COMP3220 — Document Processing and the Semantic Web

Week 06 L1: Advanced Topics in Deep Learning

Diego Mollá

Department of Computer Science
Macquarie University

COMP3220 2021H1

Programme

1. Text Generation
2. Encoder-Decoder Architecture
3. Pre-training and Fine-tuning

Reading

- Deep Learning book, section 8.1.

Additional Reading

- Jurafsky & Martin, Chapter 9.
- The Illustrated BERT, ELMo, and co.: http://jalammar.github.io/illustrated-bert/

Programme

1. Text Generation
2. Encoder-Decoder Architecture
3. Pre-training and Fine-tuning

Generating Text Sequences

- One of the advances of deep learning versus shallower approaches to machine learning is its ability to process complex contexts.
- This has allowed significant advances in image and text processing.
- We have seen how to process text sequences for text classification.

Text generation as a particular case of text classification

- Given a piece of text ...
- Predict the next character.

Text Generation as Character Prediction

- Our training data is a set of samples of the form:
  - **Input** Text fragment.
  - **Label** Next character to predict.
- We do not need to manually annotate the training data: the data are self-annotated.
- This means that we can easily gather training data for text generation.
- This is the idea for training language models (next slide).

Language Models

- Given a collection of text, we can train a language model that can be used to generate text in the same style.

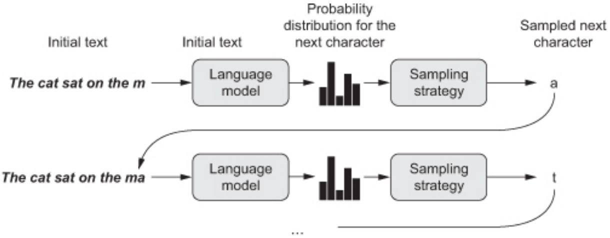


Figure 8.1 of Chollet (2018).

Implementing Character-level LSTM Text Generation

- The architecture of the model is of the kinds we have seen for text classification.
  - The input is a sequence of characters.
  - The "class" to predict is the next character to generate.
- If we add an embedding layer after the input, This layer will learn character embeddings.

```
model = tf.keras.models.Sequential()
model.add(layers.Embedding(len(chars), 20, input_len=maxlen))
model.add(layers.LSTM(128))
model.add(layers.Dense(len(chars), activation='softmax))
```

Generating Text

- Remember that the output of a prediction is a probability distribution.
- To generate the next character, we can sample from the probability distribution.
- We can determine how deterministic the sampling is:
  - We can always return the character with highest probability ...
  - Or we can select a character randomly ...
  - Or we can do something in between, according to a "temperature" parameter.

```
import numpy as np
def reweight_distribution(original_distribution, temperature=0.5):
    distribution = np.log(original_distribution) / temperature
    distribution = np.exp(distribution)
    return distribution / np.sum(distribution)
```

Figure: Different Reweightings

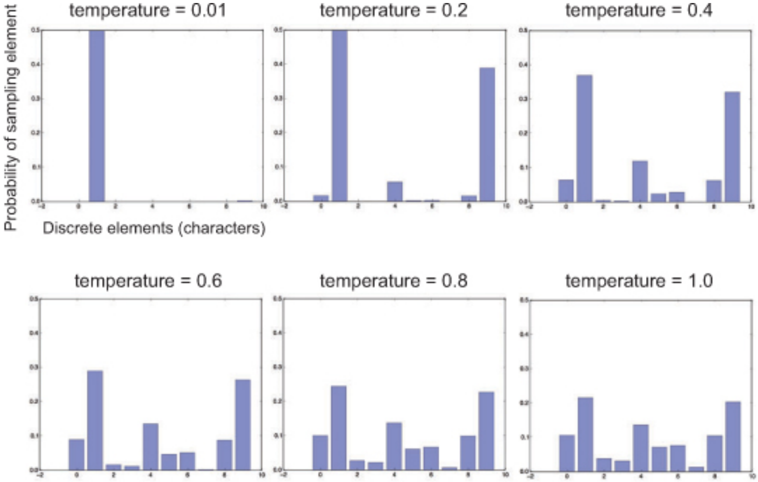


Figure 8.2 of Chollet (2018)

Example

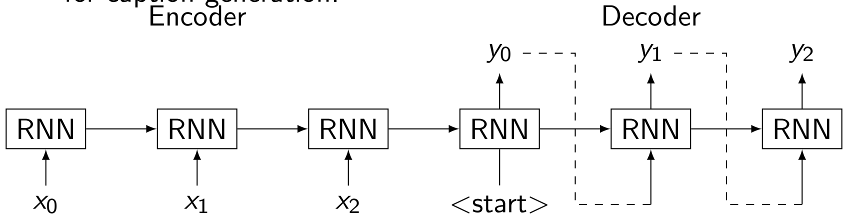
See notebook . . .

Programme

1. Text Generation
2. Encoder-Decoder Architecture
3. Pre-training and Fine-tuning

The Encoder-Decoder Architecture

- Composed of an encoder and a decoder.
  - The encoder can be an RNN chain that takes the input.
  - The decoder can be an RNN that takes the output of the previous RNN as input.
- Revolutionised machine translation and many other text processing applications.
- The encoder stage can be something non-textual, e.g. images for caption generation.

