Large-scale CCG Induction from the Groningen Meaning Bank

Sebastian Beschke[*], **Yang Liu**[†] and **Wolfgang Menzel**[*]
[*]Department of Informatics, University of Hamburg, Germany
{beschke,menzel}@informatik.uni-hamburg.de
[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
liuyang2011@tsinghua.edu.cn

Abstract

In present CCG-based semantic parsing systems, the extraction of a semantic grammar from sentence-meaning examples poses a computational challenge. An important factor is the decomposition of the sentence meaning into smaller parts, each corresponding to the meaning of a word or phrase. This has so far limited supervised semantic parsing to small, specialised corpora. We propose a set of heuristics that render the splitting of meaning representations feasible on a large-scale corpus, and present a method for grammar induction capable of extracting a semantic CCG from the Groningen Meaning Bank.

1 Introduction

Combinatory Categorial Grammar (CCG) forms the basis of many current approaches to semantic parsing. It is attractive for semantic parsing due to its unified treatment of syntax and semantics, where the construction of the meaning representation directly follows the syntactic analysis (Steedman, 2001). However, the supervised induction of semantic CCGs—the inference of a CCG from a corpus of sentence-meaning pairs—has so far only been partially solved. While approaches are available that work on small corpora focused on specific domains (such as Geoquery and Freebase QA for question answering (Zelle and Mooney, 1996; Cai and Yates, 2013)), we are not aware of any approach that allows the extraction of a semantic CCG from a wide-coverage corpus such as the Groningen Meaning Bank (GMB) (Basile et al., 2012). This work attempts to fill this gap.

Analogous to the work of Kwiatkowski et al. (2010), we view grammar induction as a series of splitting steps, each of which essentially reverses a CCG derivation step. However, we diverge from their approach by applying novel heuristics for searching the space of possible splits. The combination of alignment consistency and single-branching recursion turns out to produce a manageable number of lexical items for most sentences in the GMB, while statistical measures and manual inspection suggest that many of these items are also plausible.

2 Searching the space of CCG derivations

Our search heuristics are embedded into a very general splitting algorithm, Algorithm 1. Given a sentence-meaning pair, it iterates over all possible sentence-meaning splits in two steps. First, a split index in the sentence is chosen along with a binary CCG-combinator to be reversed (the *syntactic split*). Then, the meaning representation is split accordingly to reverse the application of the selected combinator (the *semantic split*). E. g., for the forward application combinator, the meaning representation z is split into f, g so that $z = fg$ (modulo α, β, η conversions). By identifying f with the left half l of the sentence and g with the right half r, we obtain two new phrase-meaning pairs, which are then split recursively.

This algorithm combines two challenging search problems. Recursive syntactic splitting searches the space of syntactic CCG derivations that yield the sentence, which is exponential in the length of the sentence. Semantic splitting, given the flexibility of λ-calculus, has infinitely many solutions. The crucial question is how to prune the parts of the search space that are unlikely to lead to good results.

Our strategy to address this problem is to apply heuristics that constrain the results returned by semantic splitting. By yielding no results on certain inputs, this at the same time constrains the syntactic search space. The following descriptions there-

Proceedings of the ACL 2014 Workshop on Semantic Parsing, pages 12–16,
Baltimore, Maryland USA, June 26 2014.

fore relate to the implementation of the SEMSPLIT function.

Algorithm 1 A general splitting algorithm. $\mathcal{C}$ is the set of binary CCG combinators. The SEMSPLIT function returns possible splits of a meaning representation according to the reverse application of a combinator.

function SPLIT(x, z)
  if $|x| = 1$ **then**
    return $\{(x, z)\}$
  else
    $G \leftarrow \emptyset$
    for $0 < i \leq |x| - 1$ and $c \in \mathcal{C}$ **do**
      $l \leftarrow x_0 \ldots x_{i-1}$
      $r \leftarrow x_i \ldots x_{|x|-1}$
      $S \leftarrow$ SEMSPLIT(c, z)
      for $(f, g) \in S$ **do**
        $G \leftarrow G \cup$ SPLIT(l, f)
          $\cup$ SPLIT(r, g)
      end for
    end for
    return G
  end if
end function

2.1 Alignment consistency

The first heuristic we introduce is borrowed from the field of statistical machine translation. There, alignments between words of two languages are used to identify corresponding phrase pairs, as in the well-known GHKM algorithm (Galley et al., 2004). In order to apply the same strategy to meaning representations, we represent them as their abstract syntax trees. Following Li et al. (2013), we can then align words in the sentence and nodes in the meaning representation to identify components that correspond to each other.

