Using Shallow Processing for Deep Parsing

Sandra Kübler
kuebler@sfs.uni-tuebingen.de

Seminar für Sprachwissenschaft
Eberhard-Karls-Universität Tübingen
Germany
Task Description

- shallow parsing, i.e. chunk parsing: assigning non-recursive phrasal and clausal boundaries, not including postmodification
Task Description

- **shallow parsing, i.e. chunk parsing**: assigning non-recursive phrasal and clausal boundaries, not including postmodification

- **deep parsing**: adding recursive syntactic structure as well as function argument structure
Task Description

- **Shallow parsing, i.e. chunk parsing**: assigning non-recursive phrasal and clausal boundaries, not including postmodification.

- **Deep parsing**: adding recursive syntactic structure as well as function argument structure.

- Task: take a (German) sentence as input and automatically assign.
Task Description

- **shallow parsing, i.e. chunk parsing:** assigning non-recursive phrasal and clausal boundaries, not including postmodification

- **deep parsing:** adding recursive syntactic structure as well as function argument structure

- task: take a (German) sentence as input and automatically assign

- use chunk parse as guiding information
Task Description

- **shallow parsing**, i.e. **chunk parsing**: assigning non-recursive phrasal and clausal boundaries, not including postmodification

- **deep parsing**: adding recursive syntactic structure as well as function argument structure

Task: take a (German) sentence as input and automatically assign

- use chunk parse as guiding information

- not only add recursive and functional information, but also deepen phrase structure
Preprocessing and Chunk Parsing

Steps consists of:

- POS tagging – TnT (Thorsten Brants)
- Tagfixing – making tags for specific words more informative e.g. abends (in the evening) ADV
- Chunkparsing – CASS (Steve Abney)

Using Shallow Processing for Deep Parsing – p.3
Preprocessing and Chunk Parsing

Steps consists of:

- POS tagging – TnT (Thorsten Brants)
Preprocessing and Chunk Parsing

Steps consists of:

- POS tagging – TnT (Thorsten Brants)
- tagfixing – making tags for specific words more informative
  e.g. abends (in the evening) ADV → TOD
  example: ...um sechs Uhr abends ...
  (at six o’clock in the evening) vs. ...um sechs Uhr jedenfalls ...
  (at six o’clock in any case)
Preprocessing and Chunk Parsing

Steps consists of:

- POS tagging – TnT (Thorsten Brants)
- tagfixing – making tags for specific words more informative
  e.g. abends (in the evening) ADV → TOD
  example: ...um sechs Uhr abends ...
  (at six o’clock in the evening) vs. ...um sechs Uhr jedenfalls ...
  (at six o’clock in any case)
- chunk parsing – CASS (Steve Abney)
**Chunk Parsing – Example**

- **sentence:** da muß ich leider zu einem Treffen nach Köln *(unfortunately I have to go to Cologne for a meeting)*
**Chunk Parsing – Example**

- **sentence:** da muß ich leider zu einem Treffen nach Köln *(unfortunately I have to go to Cologne for a meeting)*

```plaintext
[simpx
 [da da]
 [vmfi nmuß]
 [nx4
  [pper ich]]
 [advx
  [adv leider]]
 [px
  [zu zu]
  [nx1
   [art einem]
   [nn Treffen]]]
 [px
  [appr nach]
  [nx1
   [ne Köln]]]
```
The Treebank

- Tübingen Treebank of Spoken German: ca. 38,000 sentences, i.e. 66,000 trees
The Treebank

- Tübingen Treebank of Spoken German: ca. 38,000 sentences, i.e. 66,000 trees
- text domain: arrangement of business appointments, travel scheduling, and hotel reservations
The Treebank

- Tübingen Treebank of Spoken German: ca. 38,000 sentences, i.e. 66,000 trees
- text domain: arrangement of business appointments, travel scheduling, and hotel reservations
- transliterations of spontaneous speech
The Treebank

- Tübingen Treebank of Spoken German: ca. 38,000 sentences, i.e. 66,000 trees
- text domain: arrangement of business appointments, travel scheduling, and hotel reservations
- transliterations of spontaneous speech
- natural segmentation of spontaneous speech = *dialogue turn*, i.e. a single, generally uninterrupted contribution of one participant to the dialogue ↔ treebank: sentences
The Treebank

