COMP348 — Document Processing and the Semantic Web

Week 06 L1: Advanced Topics in Deep Learning

Diego Mollá

COMP348 2019H1

Abstract

This is the final lecture on deep learning where we will introduce several advanced use of deep learning for text processing. The emphasis here is on aspects related to the generation of text. We will see an approach that generates text by learning a language model based on a corpus, and we will advance some topics on the use of encoding and decoding architectures that are able to generate text based on some input context. This can be used in multiple tasks, such as machine translation (e.g. French to English, text summarisation (from text to a summary), or even caption generation (from an image to text). We will conclude with open challenges in deep learning that are the subject of current research.

Update March 27, 2019

Contents

Reading

- Deep Learning book, section 8.1.

1 Text Generation

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 enormous 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:
  - Text fragment.
  - Next character to predict.
- This kind of data can be gathered from any corpus.
- Given a corpus, we can train a language model that can be used to generate text in the same style.

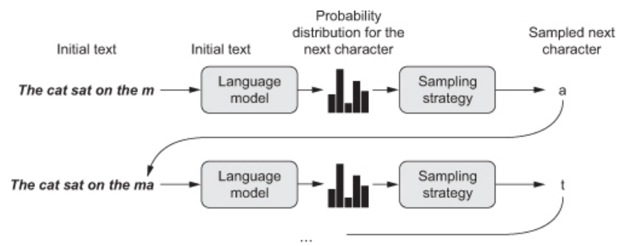


Figure 8.1 of Chollet (2018).

