COMP3220 — Document Processing and Semantic Technologies

Week 06 L1: Processing Text Sequences

Diego Mollá

COMP3220 2022H1

Abstract

This is the final lecture on deep learning where we will introduce several advanced topics on deep learning for text processing. The main focus here is on the processing of text sequences. We will explore the use of recurrent neural networks that have the ability to memorize lon-term dependencies. We will then introduce a hot topic in current research, namely the idea of pre-training and fine tune. We will illustrate this by using a popular machine learning library that uses some of the state of the art pre-trained models.

Update March 26, 2022

Contents

Reading

- Deep Learning book (2nd edition), Chapter 11.
- Jurafsky & Martin, Chapter 9.
- The Illustrated BERT, ELMo, and co.: `http://jalammar.github.io/illustrated-bert/`

1 Text Sequences

Handling Text Sequences

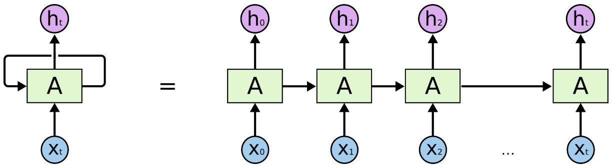
- A document is a sequence of words.
- Many document representations are based on a bag-of-words approach.
  - Word order is ignored.
  - The context around a word is ignored.
- Even word embeddings ignore word order.

Why context matters
“I can$_1$ kick the can$_2$”

- The meaning of "can_1" is different from that of "can_2".
- "can_1" and "can_2" should have different word embeddings.
- We can tell the meaning because of the context:
    - "I can kick . . . "
    - ". . . kick the can"

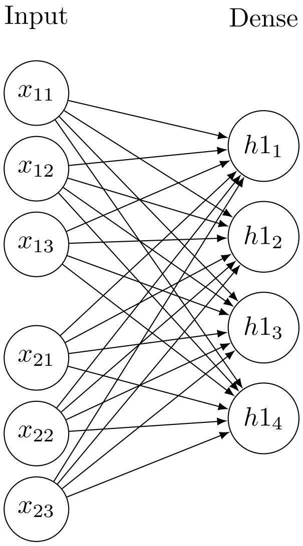
Recurrent Neural Networks

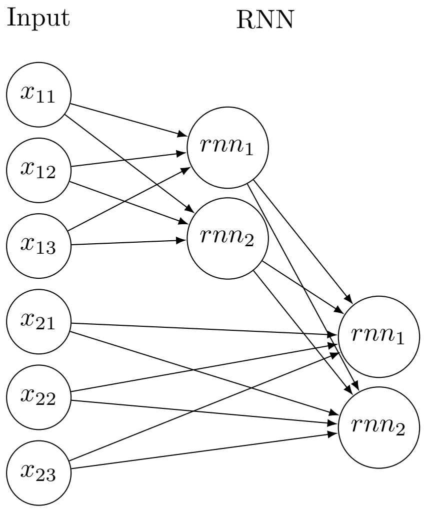
- A Recurrent Neural Network (RNN) contains layers that are designed to process sequences.
- A RNN layer is composed of RNN cells.
- Each RNN cell takes as input two pieces of information:
    1. A vector representing an item x_i in the sequence.
    2. The state resulting from processing the previous items.
- The output of the RNN cell is a state that can be fed to the next cell in the sequence.
- All cells in an RNN layer share the same parameters.
- In a sense, we can say that an RNN cell is the same for all words in the sequence, but now context also matters.



`https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Example: Dense layer vs RNN





This example illustrates the difference between a regular dense layer and an RNN layer for a sequence of two words, where each word is represented as a vector of 3 elements.

In a dense layer, the two words are processed simultaneously in the dense layer.

In a RNN layer, the first word is processed first (cells rnn_1 and $rnn2$), and the output of this is processed later using a copy of rnn_1 and rnn_2 that uses exactly the same input weights.

- The dense layer has 4 cells in this example.
- The RNN layer is a simple dense layer with 2 cells in this example (there are other possibilities).

