Clause Identification

Antony R. Neu
neua@informatik.uni-freiburg.de

Efficient Natural Language Processing
Lehrstuhl für Algorithmen und Datenstrukturen
Lehrstuhlinhaberin: Prof. Hannah Bast
Universität Freiburg
Motivation

- POS Tagging
  - Words have a Part-Of-Speech tag
- Text chunking
  - Which words belong together
  - Not embedded or recursive
- Clause identification:
  - E.g. relative clauses
  - Recursive problem
  - Applications: text to speech
Overview

- Introduction
  - Problem definition
  - Applications
- Solutions
  - Rule-based approaches
  - Machine-Learning-based approaches
    - Demo
  - Hybrid systems
- Summary
Clause: *Group of words containing a subject and a predicate. Subject may be implicite.*

Latin: claudere: close, conclude, enclose

Two types:
- Independent clause: sentence
- Dependent clause:
  - sentence-like structure within a sentence
  - cannot exist without a main clause

Examples:
1. "*The man, who is walking over the street, is my father.*" (DC/IC)
2. "*He went to school and she went to work.*" (IC/IC)
Introduction - Definition

- **clause vs. phrase**: phrase has no subject and predicate
- Examples:
  - a known writer
  - an entirely new culture
  - when they learn how to solve their problems with wikis
- Debatable definitions
Task to solve

- Clause identification (also: clause splitting, clause boundary recognition)
- Shared Task of CoNLL-2001 (Computational Natural Language Learning)
  - Find start and ending point of a clause
  - Determine clause structure of the sentence
  - Type of clause, e.g. relative clause, temporal clause is ignored
- Examples:
  - ((The space shuttle Atlantis blasted into orbit from Cape Canaveral) and (its crew launched the Galileo space probe on a flight to the planet Jupiter).)
  - (The deregulation of railroads and trucking companies (that (began in 1980)) enabled (shippers to bargain for transportation).)
Applications

- Text-To-Speech systems
- Machine-Translation
- Question-Answering
- Preprocessing for bilingual alignment
- Brokkoli?
"You will start to see shows where viewers program the program."

- **Chunked:**
  
  (NP You) (VP will start to see) (NP shows) (ADVP where) (NP viewers) (VP program) (NP the program)

- **Clauses:**
  
  (S You will start to see shows (S where (S viewers program the program ))) .

- **Nevertheless:**
  
  - Fuzzy transitions
  
  - Some chunkers provide simple clause identification
CI vs. full parsing

- Clause identification as intermediate step (Ejerhed '90)
- Form of shallow parsing
- Full parsing: better precision
- Why not extract clauses from full parse?
  - Classification frameworks:
    - Faster (e.g. needed for question answering)
    - Easier to implement
    - More easily portable to new languages
Solutions and Implementations

- Rule-Based-Systems (1990s)
- Machine Learning based systems (2000s)
- Hybrid systems (late 2000s)
Rule-based systems

- Clauses identified by predefined rules
- POS tags and/or chunk tags are taken into consideration
- Disadvantages:
  - Human work needed
  - Not easily adaptable to other languages
- Example:

-1: <VP>
0: <NP>
1: ,
2: say (past o. Present)
3: <NP>

Mark 0 as end of clause boundary.
Rule-based systems

- **Ejerhed '96:**
  - Only independent clauses identified
  - Starts and end identified
    - *(There was something true in that) (what he said).*
  - Regular expressions and stochastic approach
    - DL_MAD XX => DL_MAD <c> XX
  - DL_MAD: major delimiter (., ?, !)

- **Papageorgiou '97**
  - Addresses embedded clauses
  - Inspired by Abbney's Cascaded Analysis of Syntactic Structure (CASS) parser ('91) (Full parser)
  - Text is tokenized and tagged (Brill tagger)
  - Clause tag marking module
    - What marks the clause, e.g. ”if” or ”as if”
  - Partial parsing generates clause structure
Rule-based systems

- Leffa '98:
  - Considers POS tags and **valence** of verb
  - Valence: How many other words does the verb bind?
    - 0: (It) is raining. (not a real subject)
    - 1: The dog runs. (a subject)
    - 2: I hate maths. (a subject and an object)
  - Read sentence left to right and mark clause initiators/terminators.
  - Clauses are segmented and processed
  - Valence is considered
    - (I know (when I have time).)
    - (I work (when (I have time))).
Evaluation

- Not identical corpora used for evaluation
- No standard
- Interpretation: good results

<table>
<thead>
<tr>
<th>Author</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_{β=1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ejerhed (1988)</td>
<td>87.01</td>
<td>98.89</td>
<td>92.57</td>
</tr>
<tr>
<td></td>
<td>95.07</td>
<td>96.01</td>
<td>95.54</td>
</tr>
<tr>
<td>Ejerhed (1996)</td>
<td>100.00</td>
<td>95.80</td>
<td>97.85</td>
</tr>
<tr>
<td></td>
<td>98.80</td>
<td>90.70</td>
<td>94.58</td>
</tr>
<tr>
<td>Papageorgiou (1997)</td>
<td>95.44</td>
<td>93.06</td>
<td>94.23</td>
</tr>
<tr>
<td>Leffa (1998)</td>
<td>–</td>
<td>95.00</td>
<td>–</td>
</tr>
<tr>
<td>Sang &amp; Déjean (2001)</td>
<td>84.82</td>
<td>73.28</td>
<td>78.63</td>
</tr>
<tr>
<td>Sang &amp; Déjean (2001)</td>
<td>98.44</td>
<td>31.48</td>
<td>47.71</td>
</tr>
</tbody>
</table>