Implementing Character-level LSTM Text Generation

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

```
model = 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

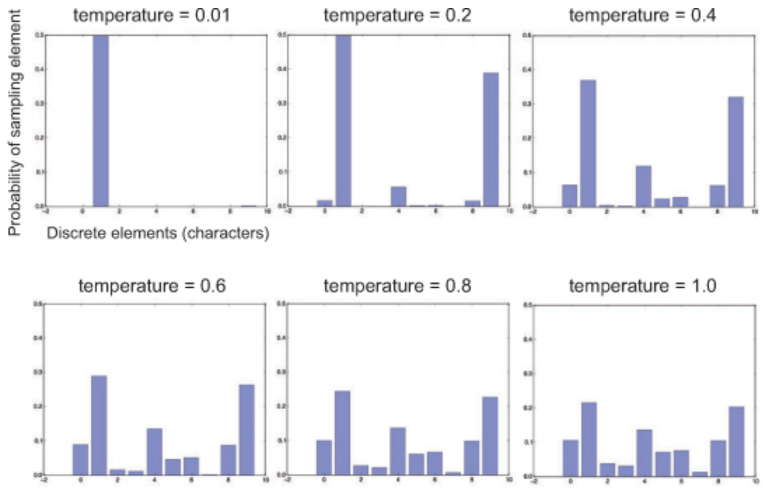


Figure 8.2 of Chollet (2018)

The figure shows the reweightings of a sample distribution as we change the temperature. Low temperature will generate a deterministic distribution where only one value has probability near 1, and the other values have probabilities near 0. In contrast, high temperature will generate probabilities that are nearly identical, simulating random choice.

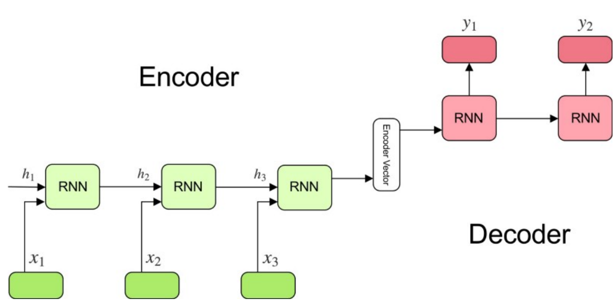
Example

See notebook . . .

2 Encoder-Decoder Architecture

The Encoder-Decoder Architecture

- Composed of an encoder and a decoder.
- Revolutionised machine translation and many other text processing applications.
- The encoder stage can be something non-textual, e.g. images for caption generation.



https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346

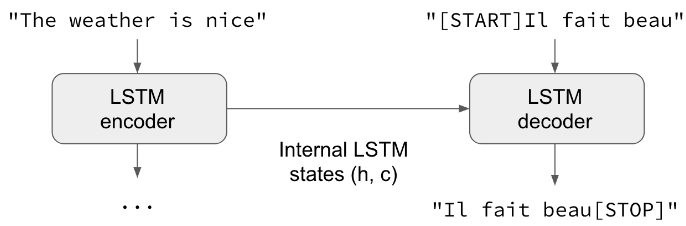
The encoder-decoder architecture is a general architecture that can be used for any case where the desired output is a sequence of length different from the input sequence. It can even be used to generate text based on non-textual information, such as image caption generation.

In the most basic approach, the encoder-decoder architecture can be implemented as two RNN layers: the encoder is an RNN layer that generates an output. This outut is then the input to the decoder. Many variants and enhancements of this architecture are being proposed.

Training the Encoder-Decoder Architecture

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

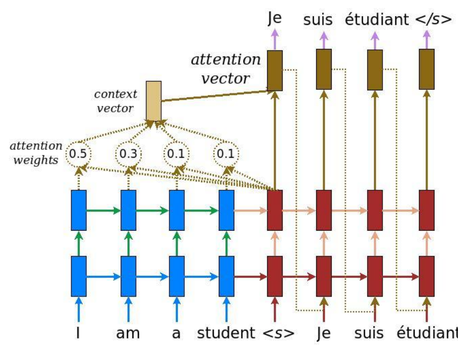
- Use the target sequence to guide the training of the decoder.
- For example, in an English to French machine translation system, we feed the target French translation to the decoder.



https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

Attention: An Improvement on the Encoder-Decoder Architecture

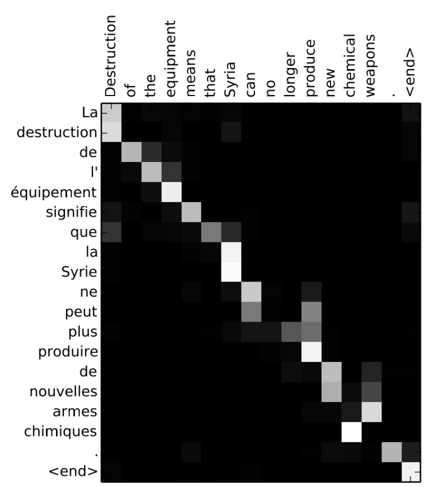
Attention is an enhancement in the seq2seq architecture that allows to focus on parts of the input for generation.



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.



Bahdanau et al. (2015) arXiv:1409.0473

Attention in Caption Generation

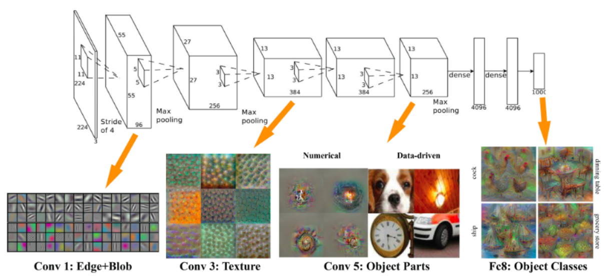
A woman is throwing a frisbee in a park.

Xu et al. (2015) arXiv:1502.03044

3 Open Challenges in Deep Learning

Interpretability

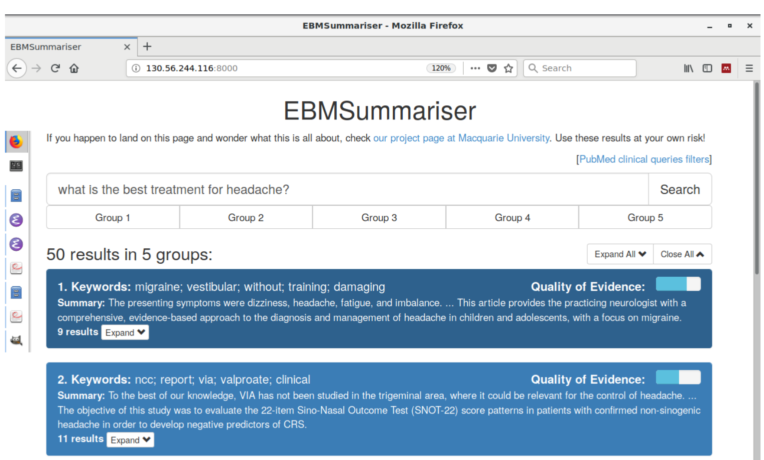
