Deep comedy

Baseline & first thoughts

Alessandro Pacielli

Michela Lapenna

Tasks

1. Syllabification
2. Text generation

Training Set

Target introduces a “syllable separator” character:

Input:

Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura,
ché la diritta via era smarrita.

Target:

|Nel |mez|zo |del |cam|min |di |no|stra |vi|ta
|mi |ri|tro|vai |per |u|na |sel|va o|scu|ra,
|ché |la |di|rit|ta |via |e|ra |smar|ri|ta.

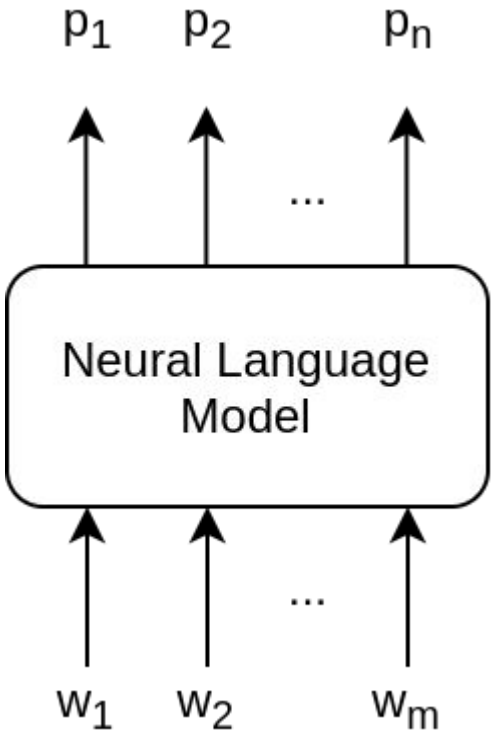
How to perform syllabification with Neural Networks?

Neural language models employ NNs capable of processing sequences (RNNs, transformers)

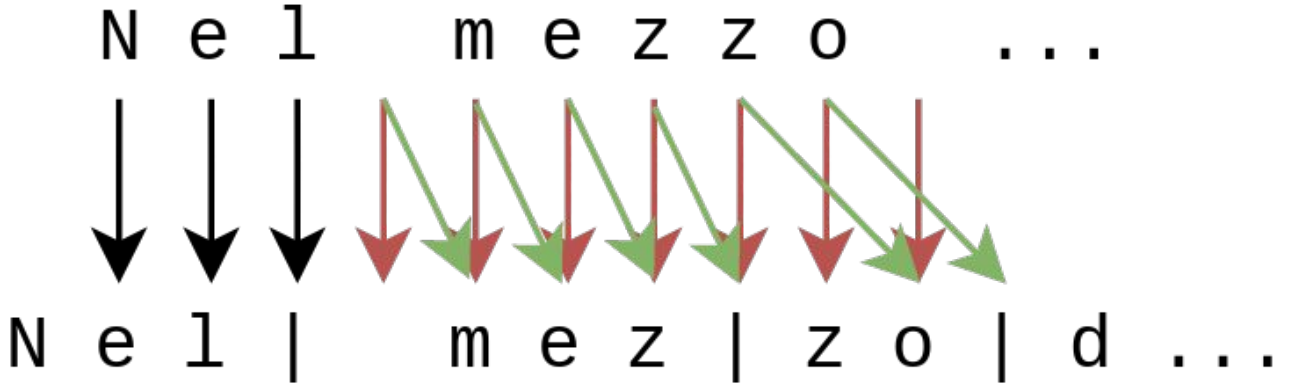
A sequence of tokens is fed to the LM, its output is a distribution over the output vocabulary that predicts the probability of the next word.

Can we model syllabification with a neural LM?

We could if len(input) == len(output), but that's not the case with syllabification



Char-level Neural Model



HARD TO DO WITH A CLASSICAL LANGUAGE MODEL

Input and output are **misaligned** because their sequence length differ

We could output 2 chars at once (artificial chars like “|a”, “|b”, …, “|z”) but the training set is small

Word/Subword-level Neural Model

Didn’t work well because:

1. Word-level
   a. Loses generality (only words in Dante’s work)
   b. Intuition: **syllabification is done at char-level + word before/after current word**
2. Sub-word
   a. Split word in syllables in order to split them into syllables (?) (Chicken-egg problem)
   b. Same intuition as word-level

Is there another way?

What if...

… we model the syllabification task as a **Neural Machine Translation** task?