- Tübingen Treebank of Spoken German: ca. 38,000 sentences, i.e. 66,000 trees
- text domain: arrangement of business appointments, travel scheduling, and hotel reservations
- transliterations of spontaneous speech
- natural segmentation of spontaneous speech = *dialogue turn*, i.e. a single, generally uninterrupted contribution of one participant to the dialogue ↔ treebank: sentences
- hesitation noises, false starts, interruptions, repetitions, fragmentary or ungrammatical utterances
levels of annotation: morpho-syntax (POS tags), syntactic phrase structure, topological fields, function-argument structure
The Treebank – Annotation

- levels of annotation: morpho-syntax (POS tags), syntactic phrase structure, topological fields, function-argument structure
- as theory-neutral as possible and surface-oriented
The Treebank – Annotation

- levels of annotation: morpho-syntax (POS tags), syntactic phrase structure, topological fields, function-argument structure
- as theory-neutral as possible and surface-oriented
- no traces, no crossing branches: using function labels instead
(1) ich habe hier übrigens auch schon mir Unterlagen zuschicken lassen von verschiedenen Hotels

different hotels

'by the way here I have also had brochures sent to me about different hotels already'
(1) ich habe hier übrigens auch schon mir Unterlagen zuschicken lassen von verschiedenen Hotels

've by the way here I have also had brochures sent to me about different hotels already'
combining shallow and deep approaches in parsing very natural: shallow parsers and deep parsers
combining shallow and deep approaches in parsing very natural: shallow parsers and deep parsers

general approach when applying ML techniques: learning boundaries of chunks then dependencies between chunks
Standard ML Approaches to Parsing

- combining shallow and deep approaches in parsing very natural: shallow parsers and deep parsers
- general approach when applying ML techniques: learning boundaries of chunks then dependencies between chunks
- standard architecture: cascaded classifiers
Typical ML Parsing

cascaded classifiers: NP level, PP level, VP level, clause level, function argument structure
Typical ML Parsing

- cascaded classifiers: NP level, PP level, VP level, clause level, function argument structure

- example:

  I saw the man with the white hat
Typical ML Parsing

- cascaded classifiers: NP level, PP level, VP level, clause level, function argument structure

- example:

```
NP: [NP] [NP NP] [NP NP] [NP NP]
```

I saw the man with the white hat
Typical ML Parsing

- cascaded classifiers: NP level, PP level, VP level, clause level, function argument structure

- example:

PP: [PP PP]
NP: [NP] [NP NP] [NP NP]

I saw the man with the white hat
Typical ML Parsing

- cascaded classifiers: NP level, PP level, VP level, clause level, function argument structure

- example:

  VP: [VP VP]
  PP: [PP PP]
  NP: [NP NP NP]

  I saw the man with the white hat
Typical ML Parsing

- cascaded classifiers: NP level, PP level, VP level, clause level, function argument structure

- example:

CL.: [S  
VP:  [VP  
PP: [PP  
NP: [NP   [NP  NP]  [NP   NP]

I saw the man with the white hat
Typical ML Parsing

- cascaded classifiers: NP level, PP level, VP level, clause level, function argument structure

- example:

  func:  SB  DO  –
  CL.:  [S]
  VP:  [VP]
  PP:  [PP]
  NP:  [NP] [NP] [NP] [NP]

  I saw the man with the white hat
Problems with Cascades

- recursive structures such as complex clauses:

\[
S_1: \quad [S \quad S]
\]

\[
S_2: \quad [S \quad S]
\]

the man who bought everything made a fortune
Problems with Cascades

- recursive structures such as complex clauses:

  \[ S_1: \quad [S \quad S] \]

  \[ S_2: \quad [S \quad S] \]

  the man who bought everything made a fortune

- independence assumption:

  func: SB \quad SB

  I saw the man with the white hat
Problems with Cascades

- recursive structures such as complex clauses:

\[ S_1: [S \quad S] \]
\[ S_2: [S \quad S] \]
the man who bought everything made a fortune

- independence assumption:

func: SB \quad SB -
I saw the man with the white hat

- in German: long-distance relations:

ON \quad OD \quad OA \quad OA-MOD
ich habe mir Unterlagen zuschicken lassen von Hotels
I have to me brochures sent let of hotels
Memory-Based Learning

- *machine learning* (ML) algorithm based on classification: for each instance, selecting the most likely class from a fixed set of classes
Memory-Based Learning

- *machine learning* (ML) algorithm based on classification: for each instance, selecting the most likely class from a fixed set of classes

