An Automatic Method for Building a Data-to-Text Generator

Sina Zarrieß **Kyle Richardson**
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart, Germany
sina.zarriess,kyle@ims.uni-stuttgart.de

1 Introduction

We describe our contribution to the Generating from Knowledge Bases (KBgen) challenge. Our system is learned in a bottom-up fashion, by inducing a probabilistic grammar that represents alignments between strings and parts of a knowledge graph. From these alignments, we extract information about the linearization and lexical choices associated with the target knowledge base, and build a simple generate-and-rank system in the style of (Langkilde and Knight, 1998).[1]

2 Semantic Parsing and Alignments

A first step in building our generator involves finding alignments between phrases and their groundings in the target knowledge base. Figure 1 shows an example sentence from training paired with the corresponding triple relations. A partial lexicon is provided, indicating the relation between a subset of words and their concepts.

Using the triples, we automatically construct a probabilistic context-free grammar (PCFG) by converting these triples to rewrite rules, using ideas from (Börschinger et al., 2011). The right hand side of the rules represent the constituents of the triples in all orders (initially with a uniform probability) since the linear realization of a triple relation in the language might vary. This is rewritten back to each of its constituents to allow for interaction with other concepts that satisfy further domain relations. Individual concepts, represented in the grammar as preterminals, are assigned to the associated words in the lexicon, while unknown words are mapped to all concepts with equal probability.

Following (Börschinger et al., 2011), sentences in the training are restricted to analyses corresponding to their gold triple relations, and the inside-outside algorithm, a variant of EM, is applied to learn the corresponding PCFG parameters. In intuitive terms, the learning algorithm iteratively maximizes the probability of rules that derive the correct triple relations in training, looking over several examples. For example, the unknown word *are* in Figure 1 is learned to indicate the relation *object* since it often occurs in training contexts where this relation occurs between entities surrounding it. The syntax of how triples are composed and ordered in the language is also learned in an analogous way.

We annotate the development data with the most probable trees predicted by the PCFG. Figure 1 shows the viterbi parse for the given sentence after training. Bascially, it defines a spanning tree for the knowledge graph given in the input. Each ternary subtree indicates a triple relation detected in the sentence, and the root node of this subtree specifies the head (or first argument) of the triple relation. Note that some triple relations are not found (e.g. the *base* relation), since they are implicit in the language.

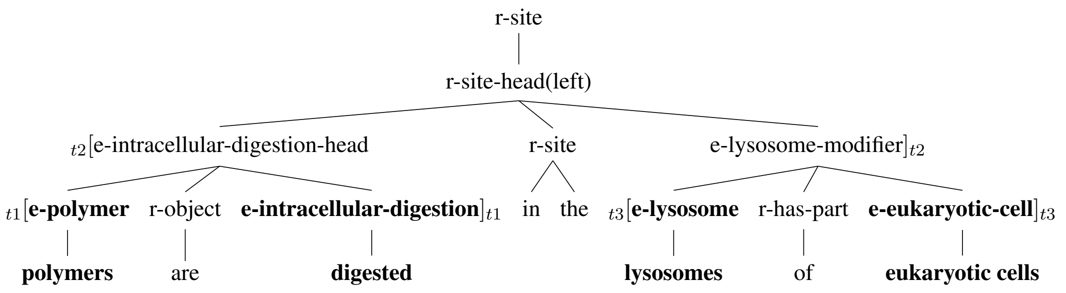
3 Grammar and Lexicon Extraction

The viterbi trees learned in the previous step for the development set are used for constructing a generation grammar that specifies the mapping between triples and surface realizations. The tree in Figure 1 indicates, for example, that the second argument of an *object* relation can be realized to the left of the relation and its first argument. We also learn that the *site* relation can be lexicalized as the phrase *in the*.