Input “language”: standard Dante

Output “language”: syllabified Dante

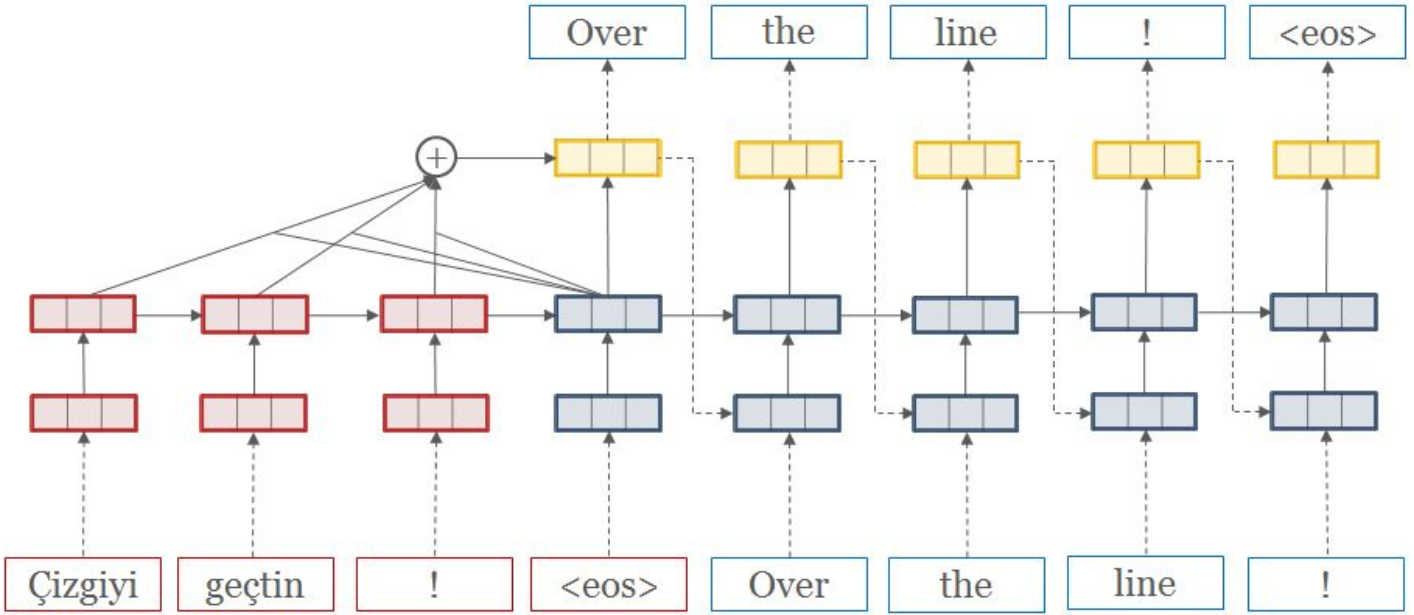
Neural Machine Translation Architectures

State of the art NMT architectures are **Encoder/Decoder** architectures:

- An Encoder takes a sequence of words and produces a **latent representation** of the input sentence
- The Decoder
  - Takes the latent representation
  - **Generates** text given its own output until end symbol is produced

The fact that the Decoder *generates* text might be useful in the generation step!

Neural Machine Translation Architectures



Teacher forcing (feed correct sentence to the decoder)

Image from opennmt.net

Syllabification Architectures

1. RNN encoder, RNN decoder ← **BAD performances**
    a. Only last hidden state of encoder is kept (enc_hidden)
    b. The decoder starts with initial_state=enc_hidden
2. Bidirectional RNN ← TODO
3. Transformer architecture ← **Very good performances**
    a. Positional encoding is added to embedding
    b. Both the encoder and decoder exhibit a mechanism of self-attention
    c. The self-attention of the decoder is followed by an encoder-decoder attention
    d. The transformer is constituted by a stack of encoders and decoders
    e. The attention is multi-headed and each head works in parallel

Transformer model

After 20 Epochs of training on single verses: ~0.02 loss (Categorical Crossentropy)

Good empirical results, high accuracy

```
sentence = "^E come l'aere, quand' è ben pïorno,$"
ground_truth = "|E |co|me |l' ae|re, |quan|d' è |ben |pï|or|no,"


translated_text, attention_weights = evaluate(sentence)
print_translation(sentence, translated_text, ground_truth)
```

```
Input:         : ^E come l'aere, quand' è ben pïorno,$
Prediction     : | E   | c o | m e   | l '   a | e | r e ,   | q u a n | d '   è   | b e n   | p ï | o r | n o , $
Ground truth   : |E |co|me |l' ae|re, |quan|d' è |ben |pï|or|no,
```

What’s next?

Poetry generation

1. Use transformer as an auto-regressive model
2. Can we combine this syllabification layer with a word-level auto-regressive generator?
3. GANs?

Training & evaluation

Training data:

1. Data augmentation: Instead of just verses, we could train the model on words and sub-sentences
2. Increase training set: use other syllabification techniques to syllabify other texts both in poetry and prose (e.g. *Orlando Furioso*)

Evaluation:

1. Word-level accuracy (i.e. how many words are correctly syllabified?)
2. Evaluate generation
    a. Tercets?
    b. Rhymes?
    c. Hendecasyllables?
    d. Lexical correctness → real words?

References

1. Speech and Language Processing, Jurafsky, Daniel and Martin, James H., https://web.stanford.edu/~jurafsky/slp3/
2. Attention is all you need, Vaswani, Ashish et al., https://arxiv.org/abs/1706.03762
3. Tensorflow tutorials
    a. https://www.tensorflow.org/tutorials/text/text_generation
    b. https://www.tensorflow.org/tutorials/text/nmt_with_attention
    c. https://www.tensorflow.org/tutorials/text/transformer