Supplementary Notes: Hybrid Simplification using Deep Semantics and Machine Translation

Shashi Narayan
Université de Lorraine, LORIA
Villers-lès-Nancy, F-54600, France
shashi.narayan@loria.fr

Claire Gardent
CNRS, LORIA, UMR 7503
Vandoeuvre-lès-Nancy, F-54500, France
claire.gardent@loria.fr

1 EM algorithm: Estimating the parameters

We use the EM algorithm (Dempster et al., 1977) to estimate our split and deletion model parameters. For an efficient implementation of EM algorithm, we follow the work of Yamada and Knight (2001) and Zhu et al. (2010); and build training graphs (Figure 1) from the pair of complex and simple sentences pairs in the training data. Yamada and Knight (2001) used it for a syntax based translation model whereas Zhu et al. (2010), later, used it to learn a tree based translation model for sentence simplification.

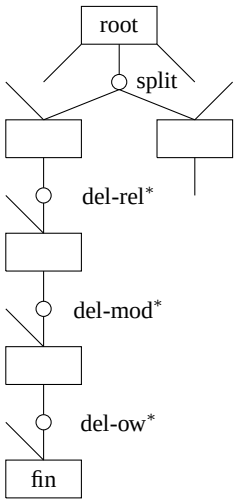

Figure 1: An example training graph

Our EM algorithm (as depicted in Algorithm 1, 2 and 3) starts with building training graphs from the training data. Each training graph represents a complex-simple sentences pair and consists of two types of nodes: major nodes (M-nodes) and operation nodes (O-nodes). An M-node contains the DRS representation D_c of complex sentence c and the associated simple sentence(s) s_i while O-nodes determines split and deletion operations on their parent M-node. Only root M-node is considered for split operations. For example, given the root M-node $(\mathrm{D}_c, (s_1, s_2))$, multiple successful split O-nodes will be created, each one further creating two M-nodes (D_{c1}, s_1) and (D_{c2}, s_2). For the training pair (c, s), the root M-node (D_c, s) is followed by a single split O-node producing an M-node (D_c, s) and counting all split candidates in D_c for failed split. The M-nodes created after split operations are then tried for multiple deletion operations of relations, modifiers and OW respectively. Each deletion candidate creates a deletion O-node marking successful or failed deletion of the candidate and a result M-node. The deletion process continues on the result M-node until there is a deletion candidate left to process. The governing criteria of the supervised construction of the training graph is that at each step it tries to minimize the Levenshtein edit distance between the complex and the simple sentences.

We initialize our probability with the uniform distribution, i.e., 0.5 because all our features are binary. The EM algorithms iterates over training graphs; estimating inside and outside probabilities and counting model features from O-nodes and

Algorithm 1 EM Algorithm

```
Construct training graph for each (complex, simple) sentence(s) pairs in the training data.
Initialize all probability tables using the uniform distribution.
for multiple iterations do
    Reset all count tables to zero
    for each training graph do
        Calculate Inside (β) Probabilities
        Calculate Outside (α) Probabilities
        Update count for each operation features in each O-node n of the training graph:
        count = count + (α_n * β_n / β_root)
    end for
end for
```

Algorithm 2 Calculate Inside (β) Probability

for each node n from the bottom to the root of training graph **do**
  if node n is a final M-node **then**
    $\beta_n = 1$
  else if node n is an O-node **then**
    $\beta_n = \prod_{n' \in child(n)} \beta_{n'}$
  else
    $\beta_n = \sum_{n' \in child(n)} p(oper|n) * \beta_{n'}$
  end if
end for

Algorithm 3 Calculate Outside (α) Probability

for each node n from the root to the bottom of training graph **do**
  if node n is the root M-node **then**
    $\alpha_n = 1$
  else if node n is an O-node **then**
    $\alpha_n = p(oper|parent_n) * \alpha_{parent_n}$
  else
    $\alpha_n = \sum_{n' \in parent(n)} \alpha_{n'} \prod_{\substack{n'' \in child(n') \\ n'' \neq n}} \beta_{n''}$
  end if
end for

updating our probability tables. The outside probability of the root M-node is 1 as it is the starting point. We assume that our supervised construction of the training graph is perfect and hence all final M-nodes have been assigned the inside probability of 1. EM algorithm (as depicted in Algorithm 1, 2 and 3) iterates over training graphs and exercises all possible paths. We refer the reader to (Yamada and Knight, 2001) for more detail. Few examples including boxer graphs and corresponding training graphs are provided as the supplementary material (examples/training).

2 Decoding

We explore the decoding graph similar to the training graph but in a greedy approach always picking the choice with maximal probability. Given a complex input sentence c, a split O-node will be selected corresponding to the decision of whether to split and where to split. Next, deletion O-nodes are selected indicating whether or not to drop each of the deletion candidate. The DRS associated with the final M-node D_{fin} is then mapped to a simplified sentence s'_{fin} which is further simplified using the phrase-based machine translation system to produce the final simplified sentence s_{simple}. Few examples including boxer graphs and corresponding decoding graphs are provided as the supplementary material (examples/test).

References

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 523–530. Association for Computational Linguistics.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 1353–1361, Stroudsburg, PA, USA. Association for Computational Linguistics.