This allows us to impose an extra constraint on the generation of splits: We require that nodes in f not be aligned to any words in the right sentence-half r, and conversely, that nodes in g not be aligned to words in l.

Alignment consistency helps the search to focus on more plausible splits by grouping elements of the meaning representation with the words that evoked them. However, by itself it does not significantly limit the search space, as it is still possible to extract infinitely many semantic splits from any sentence at any splitting index.

Example: Given the word-to-meaning alignment from Figure 1, a split that is excluded by the alignment criterion is: (Vincent : $\lambda g.\exists x.\exists y.\text{vincent}(x) \wedge \text{love}(x, y) \wedge g(y)$), (loves Mia : $\lambda y.\text{mia}(y)$). This is because the node "love" (in f) is aligned to the word "loves" (in r).

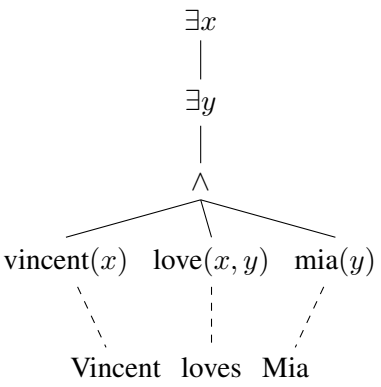

Figure 1: An example word-to-meaning alignment. Splits across any of the alignment edges are prohibited. E. g., we cannot produce a split whose meaning representation contains both vincent and mia.

2.2 Single-branching recursive splitting

The second heuristic is best described as a search strategy over possible semantic splits. In the following presentation, we presume that alignment consistency is being enforced. Again, it is helpful to view the meaning representation as an abstract syntax tree.

Recall that our goal is to find two expressions f, g to be associated with the sentence halves l, r. In a special case, this problem is easily solved: If we can find some *split node* X which governs all nodes aligned to words in r, but no nodes aligned to words in l, we can simply extract the sub-tree rooted at X and replace it with a variable. E. g., $z = a(bc)$ can be split into $f = \lambda x.a(xc)$ and $g = b$, which can be recombined by application.

However, requiring the existence of exactly two such contiguous components can be overly restrictive, as Figure 2 illustrates. Instead, we say that we decompose z into a hierarchy of components, with a split node at the root of each component. These components are labelled as f- and g-components in an alternating fashion.

In this hierarchy, the members of an f-component are not allowed to have alignments to words in l. A corresponding requirement holds for

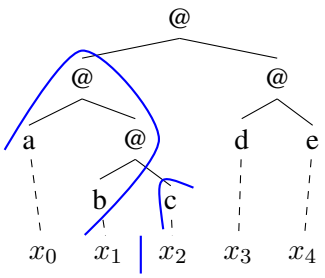

Figure 2: Illustration of single-branching recursion: Assume that the leaves of the meaning representation $a(bc)(de)$ are aligned as given to the words $x_0 \ldots x_4$, and that we wish to split the sentence at index 2. The indicated split partitions the meaning representation into three hierarchically nested components and yields $f = \lambda x.xc(de)$ and $g = \lambda y.a(by)$, which can be recombined using application.

g-components.

The *single-branching* criterion states that all split nodes lie on a common path from the root, or in other words, every component is the parent of at most one sub-component.

In comparison to more flexible strategies, single-branching recursive splitting has the advantage of requiring a minimum of additionally generated structure. For every component, we only need to introduce one new bound variable for the body plus one for every variable that occurs free under the split node.

Together with the alignment consistency criterion, single-branching recursive splitting limits the search space sufficiently to make a full search tractable in many cases.

2.3 Other heuristics

The following heuristics seem promising but are left to be explored in future work.

Min-cut splitting In this strategy, we place no restriction on which split nodes are chosen. Instead, we require that the overall count of split nodes is minimal, which is equivalent to saying that the edges cut by the split form a minimum cut separating the nodes aligned to the left and right halves of the sentence, respectively. This strategy has the advantage of being able to handle any alignment/split point combination, but requires a more complex splitting pattern and thus more additional structure than single-branching recursion.

Syntax-driven splitting Since CCG is based on the assumption that semantic and syntactic derivations are isomorphic, we might use syntactic annotations to guide the search of the derivation space and only consider splits along constituent boundaries. Syntactic annotations might be present in the data or generated by standard tools. However, initial tests have shown that this requirement is too restrictive when combined with our two main heuristics.

