Word-based translation with neural networks

Vova Zaytsev, David Chiang

April 9, 2014

Background

- According to (Devlin, to appear) neural networks joint models (NNJM) can be used as features in MT systems highly successfully and significantly improve BLEU score.
- In this model, feed-forward neural networks used to approximate the probability of target sentence T conditioned on source sentence S:

$$P(T|S) \approx \prod_{i=1}^{|T|} P(t_i|t_{i-1}, \ldots, t_{i-n+1}, s_{A_i-k}, \ldots, s_{A_i+k}) \quad (1)$$

  NNJM augments an n-gram target language model with an m-word source window. Here, target decomposed in the standard n-gram way – each target word t_i is conditioned on the previous $n-1$ target words. Source context vector is defined using alignment A, where each target word t_i is affiliated with exactly one source word at index A_i.
- Although the model is quite simple, it yields extremely strong empirical results. On NIST OpenMT12 Arabic-English NNJM showed +3.0 BLEU on top of feature-rich baseline and +6.3 BLEU on top of a simpler baseline, equivalent to Chiangs (2007).
- Since, this model is claimed to be so effective, we wonder whether it could be the basis for a standalone model. Such a model might or might not outperform a phrase-based model, but would be orders of magnitude smaller

1 DR Goal

The goal of this direct research project is to take NNJM model, develop word-based MT system based on standalone NNJM features and compare it against phrase-based baseline MT system.

2 Method

2.1 Generative Story

Given a source sentence s_1, s_2, ..., we generate target sentence t_1, t_2, ..., as follows:

1. Initialize $i \leftarrow 1$, $j \leftarrow 0$
2. Choose Δj from range $-h, \ldots, h$ with probability $P_d(\Delta j | t_{i-1}, \ldots, t_{i-n+1}, s_{j-k}, \ldots, s_{j+k})$
3. Update $j \leftarrow j + \Delta j$
4. Generate t_i with probability $P_t(t_i | t_{i-1}, \ldots, t_{i-n+1}, s_{j-k}, \ldots, s_{j+k})$
5. If $t_i =$ <\trg> stop. Otherwise, increment i and goto step 2.

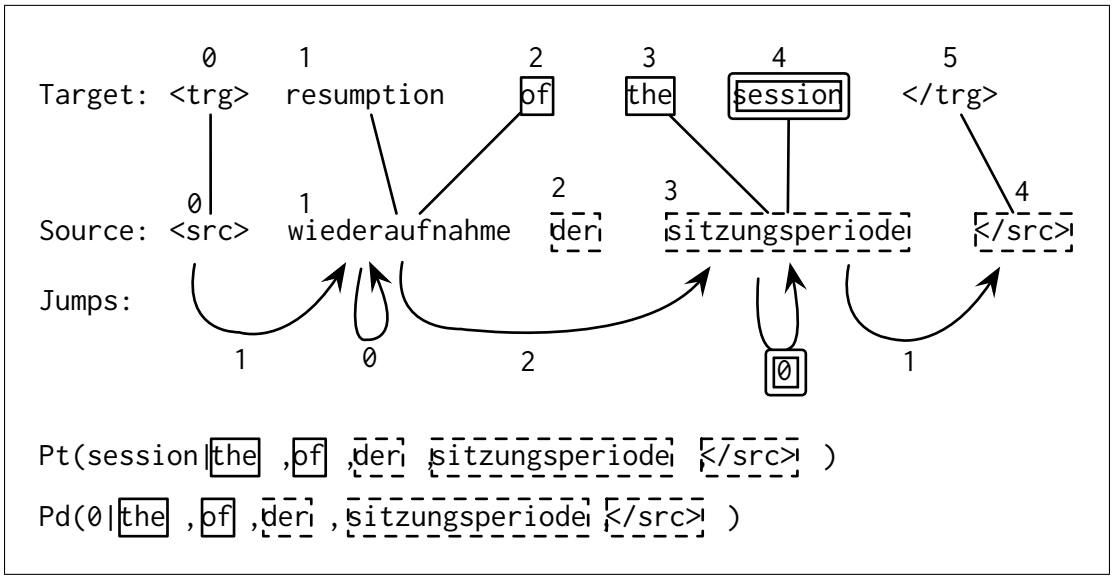

Figure 1: Word alignment and two extracted examples for translation and distortion models. Size of source context vector is $m = 3$ and length of translation history is 2 ($n = 3$).

