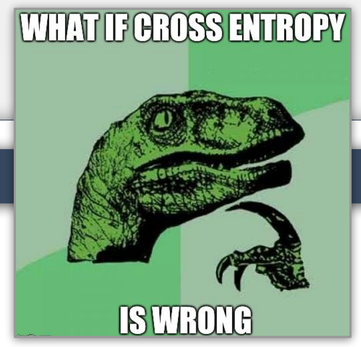
Similarity-Based Reconstruction Loss for Meaning Representation

Olga Kovaleva, Anna Rumshisky, Alexey Romanov
Department of Computer Science | University of Massachusetts Lowell
{okovaleva,arum,aromanov}@cs.uml.edu



Problem

- A great majority of deep learning models used at present for state-of-the-art machine translation, question answering, summarization, and dialogue generation employ an encoder-decoder architecture.
- Encoder-decoder architectures tend to rely on cross-entropy reconstruction loss to generate the target output.
- The standard cross-entropy loss penalizes the model whenever it fails to produce the exact word from the ground truth data used for training.

$$\mathcal{L} = -\sum_{i=1}^{V} y_i \log p_i$$

- In many NLP tasks that deal with generating text from semantic representation, **recovering the exact word is not necessarily optimal**, and often generating a near-synonym or just a semantically close word is nearly as good or even better.
- In some cases, it doesn't make sense at all! E.g. dialogue generation, summarization, etc.

Approach

- If the model fails to replicate ground truth, but generates reasonably similar output, it should not be penalized to the same extent.

SOLUTION: Weigh the penalty **by the similarity** of the generated outputs to the ground truth in the embedding space.

Weighted similarity loss

$$\mathcal{L} = -\sum_{i=1}^{V} \text{sim}(y_t, y_i) p_i$$

where ***p*** i is the softmax probability over vocabulary size ***V***, −1 ≤ sim ≤ 1 is the similarity between the tokens embeddings vectors, $\boldsymbol{y_t}$ and $\boldsymbol{y_i}$ are the ground-truth token and the predicted token, respectively.

Weighted cross-entropy loss

$$\mathcal{L} = -\sum_{i=1}^{V} \text{sim}(y_t, y_i) \log p_i$$

Here the optimization function can be seen as the "weighted" cross-entropy, meaning that every ground-truth token is represented with similarities to other words in the vocabulary rather than with a traditional one-hot- encoding scheme.

Soft label loss

$$\mathcal{L} = -\sum_{i=1}^{V} y_i^* log p_i$$

$$y_i^* = \begin{cases} \frac{\text{sim}(y_t, y_i)}{\sum_{j=1}^{N} \text{sim}(y_t, y_j)}, & y_i \in \text{top N} \\ 0, & y_i \notin \text{top N} \end{cases}$$

- We weigh the generated tokens by their similarity to the ground truth tokens across the vocabulary
- We consider only the top ***N*** closest words in the vocabulary and normalize so that the similarities add up to one.[1]
- Essentially, the loss function can be interpreted as cross-entropy with soft targets.
- We vary ***N*** from 3 to 10 in our experiments. We exclude common English stop-words from soft target encoding, i.e. we apply a regular cross entropy loss for reconstructing of these words.

[1] Our experiments showed that removing this normalization does not negatively affect system performance

Experiments

We trained several autoencoder models using the regular cross-entropy and the proposed alternative losses. For evaluation, we used the learned representations for the following tasks:

- Paraphrase detection task → Microsoft Research Paraphrase Corpus (MRPC)
- Language inference tasks
  - → Stanford Natural Language Inference (SNLI)
  - → Sentences Involving Compositional Knowledge, Entailment (SICK-E)

We selected these tasks because they seem to be likely to benefit from capturing word-level semantic similarity.