Obviously, an effective combination of heuristics needs to be found. One particular configuration which seems promising is alignment consistency combined with min-cut splitting (which is more permissive than single-branching recursion) and syntax-driven splitting (which adds an extra restriction).

3 Discussion

We present some empirical observations about the behaviour of the above-mentioned heuristics. Our observations are based on a grammar extracted from the GMB. A formal evaluation of our system in the context of a full semantic parsing system is left for future work.

3.1 Implementation

Currently, our system implements single-branching recursive splitting along with alignment consistency. We extracted the word-to-meaning alignments from the CCG derivations annotated in the GMB, but kept only alignment edges to predicate nodes. Sentence grammars were extracted by generating an initial item for each sentence and feeding it to the SPLIT procedure.

In addition to alignment consistency and single-branching recursion, we enforce three simple criteria to rule out highly implausible items: The count of arrows in an extracted meaning representation's type is limited to eight, the number of split nodes is limited to three, and the number of free variables in extracted components is also limited to three.

A major limitation of our implementation is that it currently only considers the application combinator during splitting. We take this as a main reason for the limited granularity we observe in our output. Generalisation of the splitting implementation to other combinators such as composition is therefore necessary before performing any serious evaluation.

3.2 Manual inspection

Manual inspection of the generated grammars leads to two general observations.

Firstly, many single-word items present in the CCG annotations of the GMB are recovered. While this behaviour is not required, it is encouraging, as these items exhibit a relatively simple structure and would be expected to generalise well.

At the same time, many multi-word phrases remain in the data that cannot be split further, and are therefore unlikely to generalise well. We have identified two likely causes for this phenomenon: The missing implementation of a composition combinator, and coarse alignments.

Composition splits would enable the splitting of items which do not decompose well (i. e., do not pass the search heuristics in use) under the application combinator. Since composition occurs frequently in GMB derivations, it is to be expected that its lack noticeably impoverishes the quality of the extracted grammar.

The extraction of alignments currently in use in our implementation works by retracing the CCG derivations annotated in the GMB, and thus establishing a link between a word and the set of meaning representation elements introduced by it. However, our current implementation only handles the most common derivation nodes and otherwise cuts this retracing process short, making alignments to the entire phrase governed by an intermediate node. This may cause the corresponding part of the search to be pruned due to a search space explosion. We plan to investigate using a statistical alignment tool instead, possibly using supplementary heuristics for determining aligned words and nodes. As an additional advantage, this would remove the need for annotated CCG derivations in the data.

3.3 Statistical observations

From the total 47 230 sentences present in the GMB, our software was able to extract a sentence grammar for 43 046 sentences. Failures occurred either because processing took longer than 20 minutes, because the count of items extracted for a single sentence surpassed 10 000, or due to processing errors.

On average, 825 items were extracted per sentence with a median of 268. After removing duplicate items, the combined grammar for the whole GMB consisted of about 32 million items. While the running time of splitting is still exponential and gets out of hand on some examples, most sentences are processed within seconds.

Single-word items were extracted for 46% of word occurrences. Ideally, we would like to obtain single-word items for as many words as possible, as those items have the highest potential to generalise to unseen data. For those occurrences where no single-word item was extracted, the median length of the smallest extracted item was 12, with a maximum of 49.

4 Conclusion

We have presented a method for bringing the induction of semantic CCGs to a larger scale than has been feasible so far. Using the heuristics of alignment consistency and single-branching recursive splitting, we are able to extract a grammar from the full GMB. Our observations suggest a mixed outcome: We obtain desirable single-word items for only about half of all word occurrences. However, due to the incompleteness of the implementation and the lack of a formal evaluation, these observations do not yet permit any conclusions. In future work, we will address both of these shortcomings.

5 Final remarks

The software implementing the presented functionality is available for download[1].

This work has been supported by the German Research Foundation (DFG) as part of the CINACS international graduate research group.

References

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of LREC'12*, Istanbul, Turkey.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of ACL 2013*, Sofia, Bulgaria.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL 2004*, Boston, Massachusetts, USA.

[1] `http://nats-www.informatik.uni-hamburg.de/User/SebastianBeschke`

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of EMNLP 2010*, Cambridge, Massachusetts, USA.

Peng Li, Yang Liu, and Maosong Sun. 2013. An extended GHKM algorithm for inducing λ-scfg. In *Proceedings of AAAI 2013*, Bellevue, Washington, USA.

Mark Steedman. 2001. *The Syntactic Process*. MIT Press, January.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of AAAI-96*, Portland, Oregon, USA.