The Document Context Language Model

Jacob Eisenstein

1 Basic Model

In the RNN language model of Mikolov (2010), we have

$$\mathbf{a}_n \leftarrow f(\mathbf{1}_{w_n}, \mathbf{a}_{n-1}) \tag{1}$$

$$w_{n+1} \sim \text{SoftMax}(\boldsymbol{V}\mathbf{a}_n), \tag{2}$$

where $f(\mathbf{1}_{w_n}, \mathbf{a}_{n-1}) = \sigma(\boldsymbol{U}[\mathbf{1}_{w_n}^\top, \mathbf{a}_{n-1}^\top]^\top)$; fancier RNNs such as LSTM and GRU use more complex gating systems in this function, but the basic idea is the same.

We can extend this model to incorporate a document context vector $\mathbf{h}_{i-1}$, by specifying

$$w_{i,n+1} \sim \text{SoftMax}(\boldsymbol{V}_s\mathbf{a}_{i,n} + \boldsymbol{V}_d\mathbf{h}_{i-1}), \tag{3}$$

where the vector $\mathbf{h}_{i-1}$ summarizes the document context at sentence $i-1$. There are several options for computing this vector, but it should depend on (a) the words in the sentence $\mathbf{w}_{i-1}$, and (b) the previous document context $\mathbf{h}_{i-2}$. Therefore, we propose another RNN-style model,

$$\mathbf{b}_i \leftarrow g(\mathbf{w}_i) \tag{4}$$

$$\mathbf{h}_i \leftarrow f(\mathbf{b}_i, \mathbf{h}_{i-1}), \tag{5}$$

where the function f again defines an RNN, LSTM, GRU, etc. The function $g(\dot{})$ might indicate a convolutional neural network, or could also indicate some kind of left-to-right structure. We can train the model on cross-entropy of the predictions w_{n+1}, or on a noise-contrastive estimation objective.

The first key empirical question (helpfully raised by Lingpeng) is to compare:

- The DCLM model as defined here
- The original RNNLM

w_{in}	word n in sentence i
$\mathbf{1}_{w_{in}}$	indicator vector for w_{in}
$\mathbf{a}_{in}$	RNNLM latent state at word n in sentence i
$f(\dot{})$	RNN recurrence function
$\mathbf{b}_i$	vector representation of sentence i
$g(\dot{})$	representation function for sentence i
$\mathbf{h}_i$	DCLM latent state at sentence i
$\langle \mathbf{w}_i^{(s)}, \mathbf{w}_i^{(t)} \rangle$	source-target bitext instance i (usually corresponding to a sentence in the source language)
d_i	discourse relation between sentence i and its predecessor

Table 1: Table of notation

- A sentence-sensitive version of the RNNLM, where we condition w_{n+1} the latent state $\mathbf{s}_n$ as well as the latent states $\mathbf{s}_{n'}$ and $\mathbf{s}_{n''}$, with n and n'' indexing the end of the previous two sentences.

We will perform perplexity evaluations, beginning with the PTB evaluation from Mikolov (2010).

2 Extensions

Having established the basic DCLM framework, we now consider some extensions that may make it more relevant to both discourse and machine translation.

2.1 Bilingual DCLM

Suppose we have some aligned sentences, $\{\langle \mathbf{w}_i^{(t)}, \mathbf{w}_i^{(s)} \rangle\}_{i \in 1 \ldots N}$. We can train a bilingual DCLM, in which we use the latent state on $\mathbf{w}^{(s)}$ to predict the words in $\mathbf{w}^{(t)}$. Specifically, we have,

$$\mathbf{a}^{(t)}_{i,n} \leftarrow f(\mathbf{1}_{w^{(t)}_{i,n}}, \mathbf{a}^{(t)}_{i,n-1}) \tag{6}$$

$$\mathbf{h}_{i,n} \leftarrow f(g(\mathbf{w}^{(s)}_i), \mathbf{h}_{i-1}) \tag{7}$$

$$w^{(t)}_{i,n} \sim \text{SoftMax}(\boldsymbol{V}_s \mathbf{a}^{(t)}_{i,n} + \boldsymbol{V}_d \mathbf{h}_{i-1}). \tag{8}$$