Source: Master Thesis, Benjamin Hachey, University Edinburgh
Used for CoNLL-2001 shared task
  - Baseline: Assign Clause start and end at start and end of each sentence

Basic idea:
  - Systems learn on a specific training set.
  - Classification problem (see text chunking)
  - Features are considered, e.g. the last 3 words (POS and chunk tags)
  - Decision: Is this word the beginning of a clause?
Implementations

- Carreras and Marquez (shown today)
  - Boosted decision trees
  - Perceptrons (neural networks)
  - Both concepts outperform all other participants

- Others:
  - Short-Term Memory based
  - Conditional Random Fields
  - Hidden Markov Model
Results of CoNLL-2001 shared task

<table>
<thead>
<tr>
<th>test cor</th>
<th>precision</th>
<th>recall</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CM03]</td>
<td>87.99%</td>
<td>81.01%</td>
<td>84.36</td>
</tr>
<tr>
<td>[CMPR02]</td>
<td>90.18%</td>
<td>78.11%</td>
<td>83.71</td>
</tr>
<tr>
<td>[CM01]</td>
<td>84.82%</td>
<td>78.85%</td>
<td>81.73</td>
</tr>
<tr>
<td>[MP01]</td>
<td>70.85%</td>
<td>70.51%</td>
<td>70.68</td>
</tr>
<tr>
<td>[TKS01]</td>
<td>76.91%</td>
<td>65.22%</td>
<td>70.58</td>
</tr>
<tr>
<td>[PG01]</td>
<td>73.75%</td>
<td>64.56%</td>
<td>68.85</td>
</tr>
<tr>
<td>[Dej01]</td>
<td>72.56%</td>
<td>58.69%</td>
<td>64.89</td>
</tr>
<tr>
<td>[Ham01]</td>
<td>55.81%</td>
<td>49.49%</td>
<td>52.46</td>
</tr>
<tr>
<td>baseline</td>
<td>98.44%</td>
<td>33.88%</td>
<td>50.41</td>
</tr>
</tbody>
</table>
Learning algorithm (modified Adaboost) is given large number of binary simple features

4 feature types are used:
- Word window: Surrounding sequence of words with their POS tags
- Chunk window: Surrounding chunk tags of a word
- Sentence patterns from word a to b:
  - All occurrences of punctuation marks, relative pronouns, conjunctions, the word "that" with its POS tag and VP chunks between a and b
Carreras & Marquez systems

- Sentence features:
  - Number of occurrences VP, WP (pronoun), WP$, punctuation mark, beginning/end of clauses, the word "that" to the left and right hand side of the word
  - Window size was tuned to 3

- Filtering-Ranking Perceptron Learning for Partial Parsing (2005)
  - Similar Features to CM'01
  - Perceptrons are used instead of Adaboost
  - Implementation: Phreco
Phreco - Demo

- Uses perceptrons to recognize chunks or clauses
- Carreras' dissertation
- A demo is shown
- File with 11 sentences
Run times (45 000 words):

- Test data set A: 44min 33s (743 KB, 2012 sentences, 1.3s per sentence)
- Test data set B: 39min 33s (623 KB, 1671 sentences, 1.4s per sentence)