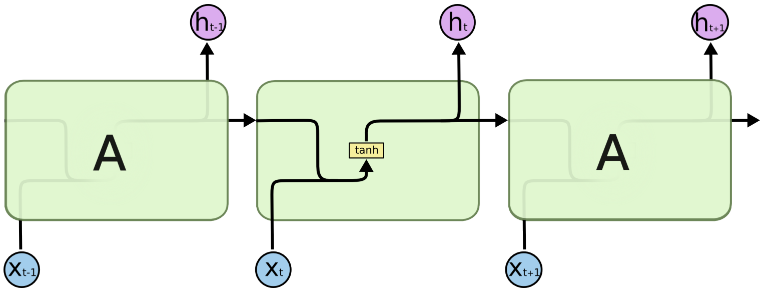
The RNN layer has many less parameters than the dense layer because:

1. a RNN cell only processes a single word; and
2. each word is processed by (a copy of) the same RNN cell.

In addition, since an RNN cell also receives the output of the previous copy of the RNN cell (which is processing the previous word), the RNN cell captures the context of the previous words.

A Simple Recurrent Neural Network

- A simple RNN cell ("vanilla RNN") has just a dense layer with an activation function (hyperbolic tangent, or "tanh" in the drawing below).
- Vanilla RNN cells have been used since 1990s.



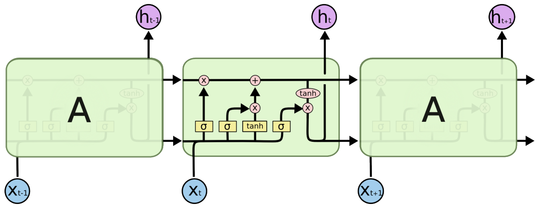
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTMs and GRUs

- Vanilla RNN cells are still too simple and they do not handle long-distance dependencies easily.
- More complex RNN cells have been designed specifically to address this issue.
- Current most popular RNN cells are:

  LSTM Long Short Term Memory (picture).

  GRU Gated Recurrent Unit; a more recent, simpler cell.



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNNs in Practice

- Most deep learning frameworks include special layers for RNNs.
- When you use an RNN layer, you have the option to specify the type of RNN cell (LSTM, GRU, etc.).
- You often have the option to access the state of the last cell, or the state of all cells.

Recurrent Neural Networks in Keras

The following notebook is based on the jupyter notebooks provided by the Deep Learning book: `https://github.com/fchollet/deep-learning-with-python-notebooks`

2 Pre-training and Fine-tuning

Problems with Supervised Learning

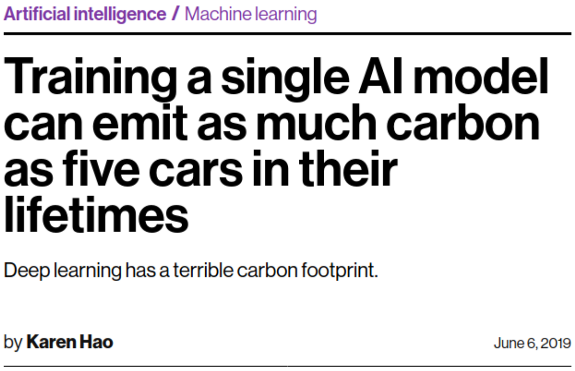
Annotated data

- 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.

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

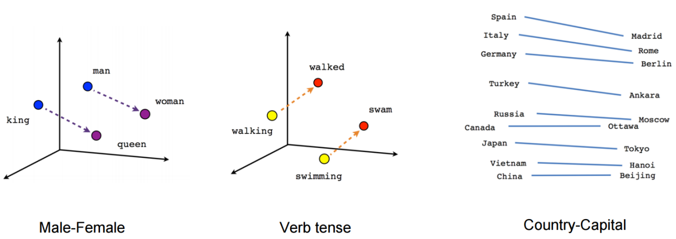
- Develop a system that can be trained with large volumes of data.
- Make the system as general as possible, so that it can be used for multiple tasks.

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

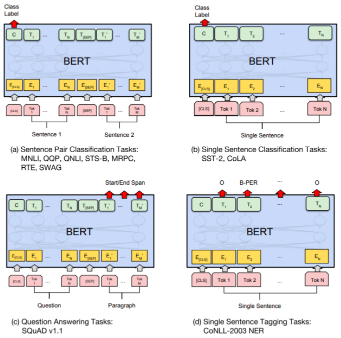
🤗 Transformers

`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.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

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

Take-home Messages

1. Use recurrent networks for text classification.
2. Transfer learning and fine-tuning.

What’s Next

Weeks 7-12

- Semantic Technologies (Rolf Schwitter).
- Assignment 2 submission deadline on Friday 22 April 2022.