- *supervised ML* algorithm: needs training data
Memory-Based Learning

- *machine learning* (ML) algorithm based on classification: for each instance, selecting the most likely class from a fixed set of classes

- *supervised ML* algorithm: needs training data

- selection of the most likely class: finding the most similar instance in the instance base (collection of training instances)
Memory-Based Learning

- *machine learning* (ML) algorithm based on classification: for each instance, selecting the most likely class from a fixed set of classes

- *supervised ML* algorithm: needs training data

- selection of the most likely class: finding the *most similar* instance in the instance base (collection of training instances)

- no abstraction from data: original data is always accessible
Memory-Based Learning

- *machine learning* (ML) algorithm based on classification: for each instance, selecting the most likely class from a fixed set of classes
- *supervised ML* algorithm: needs training data
- selection of the most likely class: finding the most similar instance in the instance base (collection of training instances)
- no abstraction from data: original data is always accessible
- very appropriate for language learning: can deal with irregularities, subregularities, etc.
Memory-Based Learning

- *machine learning* (ML) algorithm based on classification: for each instance, selecting the most likely class from a fixed set of classes
- *supervised ML* algorithm: needs training data
- selection of the most likely class: finding the *most similar* instance in the instance base (collection of training instances)
- no abstraction from data: original data is always accessible
- very appropriate for language learning: can deal with irregularities, subregularities, etc.
- intelligence = good similarity metric
standard memory-based algorithms are highly sensitive to the selection of features and to the definition of their distance function

Using Shallow Processing for Deep Parsing – p.12
Feature Weighting

- standard memory-based algorithms are highly sensitive to the selection of features and to the definition of their distance function.

- solution putting *higher* weight on more *important* features and *less* weight on *unimportant* features.
standard memory-based algorithms are highly sensitive to the selection of features and to the definition of their distance function

solution putting *higher* weight on more *important* features and *less* weight on *unimportant* features

MBL approaches have a fixed number of features for each instance $\Rightarrow$ the weight represents the importance of one type of information
Feature Weighting

- standard memory-based algorithms are highly sensitive to the selection of features and to the definition of their distance function
- solution putting *higher* weight on more *important* features and *less* weight on *unimportant* features
- MBL approaches have a fixed number of features for each instance $\Rightarrow$ the weight represents the importance of one type of information
- many different weighting schemes, e.g. information gain, Chi-Square, etc.
New Approach

new idea: find most similar tree in instance base
in one step
New Approach

- new idea: find most similar tree in instance base in one step

- for new sentence: grundsätzlich habe ich Zeit (basically I have time)

find training sentence: da habe ich Zeit (I have time then)
New Approach

- new idea: find most similar tree in instance base in one step

- for new sentence: grundsätzlich habe ich Zeit (basically I have time)
find training sentence: da habe ich Zeit (I have time then)

- problem: how define similarity?
New Approach

- new idea: find most similar tree in instance base in one step

- for new sentence: grundsätzlich habe ich Zeit (basically I have time)
  find training sentence: da habe ich Zeit (I have time then)

- problem: how define similarity?

- problem: what if structure of most similar tree is not identical?
very conservative approach: only delete parts from retrieved tree, never add!
very conservative approach: only delete parts from retrieved tree, never add!

example: new sentence
am Mittwoch habe ich Zeit
(on Wednesday I have time)

training sentence:
am Dienstag den dreizehnten von zehn bis zwölf habe ich Zeit
(on Tuesday the thirteenth from ten to twelve I have time)
Adapting the Most Similar Tree

tree:

```
  am
 APPRART

  Dienstag
 NN

  den
 ART

  dreizehnten
 NN

  von
 APPR

  zehn
 CARD

  bis
 APPR

  zw"olf
 CARD

  habe
 VAFIN

  ich
 PPER

  Zeit
 NN
```
Adapting the Most Similar Tree

tree:

Using Shallow Processing for Deep Parsing – p.15
What types of information are helpful?
Finding the Most Similar Tree

What types of information are helpful?

- the sequence of words
Finding the Most Similar Tree

What types of information are helpful?

- the *sequence* of words
- the *sequence* of POS tags
Finding the Most Similar Tree

What types of information are helpful?

- the sequence of words
- the sequence of POS tags
- the sequence of chunks
Finding the Most Similar Tree

What types of information are helpful?