Over 1 second per sentence

Excluding tagging and chunking time
# Phreco - Profiling

<table>
<thead>
<tr>
<th>%Time</th>
<th>ExclSec</th>
<th>CumulS</th>
<th>#Calls</th>
<th>sec/call</th>
<th>Csec/c</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>98.4</td>
<td>2696.</td>
<td>2696.</td>
<td>75439</td>
<td>0.0357</td>
<td>0.0357</td>
<td>ml::vperceptron_classify</td>
</tr>
<tr>
<td>0.54</td>
<td>14.68</td>
<td>14.68</td>
<td>133684</td>
<td>0.0000</td>
<td>0.0000</td>
<td>PHRECO::phrase_set::top_phrases</td>
</tr>
<tr>
<td>0.26</td>
<td>7.057</td>
<td>7.057</td>
<td>75439</td>
<td>0.0001</td>
<td>0.0001</td>
<td>mapping::map_features</td>
</tr>
<tr>
<td>0.15</td>
<td>4.048</td>
<td>4.681</td>
<td>89564</td>
<td>0.0000</td>
<td>0.0001</td>
<td>PHRECO::clausefex::word_window_features</td>
</tr>
<tr>
<td>0.12</td>
<td>3.284</td>
<td>813.73</td>
<td>2012</td>
<td>0.0016</td>
<td>0.4044</td>
<td>PHRECO::frclauer::optimal_hierarchhy</td>
</tr>
<tr>
<td>0.11</td>
<td>3.105</td>
<td>3.105</td>
<td>136515</td>
<td>0.0000</td>
<td>0.0000</td>
<td>PHRECO::clausefex::sentence_counts</td>
</tr>
<tr>
<td>0.11</td>
<td>3.102</td>
<td>5.668</td>
<td>122628</td>
<td>0.0000</td>
<td>0.0000</td>
<td>PHRECO::clausefex::sentence_counts</td>
</tr>
<tr>
<td>0.09</td>
<td>2.586</td>
<td>2.586</td>
<td>89564</td>
<td>0.0000</td>
<td>0.0000</td>
<td>PHRECO::clausefex::chunk_window_features</td>
</tr>
<tr>
<td>0.09</td>
<td>2.434</td>
<td>798.19</td>
<td>699017</td>
<td>0.0000</td>
<td>0.0011</td>
<td>PHRECO::frclauer::predict_phrase</td>
</tr>
<tr>
<td>0.08</td>
<td>2.176</td>
<td>1908.7</td>
<td>61314</td>
<td>0.0000</td>
<td>0.0311</td>
<td>PHRECO::sefilter::predict_word_signal</td>
</tr>
<tr>
<td>0.06</td>
<td>1.614</td>
<td>1.614</td>
<td>183942</td>
<td>0.0000</td>
<td>0.0000</td>
<td>PHRECO::clausefex::signal_window_features</td>
</tr>
<tr>
<td>0.05</td>
<td>1.450</td>
<td>1.450</td>
<td>3</td>
<td>0.4833</td>
<td>0.4833</td>
<td>ml::vperceptron_read_from_file</td>
</tr>
<tr>
<td>0.05</td>
<td>1.424</td>
<td>1928.4</td>
<td>2012</td>
<td>0.0007</td>
<td>0.9585</td>
<td>PHRECO::sefilter::start_end_filtering</td>
</tr>
<tr>
<td>0.04</td>
<td>1.026</td>
<td>2743.0</td>
<td>2012</td>
<td>0.0005</td>
<td>1.3634</td>
<td>PHRECO::frclauer::process_sentence</td>
</tr>
<tr>
<td>0.03</td>
<td>0.827</td>
<td>0.827</td>
<td>75439</td>
<td>0.0000</td>
<td>0.0000</td>
<td>ml::new_fvinput</td>
</tr>
</tbody>
</table>
Hybrid Systems

- Recent works based on previous ML and rule based works
- Basic idea:
  - Use machine learning approach
  - Resolve errors with rules
- Papers:
  - Sundar et. al. '08 (best values)
  - Also: Nguyen'07
Sundar et al 2008

- Uses Conditional random fields as ML approach
- Features used (word windows of 5):
  - Word itself
  - POS tag
  - Chunk tag
  - Can linguistic rules be applied? (used later)
Error analyzer and linguistic rules:

- Find wrongly marked clause boundaries
- 'Error patterns' are used for identification, e.g. unbalanced starts and endings of clauses
- Linguistic rules are applied to correct errors (inside out)

Example rule:

-1: <VP>
0: <NP>
1: <VP infinitive>  
Mark position 0 as clause boundary start.
### Sundar et al 2008 - Benchmark

<table>
<thead>
<tr>
<th>S.No</th>
<th>System</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
<th>Fmeasure (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CRFs</td>
<td>83.68%</td>
<td>78.65%</td>
<td>81.08%</td>
</tr>
<tr>
<td>2</td>
<td>CRFs with linguistic Rules</td>
<td>92.06%</td>
<td>87.89%</td>
<td>89.04%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S. No</th>
<th>References</th>
<th>Techniques</th>
<th>Precision</th>
<th>Recall</th>
<th>F measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Our method</td>
<td>CRFs + linguistic Rules</td>
<td>92.06%</td>
<td>87.89%</td>
<td>89.04%</td>
</tr>
<tr>
<td>2</td>
<td>Carreras et al. 05</td>
<td>FR-Perceptron</td>
<td>88.17%</td>
<td>82.10%</td>
<td>85.03%</td>
</tr>
<tr>
<td>3</td>
<td>Vinh Van Nyugen et al 07</td>
<td>CRFs</td>
<td>90.01%</td>
<td>78.98%</td>
<td>84.09%</td>
</tr>
<tr>
<td>4</td>
<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>5</td>
<td>Carreras et al. 01</td>
<td>AdaBoost class</td>
<td>84.82%</td>
<td>78.85%</td>
<td>81.73%</td>
</tr>
<tr>
<td>6</td>
<td>Monila and Pla 01</td>
<td>HMM</td>
<td>70.85%</td>
<td>70.51%</td>
<td>70.68%</td>
</tr>
</tbody>
</table>
Summary

- Time-expensive intermediate task
- Not a lot of open-source implementations available
  - Lots of POS taggers and chunkers
  - Lots of Full parsers, role labelers etc.
  - Missing: intermediate task
- Hybrid systems seem to be an interesting approach
Sources

- **Overviews:**

- **References:**
Sources

- References: