Graph- and surface-level sentence chunking

Ewa Muszyńska
Computer Laboratory
University of Cambridge
emm68@cam.ac.uk

Abstract

The computing cost of many NLP tasks increases faster than linearly with the length of the representation of a sentence. For parsing the representation is tokens, while for operations on syntax and semantics it will be more complex. In this paper we propose a new task of *sentence chunking*: splitting sentence representations into coherent substructures. Its aim is to make further processing of long sentences more tractable. We investigate this idea experimentally using the Dependency Minimal Recursion Semantics (DMRS) representation.

1 Introduction

Long sentences pose a challenge in many Natural Language Processing (NLP) tasks, such as parsing or translation. We propose chunking as a way of making such sentences more tractable before 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. Such an operation 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.

Some approaches to parsing have space and time requirements which are much worse than linear in sentence length. This can lead to practical difficulties in processing. For example, the ACE processor[1] running the English Resource Grammar (ERG) (Copestake and Flickinger, 2000) requires roughly 530 MB of RAM to parse Sentence 1. In fact, longer and more complicated sentences can cause the parser to time out or run out of memory before a solution is found.

(1) Marcellina has hired Bartolo as her counsel, since Figaro had once promised to marry her if he should default on a loan she had made to him, and she intends to enforce that promise.

Chunking would make processing of long sentences more tractable. For example, we aim to split sentences like Sentence 1 into chunks 2a–d.

(2) a. Marcellina has hired Bartolo as her counsel.
b. Figaro had once promised to marry her.
c. He should default on a loan she made to him.
d. She intends to enforce that promise.

Each of these shorter sentences can be parsed with less than 20 MB, requiring in total less than a fifth of RAM needed to parse the full sentence.

What exactly constitutes a valid chunk has to be considered in the context of the task which we want to simplify by chunking. In this sense a potentially useful analogy could be made to the use of factoids in summarisation (Teufel and Van Halteren, 2004; Nenkova et al., 2007). However, we can make some general assumptions about the nature of 'good' chunks. They have to be semantically and grammatically self-contained parts of the larger sentence.

Sentence chunking resembles clause splitting as defined by the CoNLL-2001 shared task (Tjong et al., 2001). Each of the chunks a–d is a finite clause, although each consists of multiple smaller clauses. This points to a crucial difference between sentence chunking and clause splitting which justifies treating them as separate tasks.

[1] Woodley Packard's Answer Constraint Engine, http://sweaglesw.org/linguistics/ace/

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics – Student Research Workshop, pages 93–99,
Berlin, Germany, August 7-12, 2016. ©2016 Association for Computational Linguistics

We define chunking in terms of its purpose as a pre-processing step and because of that it is more restrictive. Not every clause boundary is a chunk boundary. A key aspect of sentence chunking is deciding where to place a chunk border so that the resulting chunks can be processed and recombined without loss of information.

Another difference between sentence chunking and clause splitting is the domain of the task. Clause splitting is performed on the surface string of a sentence, while we can define chunking not only on the surface representation but also on more complex ones, such a graph-based semantic representation.

There are two reasons why chunking a semantic representation is a good idea:

1. Many operations on graphs have worse than linear complexity, some types of graph matching are NP-complete. Chunking semantic representations can make their manipulation more tractable (Section 1.1).

2. Such a form of chunking, apart from being useful in its own right, can also help chunking surface sentences (Section 1.2).

1.1 Chunking semantic representations

In this paper we describe an approach to sentence chunking based on Dependency Minimal Recursion Semantics (DMRS) graphs (Copestake, 2009). We chunk a sentence by dividing its semantic representation into subgraphs corresponding to logical chunks. The link structure of a DMRS graph reveals appropriate chunk boundaries. Since we envision chunking to be 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.

Sentence chunking understood as graph chunking of a semantic representation can be directly useful for applications that already use the representation. Although we use the DMRS, chunking could be just as well adapted for other semantic representations, for example AMR (Abstract Meaning Representation) (Banarescu et al., 2013). Part of our reason to choose the DMRS framework was the fact that the DMRS format is readily interchangeable with Minimal Recursion Semantics (MRS). Thanks to this relationship our system is compatible with any applications stemming from the DELPH-IN initiative[2].

Horvat et al. (2015) introduce a statistical approach to realization, in which they treat realization like a translation problem. As part of their approach, they extract grammatical rules based on DMRS subgraphs. Since operations on subgraphs are computationally expensive, chunking the sentence before the algorithm is applied could reduce the complexity of the task.

Another task which could benefit from chunking is treebanking. LinGO Redwoods 2 (Oepen et al., 2004) is an initiative aimed at designing and developing a treebank which supports the HPSG grammar. The treebank relies on discriminants to differentiate and choose between possible parses. Chunking could be used to preferentially select parses which contain subtrees corresponding to well-formed chunk subgraphs.

1.2 Towards string chunking

The DMRS-based rule approach cannot be itself used to improve parsing because it requires a full parse to find the chunks in the first place. However, development of the surface chunking machine learning algorithm can extend applicability of chunking to parsing and other tasks for which a deep parse is unavailable.

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 create training data for a minimally supervised machine learning algorithm.

Following Rei (2013, pp. 11-12) we use the term 'minimally supervised' to mean a system trained using "domain-specific resources, other than annotated training data, which could be produced by a domain-expert in a relatively short time". In our case the resource is a small set of manually coded rules developed through examination of data.

The ultimate goal of our work is the creation of a reliable tool which performs chunking of sentence strings without relying on semantic representation and deep parsing. The applicability of chunking would then extend to tasks which cannot rely on deep parsing, such as statistical machine translation or parsing itself.

The next sections give more details on the

[2]Deep Linguistic Processing with HPSG, `www.delph-in.net`

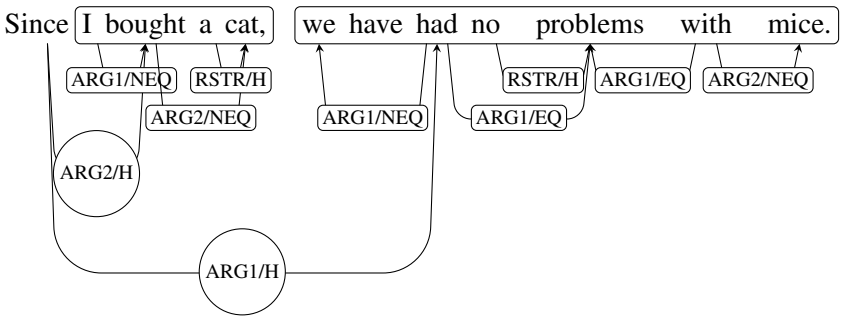

Figure 1: A DMRS graph of a sentence *Since I bought a cat, we have had no problems with mice*. The two chunks are marked, while *since* is separated as a functional chunk and chunking trigger. The links with circular labels are crucial for chunking.

DELPH-IN framework, DMRS and our approach to rule-based chunking. We present our preliminary results in Section 4 and outline our current investigation focus and future research directions in Sections 5. Chunking is a new task, however it is related to several existing ones as discussed in Section 6.

2 DELPH-IN framework and DMRS

The rule-based chunking system we developed is based on the English Resource Grammar (ERG) (Flickinger, 2000), a broad-coverage, symbolic grammar of English. It was developed as part of DELPH-IN initiative and LinGO[3] 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 Dependency Minimal Recursion Semantics (DMRS) graph (Copestake, 2009), which represents its dependency structure. The nodes correspond to predicates; edges, referred to as links, represent relations between them. An example of a DMRS graph is shown in Figure 1.

DMRS graphs can be manipulated using two existing Python libraries. The `pyDelphin`[4] is a more general MRS-dedicated library. It allows conversions between MRS and DMRS representations but internally performs operations on MRS objects. The `pydmrs` library[5] (Copestake et al., 2016) is dedicated solely to DMRS manipulations. The work described in Section 4 used `pyDelphin`.

[3]Linguistic Grammars Online, `lingo.stanford.edu`

[4]`https://github.com/delph-in/pydelphin`

[5]`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 representations using chart generation. In our experiments we use ACE[6] to obtain MRSs and to generate from them, so that parsing and generation themselves are performed using already existing DELPH-IN tools. The chunking algorithm operates on graphs – we use the `pyDelphin` and `pydmrs` libraries for MRS-DMRS conversion and for manipulating DMRS objects.

3 DMRS-based chunking

In our research so far we have restricted valid chunks to finite clauses. A sentence is chunked correctly if all the chunks are either full finite clauses with a subject-verb structure or functional trigger chunks, such as *since* or *and*. A chunk can consist of multiple clauses if it is needed to ensure that all chunks are satisfactory.

The finite clause restriction was introduced because well-formedness of finite clauses can be easily checked and they can be more readily processed independently and recombined than other types of clauses.

We developed the chunking rules through examination of data and finding structural patterns in DMRS graphs. Currently chunking is based on three grammatical constructions: clausal coordination (3), suboordinating conjunctions (4ab) and clausal complements (5).

(3) The cat chased a toy and the dog slept under the table.

[6]Woodley Packard's Answer Constraint Engine, `http://sweaglesw.org/linguistics/ace/`

(4) a. The cat chased a toy because it was bored.

b. Since the dog slept, Kim didn't offer it a snack.

(5) Kim thought that they should talk.

Extending the coverage of the technique to other structures is one of future directions of investigation.

We discover potential chunking points by spotting trigger nodes. Those are the nodes which correspond to coordinating and subordinating conjunctions, and to verbs with clausal complements. In the example from Figure 1 *since* is a trigger.

After a trigger is found, we check whether the clauses associated with it are finite. We can do that by following links outgoing from the trigger node which lead to heads of the clauses. We marked these links in the figure with circular labels. In symmetric constructions, such as coordination, chunks are separated unambiguously by a conjunction. In other cases, such as the one in the example, we can find the chunk border by detecting a gap in the graph's link structure. No links outgoing from either of the main chunks span the gap between *cat* and *we* in Figure 1.

4 Preliminary results

So far we evaluated the system using a parsing and regeneration procedure, leveraging bidirectionality of the ERG. The surface of each sentence was chunked into substrings based on its semantic representation. Each of the resulting surface chunks was then parsed using the ACE. Next we fed the top parse for each chunk as input to the ACE generator, which produced the surface matching the semantic representation of the chunk. Finally, we recombined the surfaces generated in this fashion and compared the results with the original sentence.

The parsing and regeneration is a way of checking whether any information loss was caused by chunking. We do not attempt to improve parsing, only to evaluate how well the chunks meet the criteria of well-formedness and applicability we posit. At the same time this form of evaluation assesses the semantic representation chunking only indirectly, focusing on the quality of produced surface chunks. This is desirable in for creating a good quality dataset for the minimally supervised machine learning algorithm discussed in Section 1.2.

As our dataset, we used the 1212 release of the WikiWoods corpus (Flickinger et al., 2010) which is a snapshot of Wikipedia from July 2008. The entire corpus contains 44,031,336 entries, from which we selected only long sentences, viz. sentences with more than 40 nodes in their DMRS graph. Additionally we filtered out some non-sentential entries.

We compared the results obtained using the DMRS-based system with a simple string-based heuristic baseline, similar to one of the techniques used currently in statistical machine translation community[7]. The baseline attempts to chunk 67% of long sentences it encounters, compared with 25% attempted by the DMRS-based approach. As a result, the absolute number of sentences the baseline chunks correctly is greater but low precision makes the heuristic approach highly unreliable. Any application which used it would require a lot of human supervision. The DMRS-based procedure correctly chunks 42.0% of sentences in which it finds chunking opportunities, while baseline correctly chunks only 19.6% of sentences.

The evaluation method with which we obtained these results was harsh. It required all non-functional chunks to be finite clauses. If even one of many chunks was not a finite clause, we counted the entire sentence as chunked incorrectly. Some errors occurred in the final step of the evaluation: generation from chunk's surface string. We required a high similarity between the reconstructed sentence and the original. For example, according to the ERG lexicon, *St* and *Street* have the same semantic representation and the generator can't choose between them. If a generated string contained *Baker Street* when the original used *Baker St*, the difference would be penalised even though the two are equivalent. More than one mistake of this kind in a sentence would be enough to reject the result as incorrect.

A significant percentage of errors stems from the dataset itself. Sentences and parses in the WikiWoods dataset were not checked by humans. In fact, not all Wikiwoods entries are grammatical sentences and many of them could not be easily filtered out. Bearing that in mind we briefly repeated the experiment with a smaller WeScience corpus[8] (Ytrestøl et al., 2009). Like WikiWoods,

[7] Cambridge SMT system: Source sentence chopping, `http://ucam-smt.github.io/tutorial/basictrans.html#chopping`

[8] `http://moin.delph-in.net/WeScience`

Algorithm (Dataset)	Precision	Correct	Incorrect	Attempted
DMRS-based (WikiWoods)	42.0%	3036	4195	24.9%
Baseline (WikiWoods)	19.6%	3783	15526	66.6%
DMRS-based (WeScience)	62.7%	106	63	22.7%
Baseline (WeScience)	14.2%	60	362	56.7%

Table 1: Performance of the DMRS-based chunking algorithm and the baseline on the WikiWoods and WeScience datasets. Precision is the percentage of attempted sentences which were chunked correctly, while Correct and Incorrect columns give absolute numbers of correctly and incorrectly chunked sentences. Attempted column is the percentage of sentences for which a chunking opportunity was found and attempted.

it originates from Wikipedia but has been checked by human annotators.

