</td>
<td>IN</td>
<td>B-PP</td>
<td>*</td>
<td>0</td>
<td>(AM-TMP*</td>
</tr>
<tr>
<td>noon</td>
<td>NN</td>
<td>B-NP</td>
<td>*</td>
<td>0</td>
<td>*AM-TMP)</td>
</tr>
<tr>
<td>yesterday</td>
<td>NN</td>
<td>B-NP</td>
<td>*</td>
<td>0</td>
<td>(AM-TMP*AM-TMP)</td>
</tr>
<tr>
<td>that</td>
<td>WDT</td>
<td>B-NP</td>
<td>(S*</td>
<td>0</td>
<td>(C-A1*</td>
</tr>
<tr>
<td>was</td>
<td>VBD</td>
<td>B-VP</td>
<td>(S*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>filled</td>
<td>VBN</td>
<td>I-VP</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>entirely</td>
<td>RB</td>
<td>B-ADVP</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>with</td>
<td>IN</td>
<td>B-PP</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>earthquake</td>
<td>NN</td>
<td>B-NP</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>news</td>
<td>NN</td>
<td>I-NP</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>and</td>
<td>CC</td>
<td>I-NP</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>information</td>
<td>NN</td>
<td>I-NP</td>
<td>*S)S)</td>
<td>0</td>
<td>*C-A1)</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>0</td>
<td>*S)</td>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>
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 some constraints:
  - 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 Recognition: schematic view
Hierarchical Phrase Recognition: schematic view
Observation 1: Huge Output Space

Output space is exponential: Parsing strategy required.
Observation 2: Recursive Structures

Desirable to put learning in high-order level.
Recognizing Structures: General Approach

- Decompose global problems into *tractable* and *learnable* subproblems:
  - Chunking: as tagging, with BIO, Open-Close, . . .
  - Hierarchies: CKY-style decisions, shift-reduce, . . .

- Design decoder algorithm which *infers* global solution given the local predictions: greedy, beam search, dynamic programming, . . .

- Learn local functions for each local subproblem.
Recognizing Structures: Learning

- Local learning: each local function is trained separately, as a (binary) classification algorithm.
  - Good understanding on learning classifiers.
  - but local accuracies don’t guarantee global accuracy (after inference).
  - that is, a local classification behavior might not be the optimal within the decoder.

- Global learning: train the Recognizer as a composed function for the final problem.
Outline

• Phrase Recognition

• Filtering-Ranking Strategy

• FR-Perceptron

• Experiments
Filtering-Ranking Strategy (i)

1. Generate phrase candidates with Start-End functions.
2. Score each phrase candidate.
3. Build the hierarchy with best phrases.
Filtering-Ranking Strategy (ii)
Filtering-Ranking Strategy (ii)
Filtering-Ranking Strategy (ii)
Filtering-Ranking Strategy (ii)

[Diagram showing a hierarchical structure with positive and negative scores indicated by different symbols]
Filtering-Ranking Strategy (ii)

Positive Scores

Negative Score
Filtering-Ranking Strategy (ii)
Filtering-Ranking Strategy (ii)
Filtering-Ranking Strategy (ii)

Positive Scores
Negative Score
Filtering-Ranking Strategy (ii)
Filtering-Ranking Strategy (ii)

Positive Scores

Negative Score
Filtering-Ranking Strategy (ii)
Filtering-Ranking Strategy (ii)
Filtering-Ranking Strategy (ii)
Filtering-Ranking Strategy (ii)

- **Positive Scores**
- **Negative Score**
- **Correct**
- **Over-predicted**
- **Missed**
Filtering-Ranking Strategy (ii)

- Positive Scores
- Negative Score
- Correct
- Over-predicted
- Missed
Filtering-Ranking Strategy (iii)

\( \mathcal{Y} \): solution space, i.e. set of all coherent phrase sets.
\( \mathcal{Y}_{SE} \): practical solution space, filtered at word level.

\[
R(x) = \arg \max_{y \in \mathcal{Y}_{SE}} \sum_{(s,e)_k \in y} \text{score}((s,e)_k, x, y_{s:e})
\]

\[
\mathcal{Y}_{SE} = \{ y \in \mathcal{Y} \mid \forall (s,e)_k \in y \; \text{start}^k(s) \land \text{end}^k(e) \}
\]

- Sequential case: \( O(n^2) \) Dynamic Prog. search
- Hierarchical case: \( O(n^3) \) Dynamic Prog. search
Outline

- Phrase Recognition
- Filtering-Ranking Strategy
- FR-Perceptron
- Experiments
Learning Challenges

- Learn all functions \((\text{start}^k, \text{end}^k, \text{score}^k)\) so as to maximize the \(F_1\) measure on the recognition of phrases.