We keep a local RNNLM for the target language, but also include the source-side document context vector $\mathbf{h}_{i-1}$. We may also train want to train this model on source-language text, as described in the basic DCLM. A key advantage of this model is that it brings in document-level information without requiring document-level decoding: all the document level information is extracted from the source side. The score $\boldsymbol{V}_s \mathbf{a}^{(t)}_{i,n} + \boldsymbol{V}_d \mathbf{h}_{i-1}$ can simply be handed off to a decoder, which might involve more complex translation components.

2.2 Discourse

Suppose we have discourse annotations $d_i \in \mathcal{D}$, which might come from the Penn Discourse Treebank. We can use these as an additional source of supervision, in several different ways.

Discourse-based projection We can use the discourse relations to transform the representation of the previous sentence $\mathbf{h}_{i-1}$,

$$w_{i,n+1} \sim \text{SoftMax}(\boldsymbol{V}_s \mathbf{a}_{i,n} + \boldsymbol{V}_d \tanh(\boldsymbol{R}_{d_i} \mathbf{h}_{i-1})), \tag{9}$$

where $\boldsymbol{R}_{d_i}$ is a $K \times K$ matrix. At decoding time, we can run a discourse relation classifier to obtain $\hat{d}_i$; we should probably train on these predicted discourse relations too. This captures the idea that discourse relations determine how the document context affects the words that are generated.

Discourse-based recurrence We can apply a similar idea at the RNN level, so that

$$\mathbf{h}_i \leftarrow f(\mathbf{b}_i, \tanh(\boldsymbol{R}_{d_i} \mathbf{h}_{i-1})). \tag{10}$$

Here, the role of discourse is to shape the evolution of meaning through the document. It's hard to have intuitions about whether this would work better than applying the discourse-driven transformation directly before generating the next word, but it would be easy to try. Since we have PDTB annotations on the same data used in the Mikolov RNNLM, we can compare gold-standard relations to predictions from our system.

Discourse as an auxiliary task Finally, we may treat discourse relation prediction as an auxiliary task, on the hope that discourse-informed vectors $\mathbf{h}$ will also be more useful for translation or word prediction. Specifically, we can add something like,

$$d_i \sim \text{SoftMax}(\mathbf{h}_i^\top \boldsymbol{\Theta} \mathbf{h}_{i-1}), \tag{11}$$

or

$$d_i \sim \text{MLP}([\mathbf{h}_i^\top, \mathbf{h}_{i-1}^\top]), \tag{12}$$

where MLP indicates a multilayer perceptron. In this case, it's a little harder to see how to incorporate discourse at decoding time, but maybe the document context vectors $\mathbf{h}$ will just be better thanks to this auxiliary task. (In the previous document I talked about a "retrofitting" idea, but that seems too complicated to worry about now.)

2.3 Other context

While a left-to-right order matches human reading, nothing in the model requires this order for decoding. We can easily do:

$$w_{i,n+1} \sim \text{SoftMax}(\boldsymbol{V}_s \mathbf{a}_{i,n} + \boldsymbol{V}_{d,\ell} \mathbf{h}_{i-1} + \boldsymbol{V}_{d,r} \mathbf{h}_{i+1}). \tag{13}$$

This works for either the monolingual or bilingual models.

We can also employ rhetorical structure theory. Assume sentence i is the satellite of sentence $\Gamma(i)$. Then we can extend the model further, to,

$$\begin{aligned} w_{i,n+1} \sim \text{SoftMax}(&\boldsymbol{V}_s \mathbf{a}_{i,n} + \boldsymbol{V}_{d,\ell} \mathbf{h}_{i-1} \\ &+ \boldsymbol{V}_{d,r} \mathbf{h}_{i+1} + \boldsymbol{V}_{rst} \mathbf{h}_{\Gamma(i)}). \end{aligned} \tag{14}$$

3 Decoding

The bilingual model does not require any change to the decoding algorithm: we just hand off the scoring function to MERT or whatever. The hope is that the document context obtained by this model will help focus the language model on the target side, potentially getting correct tense or definiteness information from previous sentences.