Indeed, the chunking procedure performs much better on the human-checked dataset: 62.7% correct chunkings as compared with 42% for WikiWoods (Table 1), indicating the algorithm's sensitivity to parsing errors.

The error analysis of the WeScience experiment reveals that over 25% of the errors made by the rules-based system can be explained by the presence of grammatical structures which the rules did not account for. Increasing the coverage of structures used for chunking should decrease the number of errors of this origin. Another common source of errors were adverbs and prepositional phrases left behind after chunking sentences beginning with *However, when...* or *For example, if...*. We address this issue in the newer version of the system.

For comparison, the string heuristics baseline makes chunking decisions based solely on the presence of trigger words, such as *and*, without the knowledge of what clauses are involved. The position of good chunking boundaries is often determined by dependencies between distant parts of the surface, which are difficult to capture with string-based rules, but are clearly reflected in the DMRS link structure. This results in the baseline yielding unsatisfactory chunks like those underlined in Sentence 6.

(6) The dog barked *and* chased the cat.

5 Current work and future research

Currently we are preparing a different evaluation technique which will directly compare DMRS representations of chunks and the original sentence, eliminating the generation step responsible for many errors. In the new evaluation chunk graphs are matched against the full graph using the `pyDmrs` matching module (Copestake et al., 2016) which scores the degree of the match on a continuous scale.

We are also cooperating with the authors of the statistical approach to realisation (Horvat et al., 2015) on incorporating chunking into their graph manipulations. We hope to use their system for extrinsic evaluation.

Sentences which would most benefit from chunking are also, not accidentally, sentences with which parsers struggle most. Chunking often fails because the parse on which we base it is incorrect. In the future we would like to experiment with considering a number of parses instead of just the top one. This would enable us to mix chunking into the correct parse selection procedure.

One of the investigation directions is extending the catalogue of grammatical structures on which we base the chunks. Some syntactical structures we consider as extensions are relative clauses, verb phrase coordinations, gerund-based adjuncts, parentheticals and appositions. Their inclusion would increase the coverage and quality of chunks, crucial for our purposes.

The treatment of clausal complements needs improvement as well. Some clauses are obligatory syntactic elements and their removal changes how the main clause is parsed. We do not address this issue in the current early version of the system but the lexicalist nature of the ERG offers a solution. The information about whether a clausal complement is obligatory for a given verb is contained in the grammar's lexicon and can be leveraged to improve chunking decisions. We aim to include this mechanism in a later version of the algorithm.

DMRS graphs store information about the alignment between nodes and surface fragments. This information allows us to chunk surfaces of sentences based on the results of graph chunking.

As discussed in Section 1.2, we intend to create a training dataset for a machine learning algorithm which would perform surface chunking. Since, as the WeScience experiment showed, our rule-based approach is sensitive to errors in original parses of full sentences, we might base our training corpus on the RedWoods treebank, which is larger than WeScience but still human-checked.

6 Related work

We define sentence chunking as a new task. As discussed in Introduction, it bears similarity to clause splitting but because of its definition in terms of functionality, it has to be considered separately.

The most important similarity between chunking and clause splitting is how the two problems can be defined for the purpose of machine learning. Clause splitting was the CoNLL-2001 shared task (Tjong et al., 2001) and the results of that research can guide the development of a machine learning system for chunking. Another task which can provide insights into how to design a suitable machine learning system is Sentence Boundary Disambiguation (SBD) task (Walker et al., 2001).

Other research related to chunking was conducted in the context of text simplification. Sentence chunking is a natural step in a simplification process, among other rewrite operations such as paraphrase extraction, but the two tasks have different goals. 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.

Chandrasekar et al. (1996) suggested using dependency structures for simplifying sentences. The authors gave an example of simplifying relative clauses that is similar to chunking but outside of the current scope of our experiments. This research represented early work on automatic syntactic simplification and was succeeded by Siddharthan (2010) who performs simplification by defining transformation rules over type dependency structures. Siddharthan's approach mixes lexical and syntactical transformations and cannot be directly compared with chunking.