- Start-End:
  - As filters, rather than classifiers.
  - They define the input space to the score functions.

- Score functions:
  - The negative space is too big \(\sim O(n^2)\).
  - We need to know about Start-End behavior.
  - As Rankers, rather than Classifiers.
Score as Classifier: 0-1 Phrases
Score as Classifier: 0-1 Phrases
Score as Ranker: Pairwise Order Relations
Score as Ranker: Pairwise Order Relations

$$((( \bullet \ (\bullet)) \ (\bullet \ (\bullet)) \ \bullet))$$
Score as Ranker: Pairwise Order Relations
Score as Ranker: Pairwise Order Relations

\[
\begin{align*}
&\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\end{array} \\
&\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\end{array} \\
&\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\end{array} \\
&\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\end{array} \\
&\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\end{array}
\end{align*}
\]
Score as Ranker: Pairwise Order Relations
Counterexample

- Pairwise Order Relations are not enough:

\[ \text{score}(b) > \text{score}(a) \text{ and } \text{score}(b) > \text{score}(c) \]
\[ \text{but } \text{score}(a) + \text{score}(c) > \text{score}(b) \]
Score as Ranker: Sentence Level Learning

• We follow Collins’ approach (EMNLP 2002):
  Guide learning at the sentence-level

• Two key points:
  ★ Mistake-driven learning, a.k.a. Perceptron
  ★ Learn from the output of the inference

• Our contribution (Carreras and Màrquez, NIPS 2003):
  We propagate mistakes to the filtering layer
Filtering-Ranking Perceptron

- All functions are learned together, while visiting online training sentences.

- Algorithm: Given a sentence:
  1. Predict the phrase hierarchy.
  2. Identify errors and provide feedback.
     We consider only errors at global level:
     - Missed Phrases
     - Over-predicted Phrases
Feedback on Missed phrases

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

- **Boundaries:**
  
  \[
  \begin{align*}
  &\text{if } (\mathbf{w}_S \cdot \phi_w(x_s) \leq 0) \text{ then } \mathbf{w}_S = \mathbf{w}_S + \phi_w(x_s) \\
  &\text{if } (\mathbf{w}_E \cdot \phi_w(x_e) \leq 0) \text{ then } \mathbf{w}_E = \mathbf{w}_E + \phi_w(x_e)
  \end{align*}
  \]

- **Score** :
  
  \[
  \begin{align*}
  &\text{if } (\mathbf{w}_S \cdot \phi_w(x_s) > 0 \land \mathbf{w}_E \cdot \phi_w(x_e) > 0) \text{ then } \\
  &\mathbf{w}_k = \mathbf{w}_k + \phi_p(s, e)
  \end{align*}
  \]
Feedback on Over-Predicted phrases

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

- **Score:**
  \[ w_k = w_k - \phi_p(s, e) \]

- **Boundaries:**
  - if \((x_s \notin \text{STARTS}(y^*))\) then \(w_S = w_S - \phi_w(x_s)\)
  - if \((x_e \notin \text{ENDS}(y^*))\) then \(w_E = w_E - \phi_w(x_e)\)
Learning Feedback

Correct
Missed
Over-predicted

+ Positive Scores
- Negative Score

Correct
Over-predicted
Missed
Learning Feedback

Correct
Missed
Over-predicted

Positive Scores
Negative Score

Correct
Over-predicted
Missed
Learning Feedback

- Correct
- Over-predicted
- Missed

Positive Scores
Negative Score
Learning Feedback

- Correct
- Over-predicted
- Missed

Positive Scores
Negative Score
Learning Feedback

- Correct
- Missed
- Over-predicted

+ Positive Scores
- Negative Score

Correct
Over-predicted
Missed
Local errors which do not hurt global performance are not penalized. (Crammer & Singer, 2003), (Har-Peled et al., 2002)
On Representation