2.2 Word Alignment Post-Processing

We generated our initial word alignments using GIZA++ and applied Affiliation Heuristic (Devlin, to appear) to make each target word to be affiliated with exactly one non-NULL source word. Devlin describes this heuristics as follows:

- If t_i aligns to exactly one source word, A_i is the index of the word it aligns to.
- If t_i align to multiple source words, A_i is the index of the aligned word in the middle (round down).
- If t_i is unaligned, we inherit its affiliation from the closest aligned word, starting with the right.

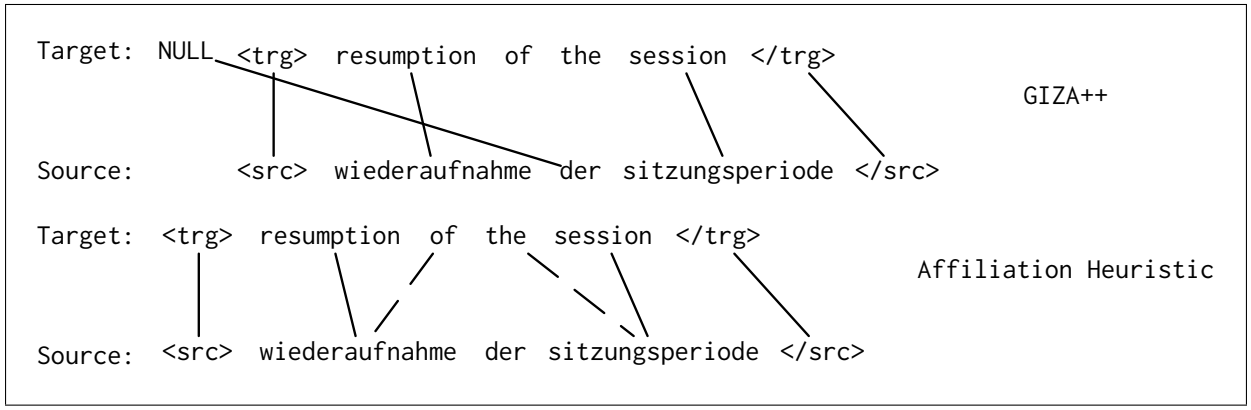

Figure 2: Original alignment (top) and alignment after applying Affiliation Heuristic (bottom).

2.3 Training

To train distortion and translation models, we prepared examples from post-processed aliments and then used NPLM toolkit (Vaswani et al., 2013) to learn joint probabilities. For each model, we generated input and output vocabulary and all out-of-vocabulary words were mapped to special `<trg_unk>` and `<src_unk>` symbols.

The input vocabulary was generated by union of vocabularies of both target and source languages because in our model, the input information includes target translation history as well as source context vector. An example of the NPLM context for a German-English parallel sentence is given in Figure 1.

2.4 Decoding

The decoding process is very similar to phrase-based decoding, except that there is no iteration over phrase sizes. Our state contains translation covering, source sentence pointer, translation history and partial score value. We used BEAM search to find the best translation for given source sentence. Here is a pseudo-code for used hypothesis expansion algorithm:

Listing 1: Hypothesis Expansion

```
function ExpandHypothesis(prev_hyp={covering, s_i, T, score})
    if covering is full:
        return
    for new_s_i in NOT_TRANSLATED(covering):
        jump = new_s_i - s_i
        source_context = [S[new_s_i-k], ..., S[new_s_i+k]]
        translation_history = T[n:]
        d_score = DMODEL[jump, translation_history, source_context]
        for new_t_word in OUTPUT_VOCAB:
            t_score = TMODEL[new_t_word, translation_history, source_context]
            new_T = update(T, new_t_word)
            new_covering = update(covering, new_s_i)
            new_score = score * t_score * d_score
            CreateHypothesis({new_covering, new_s_i, new_T, new_score})
```

3 Preliminary Results

3.1 Parallel Corpus

Sample of German-English Europarl:

	English	German
Train (words)	1072310	1006915
Train	49690	
Dev	781	
Test	792	

3.2 Baseline Moses Configuration

3.3 NPLM Parameters

3.4 Translation Examples

4 Next Steps

- Add fertility model
- Try model variations, e.g. reversed translation order and translation direction.

References

[1] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. *Fast and Robust Neural Network Joint Models for Statistical Machine Translation.*

[2] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. *Decoding with large-scale neural language models improves translation*

[3] David Chiang. 2007. *Hierarchical phrase-based translation.*