Topic Aware Neural Machine Translation

1 Motivation

Currently, neural machine translation approaches focus on a subword translation using an encoder-decoder architecture. In practise, multiple vastly different domains often have a high overlap in subwords, whereas their appropriate usage depends on the current context. Using a single softmax to retrieve the posterior distribution sets a hard limit on the modelling power.

Here we propose to focus on grouping the vocabulary into "topics" which could potentially help the NMT model focus better on which context is currently relevant and make the model explain semantically how it translated a word.

2 Word Level Topic Aware Neural Machine Translation

f_1^J is the source sequence and e_1^I is the target sequence. $topics_i$ is a set containing all possible topics for time-step i, $topics$ is then the set containing all possible aligned topics, i.e. $topics = \prod_{i \in \{1,2,\ldots,J\}} topics_i$, where the Cartesian product is applied between each set. In practise, we assume that the set of possible topics are equal between any two time steps, formally: $topics_i = topics_j$ for any i, j.

$$\begin{aligned} p(e_1^I | f_1^J) &= \sum_{t_1^J \in topics} p(e_1^I, t_1^J | f_1^J) \\ &= \sum_{t_1^J \in topics} \prod_{i=1}^{I} p(e_i, t_i | e_1^{i-1}, f_1^J, t_1^{i-1}) \end{aligned} \tag{1}$$

Assuming zero-order dependence on the topic, i.e. $p(t_i | t_1^{i-1}) = p(t_i)$:

$$\begin{aligned} &= \prod_{i=1}^{I} \sum_{t_i \in topics_i} p(e_i, t_i | e_1^{i-1}, f_1^J) \\ &= \prod_{i=1}^{I} \sum_{t_i \in topics_i} p(e_i | t_i, e_1^{i-1}, f_1^J) \cdot p(t_i | e_1^{i-1}, f_1^J) \end{aligned} \tag{2}$$

This can be modelled using a modified attention model. Semantically speaking, this model assumes that (for efficiency) the relevant topic can change within a single sentence and implies that there is a high number of topics containing a smaller amount of (sub)words.

3 Sentence Level Topic Aware Neural Machine Translation

The decomposition in section 2 can also be applied to the sentence level. Note that the set $topics$ contains all topics without alignment, in comparison to the word level approach.

$$\begin{aligned} p(e_1^I | f_1^J) &= \sum_{t \in topics} p(e_1^I, t | f_1^J) \\ &= \sum_{t \in topics} p(t | f_1^J) \cdot p(e_1^I | t, f_1^J) \\ &= \sum_{t \in topics} p(t | f_1^J) \cdot \prod_{i=1}^{I} p(e_i | e_1^{i-1}, t, f_1^J) \end{aligned} \tag{3}$$

Semantically speaking, this decomposition assumes that the relevant topic does not change over the translation of the sentence, implying that the number of topics is smaller but containing more words. In practise, the memory requirements of this model are (restrictively) high.

4 Mapping

As the occurrence of target (sub)words are dependent on the context in which they occur, we assume that they can be grouped together into topics, using a mapping $f : topics_i \rightarrow Pow(vocabulary)$, where $vocabulary$ is the full target vocabulary and Pow is the power set operator. Assuming that $p(e_i|t_i, e_1^{i-1}, f_1^J) = 0, \forall e_i \notin f(t_i)$, we only need to model a small subset of the vocabulary in equation 2, optimizing the model significantly. As f does not have any restrictions, we can group the topics in a semantically meaningful way, which also makes the model explainable during decoding.

Practically, we provide f as a hyper-parameter to the network. How f is determined is freely up to the user. Here are some exemplary ideas on how to group the target vocabulary:

- Sub-words that occur in similar counts
- Sub-words that occur in similar contexts in the training data
- Sub-words that have similar embeddings
- Sub-words that come from the same domain
- Use an external data source such as a knowledge graph (e.g. DBpedia, Wordnet) to determine groups based on the graph's properties

5 Implementation

The model with the mapping optimization has been implemented in RETURNN, with all operations being run on the GPU. The base model is based on the transformer, but the final layer is modelled using equation 2 instead of a simple softmax. The dependency on t is modelled using an embedding which is trained with the rest of the model.