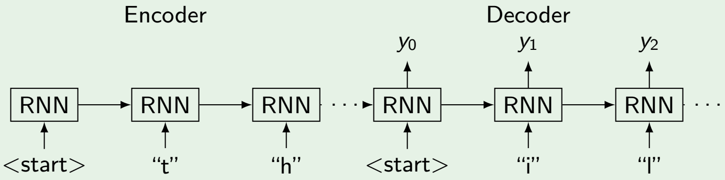


Training the Encoder-Decoder Architecture

A common approach to train the encoder-decoder architecture is to apply teacher forcing:

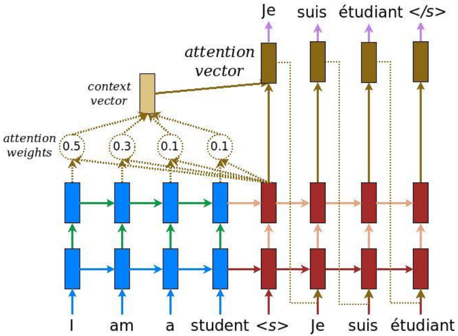
- Use the target sequence to guide the training of the decoder.
- For example, in an English to French machine translation system, we feed the target French translation to the decoder.

“The weather is fine” $\longrightarrow$ “Il fait bon”



Attention: An Improvement to the Encoder-Decoder Architecture

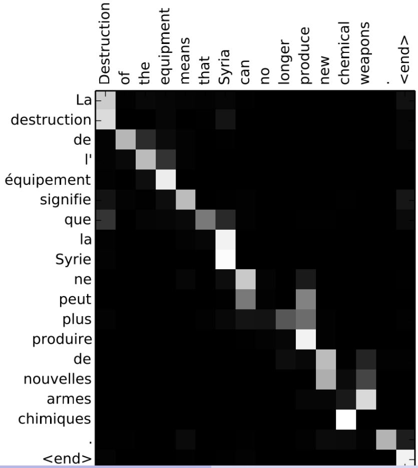
Attention is an enhancement in the seq2seq architecture that allows to focus on parts of the input during the generation stage by the decoder.



https://github.com/tensorflow/tensorflow/blob/r1.13/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb

Attention for MT

Very useful to start understanding the decision processes of the model.



Attention in Caption Generation

A woman is throwing a frisbee in a park.

Xu et al. (2015) arXiv:1502.03044

Programme

1. Text Generation
2. Encoder-Decoder Architecture
3. Pre-training and Fine-tuning

Problems with Supervised Learning

Annotated data

- Supervised learning requires (a lot of) annotated data.
- Annotated data can be costly.
- Human annotated data can contain annotation errors.

Training size

- Supervised learning requires a lot of (annotated) data.
- Large companies can afford the resources for processing large volumes of data, others can't.
- Some domains do not have much text anyway.

Problems with Supervised Learning

Annotated data

- Supervised learning requires (a lot of) annotated data.
- Annotated data can be costly.
- Human annotated data can contain annotation errors.

Training size

- Supervised learning requires a lot of (annotated) data.
- Large companies can afford the resources for processing large volumes of data, others can't.
- Some domains do not have much text anyway.

Even if we could afford training large models using large volumes of data . . .

Artificial intelligence / Machine learning

Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

by **Karen Hao** June 6, 2019

https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/

Solution: Pre-Train and Fine-Tune

Pre-training

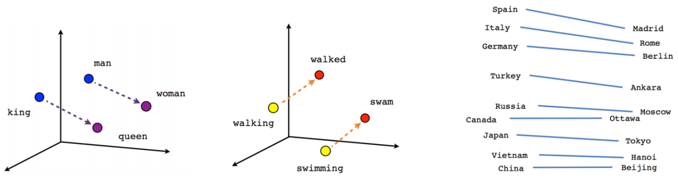
- Develop a system that can be trained with large volumes of data.
- Make the system as general as possible, so that it can be used for multiple tasks.

Fine-tuning

- Design a Deep Learning model that contains:
  - A layer pre-trained for a general task.
  - Additional layers that adapt the general task to our specific task.
- Fine-tune the system using the (smaller) training data of our specific task.

Example: Word Embeddings

- As we have seen in a previous lecture, word embeddings can be learnt using large, unlabelled data.
- These pre-trained word embeddings can be used to initialise an embeddings layer in our Deep Learning model.
- When we train our system, we have the choice to update these word embeddings, or not.



Male-Female Verb tense Country-Capital

Huggingface's transformers library

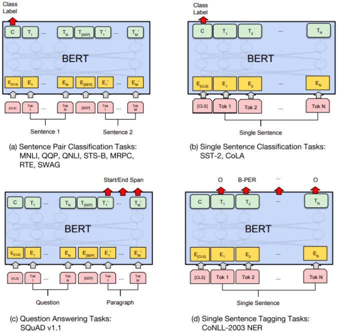
Transformers

https://github.com/huggingface/transformers

- Huggingface's transformers library contains a large repository of pre-trained models.
- These models are contributions from many researchers and developers.
- These models are being used to obtain state-of-the-art results.

Example: Using BERT in Keras

- BERT is one of the most popular architectures for pre-training and fine-tuning.
- Look at the lecture notebook for an example of use in keras.
- BERT is easy to use, but fine-tuning can take a long time.



http://jalammar.github.io/illustrated-bert/

Take-home Messages

1. Text generation as a task of character (or word) prediction.
2. Describe the encoder-decoder architecture. What is this architecture good for?
3. What is teacher forced training and what is it good for?
4. Transfer learning and fine-tuning.

What's Next

Weeks 7-12

- Semantic Web (Rolf Schwitter).
- Assignment 2 submission deadline on Friday 23 April 2021.