Sequence-to-sequence in Neural Machine Translation

English-to-Italian Translator

Eleonora Vitanza & Nicholas Redi

PROJECT GOAL AND DATASET MANAGEMENT

The purpose was to build a neural model able to translate sentences from English to Italian.

We used **PyTorch** to implement the model, we trained it using the dataset from http://www.manythings.org/anki/ and we evaluated it using the **BLEU score** metric.

The dataset has 343813 pairs of sentences; we mostly used ⅓ of it to evaluate the performances, and then we used the total set on the best founded configurations to achieve further improvements.

We splitted data into **train set**, **validation set** and **test set** with respectively 70%, 15% and 15% of the considered dataset.

We also exploited data to build the language **vocabularies**, tokenizing each sentence and keeping only those words with more than frequence 3. Every out-of-vocabulary word was replaced with a special <unk> token; <sos>, <eos> and <pad> were added too.

GENERAL MODEL

To face the translation task, it is necessary to map the input sequence to a vector of a fixed dimensionality.
In order to do that, we use a LSTM to encode the data and another LSTM to decode the target sequence from the vector.

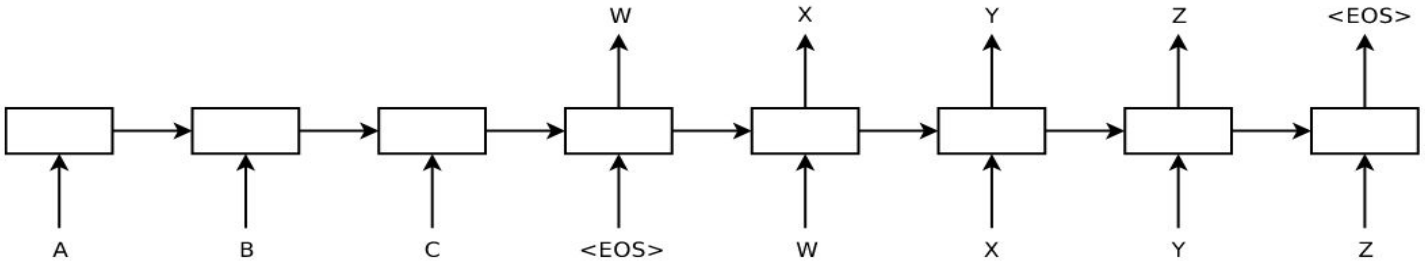


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

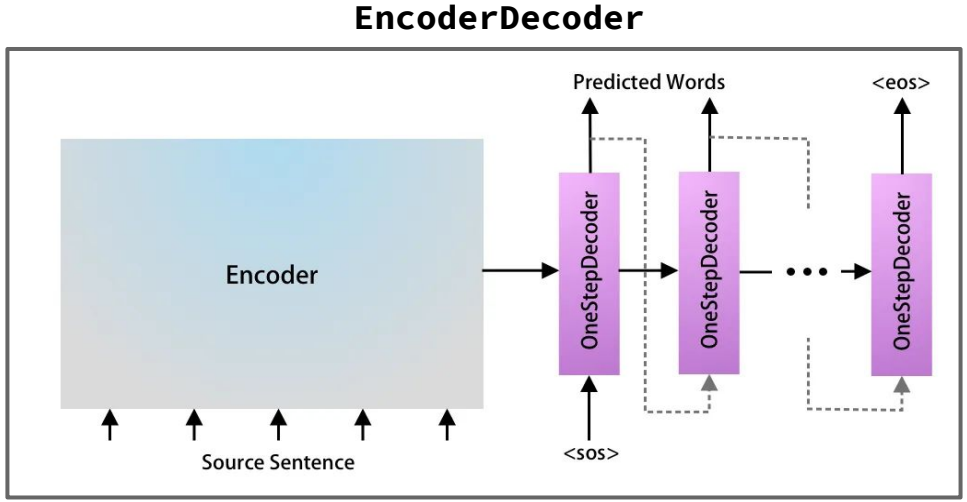
Figure from the paper https://papers.nips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

MORE ABOUT THE MODEL...

Encoder and Decoder are wrapped in a global structure called **EncoderDecoder**.

The **Encoder** takes the reversed input sentence with the embedding representation and provide a brief summary of it through an LSTM.

The **Decoder** call recursively the **OneStepDecoder** unit to predict one word at time, until <eos> is reached.