- the sequence of words
- the sequence of POS tags
- the sequence of chunks

- all of these types of information are readily available
Finding the Most Similar Tree

What types of information are helpful?

- the sequence of words
- the sequence of POS tags
- the sequence of chunks

- all of these types of information are readily available
- they do not serve as first steps of analysis but as features for finding a tree
Weighting Features?

Standard weighting techniques are impossible:

- sequential information more important: DET N V ADJ vs. ADJ, DET, N, V
- no windowing approach: find tree in one step
- use all features
- different number of features
- selecting a complete tree: very difficult task
- need all words and all other types of information as features

suggested solution: backoff strategy instead of weighting
Weighting Features?

Standard weighting techniques are impossible:

- sequential information more important:

  \[ \text{DET N V ADJ vs. ADJ, DET, N, V} \]
Weighting Features?

Standard weighting techniques are impossible:

- sequential information more important: DET N V ADJ vs. ADJ, DET, N, V
- no windowing approach: find tree in one step ⇒ use all features ⇒ different number of features
Weighting Features?

Standard weighting techniques are impossible:

- sequential information more important: \( \text{DET N V ADJ vs. ADJ, DET, N, V} \)
- no windowing approach: find tree in one step ⇒ use all features ⇒ different number of features
- selecting a complete tree: very difficult task ⇒ need all words and all other types of information as features
Weighting Features?

Standard weighting techniques are impossible:

- sequential information more important:
  DET N V ADJ vs. ADJ, DET, N, V

- no windowing approach: find tree in one step ⇒
  use all features ⇒ different number of features

- selecting a complete tree: very difficult task ⇒
  need all words and all other types of information as features

- suggested solution: backing off strategy instead of weighting
backing off: 2 main modules:
backing off: 2 main modules:

- search in a word prefix trie allowing the omission of words or phrases / chunks in input sentence as well as training sentences

backing off to less reliable information:
1. search for POS sequence
2. search for longer trees and shorten them
3. search for chunk sequences with matching heads
4. search for chunk sequences (without matching heads)
backing off: 2 main modules:

- search in a word prefix trie allowing the omission of words or phrases / chunks in input sentence as well as training sentences

- backing off to less reliable information:
  1. search for POS sequence
The Deep Parsing System

backing off: 2 main modules:

- search in a word prefix trie allowing the omission of words or phrases / chunks in input sentence as well as training sentences
- backing off to less reliable information:
  1. search for POS sequence
  2. search for longer trees and shorten them
backing off: 2 main modules:

- search in a word prefix trie allowing the omission of words or phrases / chunks in input sentence as well as training sentences

backing off to less reliable information:

1. search for POS sequence
2. search for longer trees and shorten them
3. search for chunk sequences with matching heads
backing off: 2 main modules:

- search in a word prefix trie
  allowing the omission of words or phrases / chunks in input sentence as well as training sentences

- backing off to less reliable information:
  1. search for POS sequence
  2. search for longer trees and shorten them
  3. search for chunk sequences with matching heads
  4. search for chunk sequences (without matching heads)
mache
ich
Ihnen

denn
das
mit
der
den
Flug

das
am
doch
elften

Ihren
einfach

Vorschlag

machen
ich
Sie
darf
mal

einen

flug

fertig

 Tickets

Flugverbindung

Konferenz

machen

sprechen
sentence: wie sieht das ab dem fünfundzwanzigsten aus (how does that look from the twenty fifth on)
The Omission of Words in the Trie

sentence: wie sieht das ab dem fünfundzwanzigsten aus (how does that look from the twenty fifth on)

Using Shallow Processing for Deep Parsing – p.20
The Omission of Words in the Trie

sentence: wie sieht das ab dem fünfundzwanzigsten aus (how does that look from the twenty fifth on)

Using Shallow Processing for Deep Parsing – p.20
The Omission of Words in the Trie