- Features on words: standard window-based features.
- Features on phrases:
  - Windows at start/end boundaries.
  - Patterns of the internal structure.
On Representation: internal structure

When visiting a phrase, the internal structure is already computed:

Exploitation through patterns and constraints, via:
- Linguistically-motivated, grammar-based.
- Kernels, ie. exhaustive exploration of the structure.
Final Architecture

- Exploration, incrementality, local decisions, inference.
- Learning Functions at word and phrase level.
- Learning Constraints at sentence level:
  - All functions learned together, capturing interactions.
  - Functions are modeled so as to optimize its behavior within the parser, i.e. as filters and rankers.
- Practical important tricks:
  - Feature expansion with polynomial kernels, \( d = 2 \).
  - Perceptron with averaged predictions.
Outline

• Phrase Recognition
• Filtering-Ranking Strategy
• FR-Perceptron
• Experiments
Experiments on Clause Identification

Given the same model (F&R) and the same learner (VP), we compare three training strategies:

**CB-VP** Classification feedback (0/1 Loss)
Each function trained separately, batch.

**CO-VP** Classification feedback (0/1 Loss)
Functions learned together, online.

**F&R-VP** Conservative feedback (wrt. arg max)
Functions learned together, online.
## Training functions in batch setting

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Generation</th>
<th>#Neg.</th>
<th>Precision</th>
<th>Recall</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBVP</td>
<td>goldSE</td>
<td>26,374</td>
<td>83.84</td>
<td>80.55</td>
<td>82.16</td>
</tr>
<tr>
<td>SVM</td>
<td>goldSE</td>
<td>26,374</td>
<td>84.31</td>
<td>82.83</td>
<td>83.57</td>
</tr>
<tr>
<td>SVM</td>
<td>θ = 0</td>
<td>28,165</td>
<td>88.14</td>
<td>82.85</td>
<td>85.41</td>
</tr>
<tr>
<td>SVM</td>
<td>θ = −0.3</td>
<td>28,747</td>
<td>88.34</td>
<td>82.76</td>
<td>85.46</td>
</tr>
<tr>
<td>SVM</td>
<td>θ = −0.5</td>
<td>29,145</td>
<td>88.46</td>
<td>82.61</td>
<td>85.43</td>
</tr>
<tr>
<td>SVM</td>
<td>θ = −0.7</td>
<td>29,610</td>
<td>88.54</td>
<td>82.48</td>
<td>85.41</td>
</tr>
<tr>
<td>SVM</td>
<td>θ = −0.9</td>
<td>30,432</td>
<td>88.91</td>
<td>82.58</td>
<td>85.63</td>
</tr>
<tr>
<td>SVM</td>
<td>θ = −1.0</td>
<td>59,498</td>
<td>91.12</td>
<td>81.27</td>
<td>85.91</td>
</tr>
<tr>
<td>SVM</td>
<td>θ = −1.1</td>
<td>97,101</td>
<td>91.49</td>
<td>80.80</td>
<td>85.82</td>
</tr>
<tr>
<td>SVM</td>
<td>θ = −1.2</td>
<td>120,856</td>
<td>91.33</td>
<td>80.51</td>
<td>85.58</td>
</tr>
<tr>
<td>SVM</td>
<td>θ = −1.5</td>
<td>240,463</td>
<td>92.31</td>
<td>78.01</td>
<td>84.56</td>
</tr>
</tbody>
</table>
Experiments on Clause Identification
## Results on Clause Identification

### CoNLL-2001

<table>
<thead>
<tr>
<th>Method</th>
<th>T</th>
<th>prec.</th>
<th>recall</th>
<th>$F_{\beta=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>goldSE CB-VP</td>
<td>8</td>
<td>82.22</td>
<td>78.09</td>
<td>80.10</td>
</tr>
<tr>
<td>goldSE SVM</td>
<td>-</td>
<td>83.19</td>
<td>80.00</td>
<td>81.57</td>
</tr>
<tr>
<td>CO-VP</td>
<td>19</td>
<td>89.25</td>
<td>77.62</td>
<td>83.03</td>
</tr>
<tr>
<td>learnedSE SVM</td>
<td></td>
<td>91.12</td>
<td>81.27</td>
<td>85.91</td>
</tr>
<tr>
<td>FR-Perceptron</td>
<td>20</td>
<td>88.17</td>
<td><strong>82.10</strong></td>
<td><strong>85.03</strong></td>
</tr>
<tr>
<td>AdaBoost (1)</td>
<td>–</td>
<td><strong>90.18</strong></td>
<td>78.11</td>
<td>83.71</td>
</tr>
</tbody>
</table>

(1) (Carreras, Màrquez, Punyakanok and Roth, ECML’02)
Experiments on Clause Identification
Precision/Recall on Start words

On End words, a similar behavior is observed.
Experiments on Clause Identification
Explored Phrases/Upper Bound $F_1$
Experiments on Clause Identification
Behavior of the Score Function

Three special scenarios in which to experiment with the score function:

- **GoldSE** : using the gold filters to train and test.
- **No Filtering** : all possible phrases are candidates.
- **FR Filters** : the best obtained filters.
Experiments on Clause Identification
Score Function above Gold Filters

goldSE FR-Perceptron
goldSE CO-VP
goldSE SVM
Experiments on Clause Identification
Score Function without Filters

![Graph showing the global F Measure vs Number of Epochs for no filter and FR-Perceptron models. The graph illustrates the improvement in F Measure as the number of epochs increases. The blue line represents the FR-Perceptron model, while the black line represents the no filter model. The F Measure values range from 82.5 to 88.8.]
Experiments on Clause Identification
Score Function above FR Learned Filters

![Graph showing the performance of FR-Perceptron, CO-VP, and SVM over epochs.](image)
Experiments on Clause Identification
Practical Problem: number of dual vectors

![Graph showing the number of different vectors over Number of Epochs for different methods: FR-Perc., CG-VP, goldSE CB-VP, learnedSE SVM, goldSE SVM. The graph illustrates the trend of vector growth as the number of epochs increases.]
## Results on Chunking

11 types – CoNLL-2000

<table>
<thead>
<tr>
<th>technique</th>
<th>prec.</th>
<th>recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Kudo &amp; M. 01)</td>
<td>SVM voting</td>
<td>93.89</td>
<td>93.92</td>
</tr>
<tr>
<td>(Kudo &amp; M. 01)</td>
<td>SVM single</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>FRP-Chunker</td>
<td>F&amp;R VP</td>
<td>94.20</td>
<td>93.38</td>
</tr>
<tr>
<td>(Zhang 02)</td>
<td>Winnow</td>
<td>93.54</td>
<td>93.60</td>
</tr>
<tr>
<td>BI-Chunker</td>
<td>greedy VP</td>
<td>92.83</td>
<td>92.21</td>
</tr>
</tbody>
</table>
## Results on NP Chunking – CoNLL-2000

<table>
<thead>
<tr>
<th>Reference</th>
<th>S</th>
<th>Technique</th>
<th>Prec.</th>
<th>Rec.</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRP-Chunker</td>
<td>all</td>
<td>F&amp;R VP</td>
<td>94.55</td>
<td>94.37</td>
<td>94.46</td>
</tr>
<tr>
<td>(Kudo &amp; M.)</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; Per. 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>F&amp;R VP</td>
<td>94.69</td>
<td>93.98</td>
<td>94.33</td>
</tr>
<tr>
<td>(Kudo &amp; M.)</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; Per. 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 02c)</td>
<td>NP</td>
<td>MM-VP</td>
<td>unav.</td>
<td>unav.</td>
<td>93.53</td>
</tr>
<tr>
<td>BI-Chunker</td>
<td>all</td>
<td>greedy-VP</td>
<td>92.83</td>
<td>92.21</td>
<td>92.52</td>
</tr>
</tbody>
</table>
Semantic Role Labeling

- Recognize predicate arguments and label them according to some scheme (ie. PropBank).

- Same architecture applies:
  - argument = phrase to be recognized
  - We build hierarchies of arguments.
  - Now, the score of a phrase is not only related to its internal structure, but also to a number of predicates.

- 3rd position in CoNLL-2004, not bad!
Conclusions

• Flexible learning architecture for recovering phrases:
  ✿ The parsing strategy defines the dependencies to be exploited.
  ✿ With Perceptron, the parser functions are easily adapted to work within the parser.

• Future lines:
  ✿ Analysys: understand the global margins.
  ✿ Re-consider NLP pipeline.
Thanks!