Another example of work on simplification is a paper by Woodsend and Lapata (2011). The authors call sentence chunking sentence splitting and approach it from the perspective of tree-based Quasi-synchronous Grammar (QG). Their algorithm learns possible chunking points by aligning the original sentence with two shorter target sentences. Unlike the method we propose, the QG approach requires a manually created dataset consisting of original and target sentences from which the rules can be inferred. Unfortunately, it is impossible to compare the performance of our sentence chunking and the authors' sentence splitting. The QG splitting algorithm is an integral part of the text simplification system and the paper describing it does not give any numbers regarding the performance of individual parts of the system.

7 Conclusions

We defined sentence chunking in terms of its usefulness for other tasks. Its aim is to produce chunks which can be processed and recombined without loss of information. The procedure can be defined for both the surface of a sentence and for its semantic representation.

In our experiments we perform chunking using rules based on the DMRS graphs of sentences. Our work is an early attempt at the task so we focus on easier cases, aiming to gradually increase coverage. Since chunking is intended as a preprocessing step for other tasks, the reliability and precision are more important than chunking as many sentences as possible. Bearing this in mind, we are satisfied to report that according to preliminary experiments, our chunking procedure attempted 25% of all sentences in the dataset and it chunked 42% of these correctly. For comparison, a baseline using heuristics attempted to chunk 67% of sentences, but only 19.6% of these sentences were chunked correctly.

The DMRS-based graph chunking can be used to improve existing systems such as the statistical realization algorithm (Horvat et al., 2015) or to guide the selection of parses for LinGO RedWoods 2 treebank (Oepen et al., 2004).

The surface chunking machine learning tool will extend the applicability of chunking even further. Eliminating the immediate reliance on the parse could allow chunking to replace the string heuristics for machine translation and to influence parsing itself, reducing the difficulty of the task.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

R. Chandrasekar, Christine Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING '96*, pages 1041–1044.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of LREC 2000*, pages 591–600.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2):281–332.

Ann Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszyńska. 2016. Resources for building applications with Dependency Minimal Recursion Semantics. In *Proceedings of the Tenth Language Resources and Evaluation Conference (LREC '16)*. In press.

Ann Copestake. 2009. Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 1–9, Athens, Greece, March. Association for Computational Linguistics.

Dan Flickinger, Stephan Oepen, and Gisle Ytrestøl. 2010. WikiWoods: syntacto-semantic annotation for English Wikipedia. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Matic Horvat, Ann Copestake, and William Byrne. 2015. Hierarchical Statistical Semantic Realization for Minimal Recursion Semantics. In *Proceedings of the International Conference on Computational Semantics (IWCS 2015)*.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, 4(2), May.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and ChristopherD. Manning. 2004. LinGO Redwoods. *Research on Language and Computation*, 2(4):575–596.

Marek Rei. 2013. Minimally supervised dependency-based methods for natural language processing. Technical Report 840, Computer Laboratory, University of Cambridge. PhD thesis.

Advaith Siddharthan. 2010. Complex lexico-syntactic reformulation of sentences using typed dependency representations. In *Proceedings of the 6th International Natural Language Generation Conference*, INLG '10, pages 125–133, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cambridge SMT system: source sentence chopping. `http://ucam-smt.github.io/tutorial/basictrans.html#chopping`. Accessed: 2016-04-26.

Simone Teufel and Hans Van Halteren. 2004. Evaluating information content by factoid analysis: Human annotation and stability. In *Proceedings of Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 419–426.

Erik F. Tjong, Kim Sang, and Hervé Déjean. 2001. Introduction to the CoNLL-2001 shared task: Clause identification. In *Proceedings of the 2001 Workshop on Computational Natural Language Learning - Volume 7*, ConLL '01, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel J. Walker, David E. Clements, Maki Darwin, and Jan W. Amtrup. 2001. Sentence boundary detection: A comparison of paradigms for improving MT quality. In *roceedings of MT Summit VIII: Santiago de Compostela*, pages 18–22.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 409–420, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gisle Ytrestøl, Dan Flickinger, and Stephan Oepen. 2009. Extracting and annotating Wikipedia subdomains - towards a new eScience community resource. In *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT 7)*, pages 185–197, Groningen, The Netherlands.