Sentence: wie sieht das ab dem fünfundzwanzigsten aus (how does that look from the twenty fifth on)
sentence: wie sieht das ab dem fünfundzwanzigsten aus (how does that look from the twenty fifth on)
input sentence: ab Donnerstag bin ich wieder hier (from Thursday on I will be here again)
input sentence: ab Donnerstag bin ich wieder hier (from Thursday on I will be here again)

chunk structure:
[simp x [px ab Donnerstag] [fcop bin] [nx4 ich] [adv x wieder] [adv x hier]]
input sentence: ab Donnerstag bin ich wieder hier (from Thursday on I will be here again)

chunk structure:

\[[\text{simp} \ [\text{px} \ ab \ Donnerstag] \ [\text{fcop} \ bin] \ [\text{nx}4 \ ich] \ [\text{av}x \ wieder] \ [\text{av}x \ hier]]\]

identical chunk structure from training data:

\[[\text{simp} \ [\text{px} \ ab \ Donnerstag \ dem \ dritten] \ [\text{fcop} \ bin] \ [\text{nx}4 \ ich] \ [\text{av}x \ wieder] \ [\text{av}x \ hier]]\]

\[(after \ a \ long \ week \ you \ will \ be \ back \ again)\]
input sentence: ab Donnerstag bin ich wieder hier (from Thursday on I will be here again)

chunk structure:
[simpx [px ab Donnerstag] [fcop bin] [nx4 ich] [advx wieder] [advx hier]]

identical chunk structure from training data:
[simpx [px ab Donnerstag dem dritten] [fcop bin] [nx4 ich] [advx wieder] [advx hier]]

identical chunk structure from training data:
[simpx [px nach einer langen Woche] [fcop sind] [nx4 Sie] [advx wieder] [advx zurück]]
(after a long week you will be back again)
Tree Modification

Using Shallow Processing for Deep Parsing – p.23
Using Shallow Processing for Deep Parsing – p.23
Tree Modification

Using Shallow Processing for Deep Parsing – p.23
<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>recall (syntactic)</td>
<td>82.45%</td>
</tr>
<tr>
<td>precision (syntactic)</td>
<td>87.25%</td>
</tr>
<tr>
<td>$F_1$</td>
<td>84.78</td>
</tr>
<tr>
<td>recall (+ func. cat.)</td>
<td>71.72%</td>
</tr>
<tr>
<td>precision (+ func. cat.)</td>
<td>75.79%</td>
</tr>
<tr>
<td>$F_1$</td>
<td>73.70</td>
</tr>
<tr>
<td>unattached const. in recall</td>
<td>7.14%</td>
</tr>
<tr>
<td>unattached const. in precision</td>
<td>7.60%</td>
</tr>
<tr>
<td>func. recall (att. const.)</td>
<td>95.31%</td>
</tr>
<tr>
<td>func. precision (att. const.)</td>
<td>95.21%</td>
</tr>
</tbody>
</table>
Leave-One-Out Evaluation

using **5000 test sentences**:

<table>
<thead>
<tr>
<th></th>
<th>leave-one-out:</th>
<th>previous:</th>
</tr>
</thead>
<tbody>
<tr>
<td>recall (syntactic)</td>
<td>85.15%</td>
<td>82.45%</td>
</tr>
<tr>
<td>precision (syntactic)</td>
<td>89.34%</td>
<td>87.25%</td>
</tr>
<tr>
<td>$F_1$</td>
<td>87.19</td>
<td>84.78</td>
</tr>
<tr>
<td>recall (+ func. cat.)</td>
<td>76.00%</td>
<td>71.72%</td>
</tr>
<tr>
<td>precision (+ func. cat.)</td>
<td>79.65%</td>
<td>75.79%</td>
</tr>
<tr>
<td>$F_1$</td>
<td>77.78</td>
<td>73.70</td>
</tr>
<tr>
<td>func. recall (att. const.)</td>
<td>96.56%</td>
<td>95.31%</td>
</tr>
<tr>
<td>func. precision (att. const.)</td>
<td>96.48%</td>
<td>95.21%</td>
</tr>
</tbody>
</table>
Conclusion

new definition of parsing task: find complete most similar tree; adapt this tree to input
Conclusion

- new definition of parsing task: find complete most similar tree; adapt this tree to input
- needs: POS tagger, chunk parser, treebank
Conclusion

- new definition of parsing task: find complete most similar tree; adapt this tree to input
- needs: POS tagger, chunk parser, treebank
- extremely fast: almost deterministic
Conclusion

- new definition of parsing task: find complete most similar tree; adapt this tree to input
- needs: POS tagger, chunk parser, treebank
- extremely fast: almost deterministic
- uses a backing off strategy instead of (standard) feature weighting
Conclusion

- new definition of parsing task: find complete most similar tree; adapt this tree to input
- needs: POS tagger, chunk parser, treebank
- extremely fast: almost deterministic
- uses a backing off strategy instead of (standard) feature weighting
- results still worse results than state of the art statistical parsers, but: different language, different data