**What is chunking?**

Long sentences pose a computational challenge in many NLP tasks, such as parsing or translation. Chunking makes long sentences more tractable for further processing. Chunking a sentence means cutting a complex sentence into grammatical constituents that can be processed independently and then recombined without loss of information. It can be defined both on the surface string of a sentence and on its semantic representation, and is applicable to a wide range of tasks.

Since chunking is one of the steps in a processing pipeline, we prioritize precision over coverage to minimize error propagation. The goal is to chunk fewer sentences but correctly rather than more but with low precision.

**Chunking semantic representations** We approach chunking using rules defined on Dependency Minimal Recursion Semantics (DMRS) (Copestake, 2009) graphs – the link structure of the graph reveals appropriate chunk boundaries. Chunking should be adaptable for other semantic representations.

**Chunking surface strings** The alignment between the semantic and surface representations of a sentence allows us to cut the sentence string into surface chunks. We intend to use the rule-based approach to chunking DMRSs to create training data for a minimally supervised machine learning algorithm that performs string chunking.

**Rule-based DMRS chunking**

1. Discover a potential chunking point by spotting a trigger node (marked bold in the figures).
2. Check whether the clauses associated with the trigger node are finite—follow links outgoing from the trigger node which lead to heads of the clauses (cut by red lines in the example figures).

Rules determining triggers and finite clauses were developed through examination of data and discovering structural patterns in DMRS graphs.

So far we perform chunking based on three grammatical structures:

- **subordinate conjunctions**, e.g.

  ```
  Since I bought a cat, we have had no problems with mice.
  ```

- **clausal coordinations**, e.g.

  ```
  Kim is afraid of mice and she does not like cats.
  ```

- **clausal complements**, e.g.

  ```
  John didn’t know (that) he won the lottery.
  ```

**Evaluation**

**Procedure:**

1. Chunk the surface of each sentence into substrings using its semantic representation.
2. Parse each of the resulting surface chunks with the ACE.
3. Produce a realization for the top parse for each chunk using ACE.
4. Recombine the realizations and compare the results with the original sentence.

**Baseline:** Simple string-based heuristics, similar to one of the techniques used currently in statistical machine translation community.

**Datasets:**

- 1212 WikiWoods release (Flickinger et al., 2010), a snapshot of Wikipedia from July 2008, only graphs with 40+ nodes; automatically generated parses, not verified;
- WeScience (Ytrestøl, 2009), a human-verified subset of parses for the same Wikipedia snapshot.

<table>
<thead>
<tr>
<th>Algorithm (Dataset)</th>
<th>Precision</th>
<th>Correct</th>
<th>Incorrect</th>
<th>Attempted</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMRS-based (WikiWoods)</td>
<td>42.0%</td>
<td>3066</td>
<td>4195</td>
<td>24.9%</td>
</tr>
<tr>
<td>Baseline (WikiWoods)</td>
<td>19.6%</td>
<td>3783</td>
<td>15526</td>
<td>66.6%</td>
</tr>
<tr>
<td>DMRS-based (WeScience)</td>
<td>62.7%</td>
<td>106</td>
<td>63</td>
<td>22.7%</td>
</tr>
<tr>
<td>Baseline (WeScience)</td>
<td>42.0%</td>
<td>3036</td>
<td>4195</td>
<td>24.9%</td>
</tr>
</tbody>
</table>

Fig. 4 Performance of the DMRS-based chunking algorithm and the baseline on the WikiWoods and WeScience datasets.

Prioritizing precision over coverage means that high Precision is more important than high Attempted value.

**Current and future research**

Major objective: Development of the minimally supervised surface chunking machine learning algorithm that would make chunking applicable to parsing and other tasks for which a deep parse is unavailable.

- New intrinsic evaluation: directly compare chunk and original sentence DMRSs, eliminating the realization step, score the match on a continuous scale.
- Training dataset for machine learning: high-quality dataset of sentence surfaces, created with the rule-based DMRSs system.
- Extrinsic evaluation: e.g. statistical realization (Horvat et al., 2015), ACE realization.
- Combining multiple parses: chunking can fail because the parse on which we base it is incorrect. Consider more than just the top parse.
- Extending the coverage: relative clauses, verb phrase coordinations, gerund-based adjuncts, parentheticals, appositions.
- Improved treatment of clausal complements: use the ERG lexicon to avoid chunking with obligatorily syntactic elements.

**Related tasks**

**Clause splitting** Sentence chunking resembles clause splitting as defined by the CoNLL-2001 shared task (Tjong et al., 2001). Clauses are finite clauses but each can consist of multiple smaller clauses and not every clause boundary is a chunk boundary. A key aspect of sentence chunking is deciding where to place a chunk border.

**Simplification** Sentence chunking is a natural step in a simplification process, among other rewrite operations such as paraphrase extraction, but while sentence simplification modifies sentences, replacing lexical items and rearranging order of information, sentence chunking aims to preserve as much of the original sentence as possible.

**Framework**

Our rule-based chunking system relies on the English Resource Grammar (ERG; Flickinger, 2000), a broad-coverage, symbolic grammar of English, developed as part of DELPH-IN initiative and LinGO project. The ERG uses Minimal Recursion Semantics (MRS) (Copestake et al., 2005) as its semantic representation. The MRS format can be transformed into a more readable DMRS (Copestake, 2009) which represents its dependency structure. DMRS graphs can be manipulated using two existing Python libraries:

- pydmrs (Copestake et al., 2016; https://github.com/delph-in/pydmrs)

The ERG is a bidirectional grammar which supports both parsing and generation. There exist several processors, which parse sentences into MRSs and generate surface forms from MRS. In our experiments we use ACE (Woodley Packard’s Answer Constraint Engine, http://sweaglesw.org/linguistics/ace/).