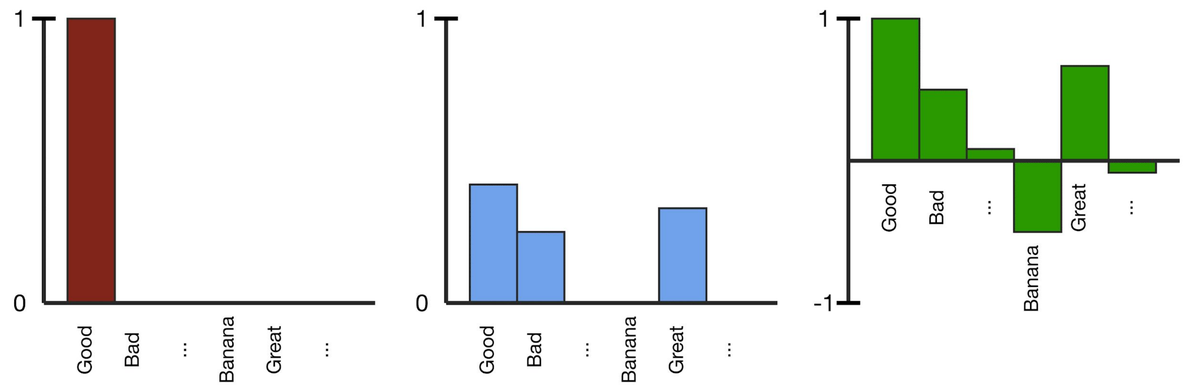


Figure 1. Schematic illustration of true-label encoding using the standard cross-entropy loss (left), soft label loss for N = 3 (center) and weighted similarity/weighted cross-entropy loss (right). All the three examples "good", "great" and "bad" are close in the embedding space, since they appear in similar contexts. Note that all the soft labels add up to 1, while weighted similarity labels for the third loss can vary in the range from -1 to 1.

Implementation

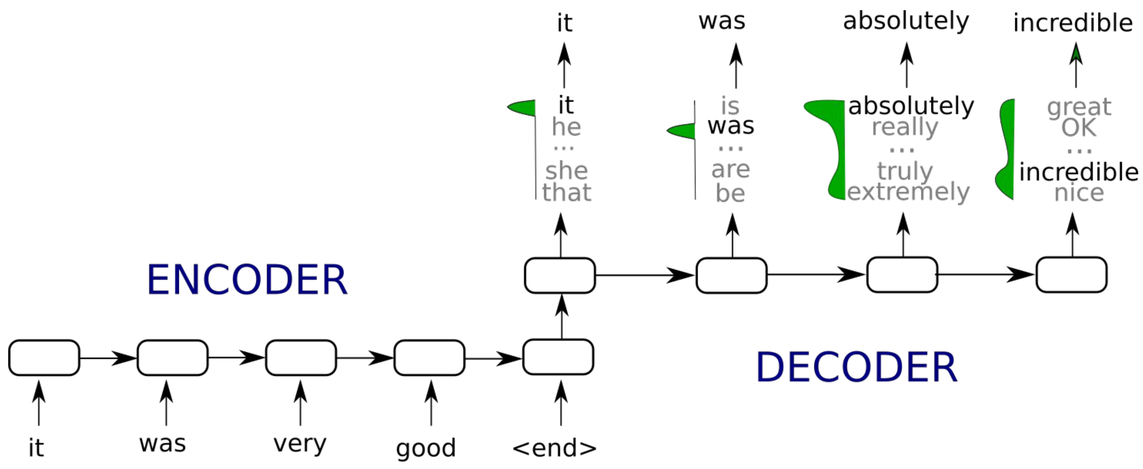


Figure 2. Model architecture. Word similarities are used as weights in the per-word reconstruction loss.

Implementation details:

- The model is implemented using the PyTorch deep learning framework
- LSTMs with the hidden size of 256 units are used for both encoder and decoder
- The resulting vocabulary size of the dataset is 9.5K tokens
- Adam optimizer with the learning rate that varies depending on the tested loss between 0.001 and 0.0001 is used for training
- Learned representations are evaluated using SentEval toolkit
- We use pre-trained fastText word vectors to compute similarities between words.

Results

Configuration	Autoencoder outputs	
	Input sentence	Reconstructed sentence
Cross-entropy	you can trust this business	you can trust this business
Soft-label, N=3	the taste was so good	the flavor was so good
Soft-label, N=5	her tone was incredibly rude	her attitude was incredibly unprofessional
Soft-label, N=10	a very nice spot for a quiet lunch	a very nice slot for a tranquil lunchtime
Weighted similarity	once again the staff were wonderful	that so that service and great
Weighted cross-entropy	great breakfast option	great food place

Table 1. Sample autoencoder reconstruction outputs for the tested loss configurations.

- The standard cross entropy reconstructs the sentence **exactly** word-by-word
- The rest of the models reconstruct the **synonyms** at the word level
  - Results in a **slightly different** expression style
  - Overall meaning is conveyed **correctly**

Model	MRPC		SNLI	SICK-E
	F1	Acc	Acc	Acc
Cross-entropy	79.0	66.9	44.8	56.8
Soft label, N=3	77.6	67.1	57.8	71.8
Soft label, N=5	79.1	67.3	57.2	71.6
Soft label, N=10	77.9	66.5	57.9	**72.4**
Weighted similarity	77.5	65.6	**69.1**	56.6
Weighted cross-entropy	**79.4**	**68.2**	57.2	70.2

Table 2. Results of transfer learning tasks performance of the proposed autoencoder models.

All the proposed loss functions **outperform** conventional cross-entropy!

These losses produce diversified output and **can be easily integrated** in any existing encoder-decoder architecture!!