- It is very difficult to interpret most weights in a neural model.
- Approaches like attention help to visualise some of the processes but much more is needed.
- Current research in image processing can visualise interpretations of middle layers. How to do the same with text?



http://vision03.csail.mit.edu/cnn_art/index.html

Justifiability

How can someone justify a decision made by a neural model?



Small Training Data

- Deep learning excels when there are large volumes of training data.
- But obtaining labelled training data is expensive ...

  ⇒ We can add unsupervised and semi-supervised tasks.
- ... and some domains and languages have very little data ...

  ⇒ Transfer learning: Pre-train on one domain and adapt the learnt model to another domain.

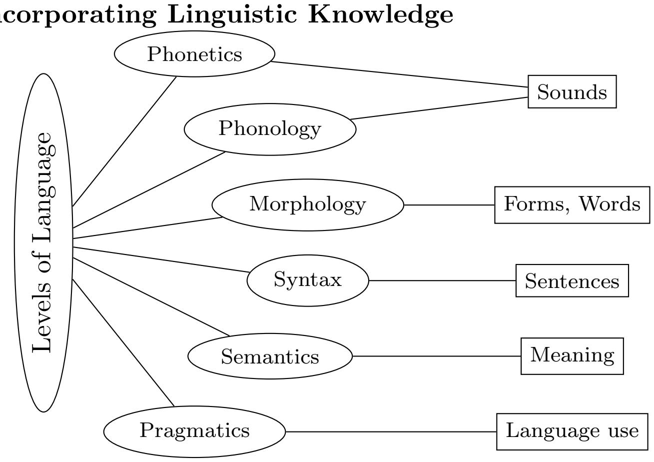
Incorporating Knowledge

- Early natural language systems could easily incorporate knowledge.
  - Ontologies, databases, information given by the user, etc.
- Deep learning approaches find this more difficult.
- Question answering and dialogue systems often do not remember what has been said before.

message Where do you live now?
response I live in Los Angeles.
message In which city do you live now?
response I live in Madrid.
message In which country do you live now?
response England, you?

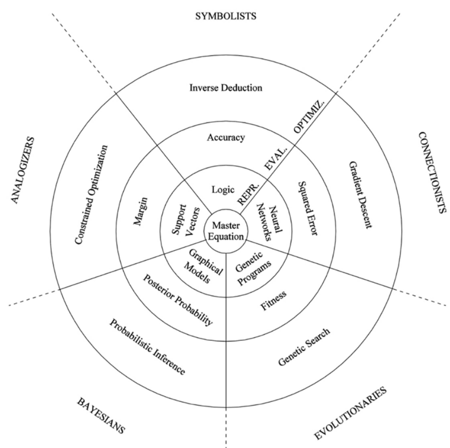
Vinyals & Le (2015) https://arxiv.org/abs/1506.05869

Incorporating Linguistic Knowledge

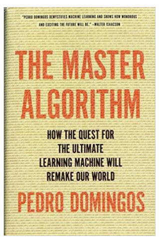


There have been many studies of language in the past. Linguists have developed very comprehensive theories of how language works. However, current deep learning approaches ignore all of this information. Is there a way to integrate this information to create better informed systems?

Deep Learning is Not Everything



Deep Learning is with the "Connectionists" tribe.



Deep Learning is only one kind of machine learning. There are others. For example, Pedro Domingos identifies 5 types of machine learning "tribes". Deep Learning would be with the "Connectionists" tribe.

Take-home Messages

1. Text generation as a task of character (or word) prediction.
2. We may want to control the level of randomness when generating text based on a "temperature" parameter.
3. Describe the encoder-decoder architecture. What is this architecture good for?
4. What is teacher forced training and what is it good for?
5. Comment on current open challenges in deep learning.

What's Next

Weeks 7-12

- Semantic Web (Rolf Schwitter).
- Assignment 2 submission deadline on Friday 26 April 2019.