DIFFERENT APPROACHES

We focused on different alternatives:

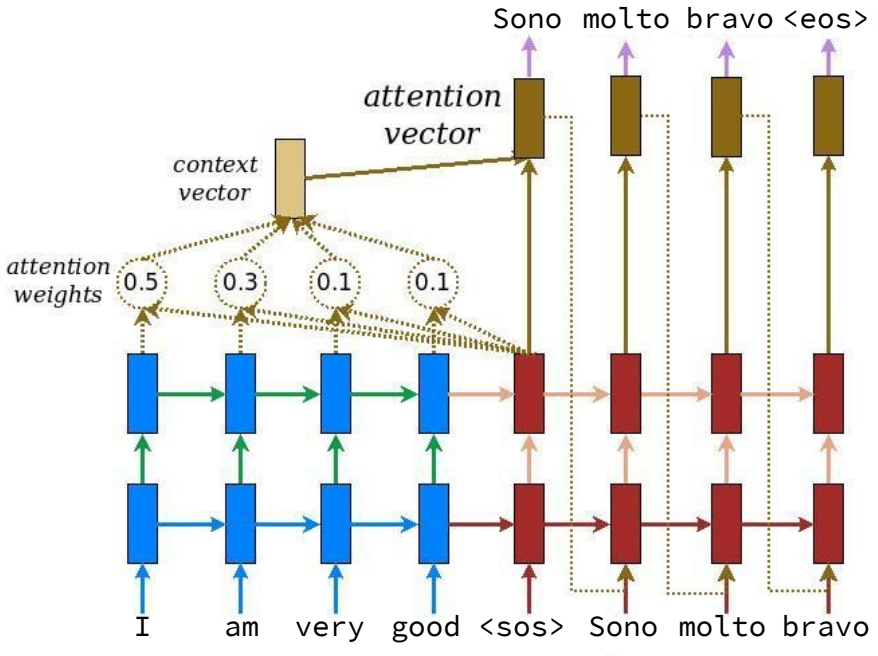
- **basic case**: single-layered and monodirectional LSTM for both Encoder and Decoder
- **multi-layered** and monodirectional LSTM for both Encoder and Decoder
- **bidirectional** and single-layered LSTM for the Encoder
- **attention mechanism** with monodirectional and single-layered LSTM for both Encoder and Decoder (note that in this case we do not revert the input sentence)

ATTENTION MECHANISM

We implemented a **global attention-based** model which derive a **context vector** that captures relevant input information to help predict the current target word.

Specifically, the global context vector is computed as the weighted average, according to the **attention weights**, over all the input tokens.

The attention weights are computed through: $a_t = \text{softmax}(W_1(\tanh(W_2[h_s + h_t])))$ where $\mathbf{W_1}$ and $\mathbf{W_2}$ are **learnable parameters** and $\mathbf{h_s}$ and $\mathbf{h_t}$ are respectively the source and the target **hidden states**.



TRAINING DETAILS

We used the **torch.nn.CrossEntropyLoss** as loss function to train the models. It is commonly used for classification problems, and we used it because the NMT can be considered as a sequence of classification tasks (each predicted token is the most probable word among the entire target vocabulary).

To make the learning more robust, we applied **dropout** to our model in the case of multilayered LSTMs; we found on early tests (and we maintained it on subsequent tests) that the optimal dropout probability is 0.1, probably because an higher value is too much for just 2 or 3 layers of LSTMs.

Moreover, we help the network with the learning process, using randomly the target instead of the predicted word in the OneStepDecoder (TFR).
From early tests, we discovered that it is better to keep the **teacher forcing ratio** (TFR) fixed at 0.5.

EXPERIMENTS

We summarize our initial experimentation on the **basic case** in the following table:

PRETRAINED_EMBED	BIDIRECTIONAL	ATTENTION	LR	EMBEDDING_DIM	HIDDEN_DIM	DROPOUT	N_LAYERS	TFR	EPOCHS	BATCH_SIZE	FRAC	TRAIN_LOSS	VALIDATION_LOSS	BLEU_SCORE
False	False	False	0.01	64	256	0.0	1	0.5	10	512	3	1.71	2.92	23.32
False	False	False	0.1	64	256	0.0	1	0.5	10	512	3	4.01	4.69	1.62
False	False	False	0.001	64	256	0.0	1	0.5	10	512	3	2.35	3.05	13.23
False	False	False	0.001	128	256	0.0	1	0.5	10	512	3	1.97	2.85	17.9
False	False	False	0.01	128	256	0.0	1	0.5	10	512	3	1.34	2.96	20.58
False	False	False	0.001	128	512	0.0	1	0.5	10	512	3	1.43	2.51	25.52
False	False	False	0.01	128	512	0.0	1	0.5	10	512	3	1.56	2.79	27.12
False	False	False	0.01	128	1024	0.0	1	0.5	10	512	3	2.12	3.64	10.73
False	False	False	0.001	128	1024	0.0	1	0.5	10	512	3	1.13	2.42	30.92
False	False	False	0.01	256	256	0.0	1	0.5	10	512	3	1.49	2.96	23.13
False	False	False	0.001	256	256	0.0	1	0.5	10	512	3	1.63	2.7	22.64
False	False	False	0.01	512	512	0.0	1	0.5	10	512	3	1.63	2.89	26.16
False	False	False	0.001	512	512	0.0	1	0.5	10	512	3	1.17	2.35	32.86
False	False	False	0.01	256	1024	0.0	1	0.5	10	512	3	1.72	3.1	21.09
False	False	False	0.001	256	1024	0.0	1	0.5	10	512	3	1.11	2.4	34.61
False	False	False	0.001	512	1024	0.0	1	0.5	10	512	3	1.08	2.30	36.61
False	False	False	0.001	1024	1024	0.0	1	0.5	10	512	3	1.14	2.33	37.5
True	False	False	0.001	50	1024	0.0	1	0.5	10	512	3	1.55	2.71	19.19

ANALYSING THE RESULTS

We tried different configurations of hyperparameters in the case of one layered LSTM with zero dropout considering ⅓ of the entire dataset, and we looked for the one with minimum validation loss.

We saw that the most appropriate **learning rate** for the task is 0.001 and that 10 **epochs** are enough to find the best model.

We also found that the best mix between the **embedding dim** and **hidden dim** is the pair (512,1024). We kept the **batch size** at the maximum possible value compatibly with the GPU memory.

In this case the best model achieves **validation loss of 2.30** and **BLEU score 36.61**.

Once we fixed the best configuration, we tried to achieve further improvements by using separately the multilayered, the bidirectional and the attention approaches.

FURTHER EXPERIMENTS

PRETRAINED_EMBED	BIDIRECTIONAL	ATTENTION	LR	EMBEDDING_DIM	HIDDEN_DIM	DROPOUT	N_LAYERS	TFR	EPOCHS	BATCH_SIZE	FRAC	TRAIN_LOSS	VALIDATION_LOSS	BLEU_SCORE
False	False	False	0.001	512	1024	0.1	2	0.5	10	512	3	1.13	2.38	36.06
False	False	False	0.001	512	1024	0.1	3	0.5	10	512	3	1.25	2.53	28.45
False	True	False	0.001	512	1024	0.0	1	0.5	10	512	3	1.0	2.28	36.98
True	True	False	0.001	50	1024	0.0	1	0.5	10	512	3	1.39	2.72	23.52
False	False	True	0.001	512	1024	0.0	1	0.5	10	32	3	1.47	2.54	38.13
True	False	True	0.001	512	1024	0.0	1	0.5	10	64	3	1.91	2.46	31.85
False	False	True	0.001	256	1024	0.0	1	0.5	10	32	3	1.39	2.41	38.23
False	False	False	0.001	512	1024	0.0	1	0.5	10	256	1	1.22	1.98	48.2
False	False	False	0.001	512	1024	0.1	2	0.5	10	256	1	1.12	1.94	49.79
False	True	False	0.001	512	1024	0.0	1	0.5	10	256	1	1.09	1.98	48.82
False	False	True	0.001	256	1024	0.0	1	0.5	10	64	1	1.09	2.03	49.95

In order to have a good result, we found that **pretrained embeddings** are not convenient, whereas **2 layered** or **bidirectional** LSTMs help the performance, as **attention** too. Finally, we repeated the best four tests on the entire dataset (FRAC 1), getting similar results between each other and a general improvement on the BLEU score.

CONCLUSIONS

The **attention-based model** has **validation loss 2.03** and **BLEU score 49.95**.

It works pretty well in some sentences, for instance:

```
      input sentence: There is nothing like cold beer on a hot day.
     target sentence: Non c'è nulla come una birra fredda in un giorno caldo .
  predicted sentence: Non c'è nulla come una birra fredda in un giorno di caldo .
```

But it is not perfect in others, e.g.:

```
      input sentence: She didn't tolerate his selfishness.
     target sentence: Lei non tollerava il suo egoismo .
  predicted sentence: Non non <unk> il suo egoismo .
```

We think that further work on mixed bidirectional and multi-layered LSTMs, with attention mechanisms, can improve the quality of the learning and the generalization capabilities even more.