However, the bilingual model does not ensure that the target translation is discourse coherent. For this, we need to condition $\mathbf{w}^{(t)}_i$ on an $\mathbf{h}_{i-1}$ that is computed on the target side. Recall that $\mathbf{h}_{i-1}$ is computed recursively from $\mathbf{w}_{i-1}$ and $\mathbf{h}_{i-2}$; it therefore summarizes the state of the target side translation up to sentence $i-1$. The score contributed by $\mathbf{w}_i$ is equal to,

$$\sum_n \log P(w_{i,n} \mid w_{i,1:n-1}, \mathbf{h}_{i-1}), \quad (15)$$

which can be computed by the feedforward network. We would like to choose the document-level translation with the best global score, $\sum_i \sum_n \log P(w_{i,n} \mid w_{i,1:n-1}, \mathbf{h}_{i-1})$, but this will not be possible in the general case. There are two possible simplifications.

3.1 Viterbi decoding

If we decouple the target-side document context vector $\mathbf{h}_i$ from $\mathbf{h}_{i-1}$, then things become considerably simpler. We now compute $\mathbf{h}_{i-1} = g(\mathbf{w}_{i-1})$, where $g()$ can be either a convolutional or recurrent network. Now imagine we can enumerate all possible translations of sentence i, reusing the superscript now to index the translation k, $\mathbf{w}_i^{(k)}$. Then the score of the best translation $\mathbf{w}_{1:i}$ is given by,

$$\psi_i = \max_k \psi_i^{(k)} \quad (16)$$

$$\psi_i^{(k)} = \max_\ell \log P(\mathbf{w}_i^{(k)} \mid \mathbf{w}_{i-1}^{(\ell)}) + \psi_{i-1}^{(\ell)}, \quad (17)$$

which is the familiar Viterbi recurrence. We can thus decode exactly in this case (assuming an infinite number of translation hypotheses at each sentence), and can still combine with the bilingual model to incorporate fully document-level context from the source side. In practice, we will need k-best lists from the per-sentence translator, and may wish to maximize diversity in these lists. Note that unlike the purely source-side model, we do enforce local coherence on the target side, but we do not propagate information across the entire document.

3.2 (Randomized) beam search

If we want to maintain fully document-level context on the target side, the proposed Viterbi algorithm no longer applies. We can instead take a (randomized) beam search approach. Consider a set of document-level hypotheses, where each hypothesis is represented by a tuple $\langle \mathbf{h}_i^{(k)}, \psi_i^{(k)} \rangle$. We can then step forward as,

$$\langle \mathbf{h}_i^{(k)}, \psi_i^{(k)} \rangle = \bigoplus_q \langle \mathbf{h}_{i-1}^{(\ell)}, \psi_{i-1}^{(\ell)} + \log P(\mathbf{w}_i^{(k)} \mid \mathbf{h}_{i-1}^{(\ell)}) \rangle \quad (18)$$

$$q_i^{(k)} \propto \exp(\psi_i^{(k)}), \quad (19)$$

with $\bigoplus_q$ serving as a general notation, which could indicate a max operation, random sampling, or perhaps even some kind of weighted average. We can use any sequential Monte Carlo or beam search algorithm, for example resampling at each time step, or taking the top K hypotheses. In principle we could also run this process in the backwards direction and then add both scores into the MERT decoder; or we could use something like Forward Filtering Backward Sampling (FFBS), which is a two-pass sequential Monte Carlo algorithm that would draw samples from the joint distribution $P(\mathbf{w}_{1:N}^{(t)} \mid \mathbf{w}^{(s)})$.

References

Tomas Mikolov. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.