Grammar A non-lexical production in a tree corresponds to a surface realization of an input triple. We iterate over all productions of the trees in the development data and aggregate counts of concept orderings over all instances of a relation. We distinguish preterminal concepts ($preterm$) that map to a lexical entry and nonterminal concepts ($nonterm$) that embed another subtree. Example (1) and (2) illustrate rules that apply to the tree in Figure 1 for ordering the *site* and *object* relation. The rule for *object* introduces ambiguity. Note that (2-a) deletes the *object* phrase.

(1) $Input$: (A$_{nonterm}$,r-site,B$_{nonterm}$)

[1]This work was supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) in SFB 732 *Incremental Specification in Context*, project D2 (PI: Jonas Kuhn). We thank Thomas Müller for help with the language models.



```
:TRIPLES ( (|Intracellular-Digestion36204| |object| |Polymer36220|)
           (|Intracellular-Digestion36204| |base| |Eukaryotic-Cell36203|)
           (|Intracellular-Digestion36204| |site| |Lysosome36202|)
           (|Eukaryotic-Cell36203| |has-part| |Lysosome36202|))
```

Figure 1: A semantic parse (top) learned from the the triples (bottom) provided during training. Words/concepts in bold are known from the lexicon, while the rest is learned along with the syntax of triple combination. Triple instances in the tree are marked with square brackets.

	a.	*rhs*: $A_{nonterm}$ r-site $B_{nonterm}$;	1.0
(2)		*Input*: ($A_{preterm}$,r-object,$B_{preterm}$)	
	a.	*rhs*: $A_{preterm}$ $B_{preterm}$;	0.33
	b.	*rhs*: $B_{preterm}$ r-object $A_{preterm}$;	0.3
	c.	...	

Lexicon For each preterminal in the trees, we extract its lexical span in the surface sentence. For instance, we extract 15 phrases as possible realizations for the *base* relation (e.g. "for the", "in the", "of a", "from a"). This is merged with the provided lexicon, to create an expanded lexicon.

4 Generation Pipeline

The main idea of the generator is to produce a (possibly large) set of output candidates licensed by the grammar and the lexicon. In a final step, these candidates are ranked with the help of a language model, a common approach in statistical generation (Langkilde and Knight, 1998). We train our language model on the GENIA corpus (Ohta et al., 2002). Below is our overall pipeline.

1. compute all spanning trees licensed by the input triples
2. for each spanning tree from step 1, compute all surface linearizations licensed by the generation grammar
3. for each linearized tree from step 2, compute all surface sentences licensed by the expanded lexicon
4. rank surface candidates with a language model

The set of spanning trees produced in step 1 is typically small. We prune the set of possible linearizations based on the counts in the generation grammar, and consider only the two most likely orderings for each input triple. We also prune the set of possible lexicalizations and refine it with some linguistic constraints described below.

Linguistic Constraints The viterbi trees learned in the alignment step do not capture any linguistic properties of the sentences in terms of morpho-syntactic categories. As a consequence, most of the output candidates coming from step 3 are ungrammatical. Ungrammatical sentences do not necessarily get low scores from the language model as it captures local relations between neighbouring words. We introduce some simple candidate filters to ensure some basic linguistic constraints. With the help of the lexicon and some heuristics, we tag all lexical entries containing a finite verb. In step 3, we filter all candidates that a) have no finite verb, b) have a finite verb as the first or last word, c) realize two finite verbs next to each other.

Conclusion We explore the use of Semantic Parsing techniques, coupled with corpus-based generation. We expect that our prototype would benefit from further development of the linguistic components, given that it is built with minimal resources.

References

Börschinger, Benjamin, Jones, Bevan K, Johnson, Mark. 2011. Reducing Grounded Learning to Grammatical Inference In *Proc. of EMNLP'11*, pages 1416-1425.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proc. of ACL 1998*, pages 704–710.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The genia corpus: an annotated research abstract corpus in molecular biology domain. In *Proc. of HLT '02*, pages 82–86.