Evolution of Neural Network with respect to NLP

Andrew Moses

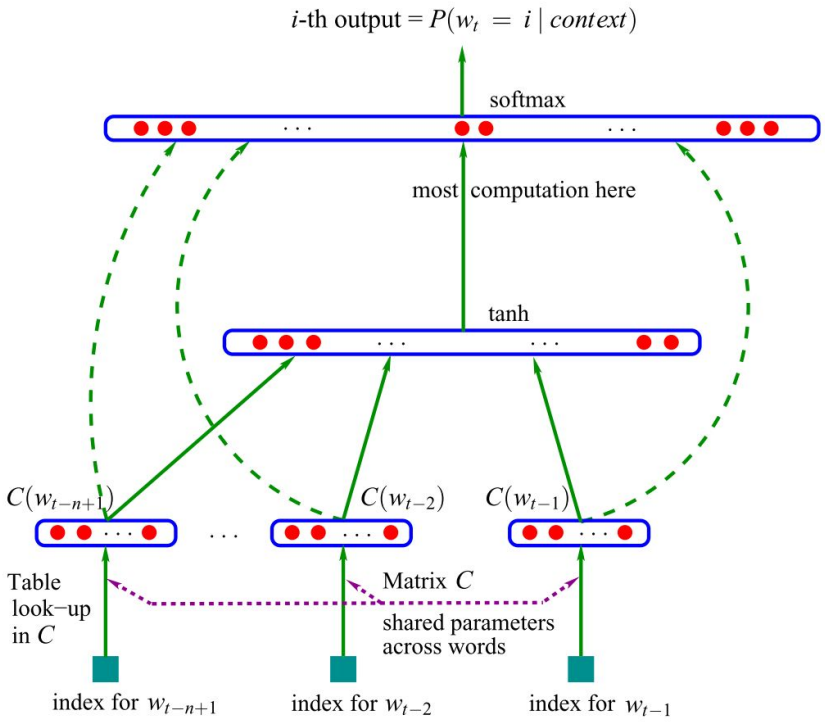
Language Modelling

Simply predict the next word in a given text.

Idea: Shannon posed the question: given a sequence of letters (for example, the sequence "for ex"), what is the **likelihood** of the next letter? From training data, one can derive a **probability distribution** for the next letter given a history of size *n*: a = 0.4, b = 0.00001, c = 0, ...; where the probabilities of all possible "next-letters" sum to 1.0.

The first neural language model, a feed-forward neural network was proposed in 2001 by Bengio

First Neural net for NLP



This model takes as input vector representations of the *n* previous words, which are looked up in a table *C*.

Nowadays, such vectors are known as word embeddings. These word embeddings are concatenated and fed into a hidden layer, whose output is then provided to a softmax layer.

NLP High level tasks

Word embeddings: The objective of word2vec is a simplification of language modelling.

Sequence-to-sequence models: Such models generate an output sequence by predicting one word at a time.

Pretrained language models: These methods use representations from language models for transfer learning.

This conversely means that many of the most important recent advances in NLP reduce to a form of language modelling. In order to do “real” natural language understanding, just learning from the raw form of text likely will not be enough and we will need **new methods** and **models**.

Word Embedding

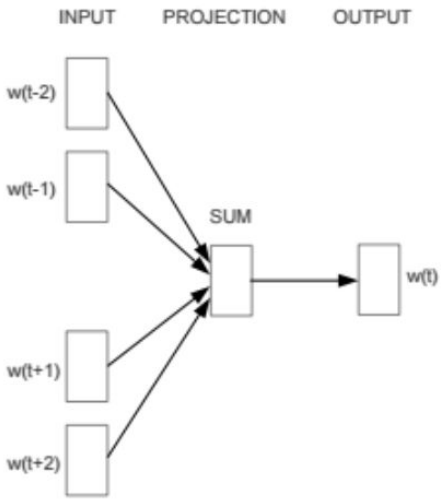
Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space

GloVe, coined from Global Vectors, is a model for distributed word representation. The model is an unsupervised learning algorithm for obtaining vector representations for words. This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

CBOW Vs Skip-gram

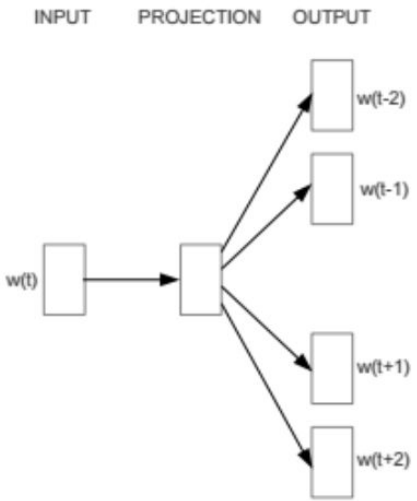
Given a set of (neighboring) words, **guess single words** that potentially occur along with this set of words.



CBOW

(Continuous Bag Of Words)

or



Skip-gram

Guess potential neighboring words based on the single word being analyzed.

Skip-gram: works well with **small amount of the training data**, represents well even **rare words** or phrases

CBOW: several times **faster to train** than the skip-gram, **slightly better accuracy for** the **frequent** words

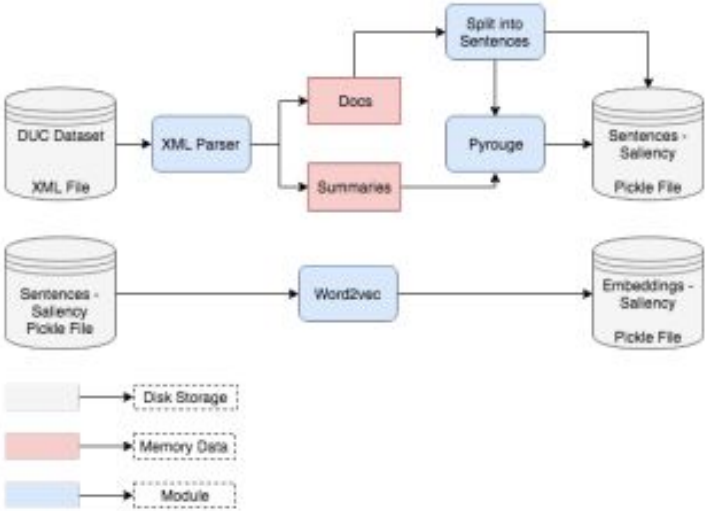
2013 and 2014 marked the time when neural network models started to get adopted in NLP.

Three main types of neural networks became the most widely used:

1. recurrent neural networks
2. convolutional neural networks
3. recursive neural networks.

Recurrent neural networks (RNNs) are an obvious choice to deal with the dynamic input sequences ubiquitous in NLP. Vanilla RNNs (Elman, 1990) were quickly replaced with the classic **long-short term memory networks** (Hochreiter & Schmidhuber, 1997), which proved more resilient to the vanishing and exploding gradient problem. Before 2013, RNNs were still thought to be difficult to train; Ilya Sutskever's PhD thesis was a key example on the way to changing this reputation. A **bidirectional LSTM** (Graves et al., 2013) is typically used to deal with both left and right context.

Extractive Document Summarization Using Convolutional Neural Networks - Reimplementation



Algorithm 1 Preprocessing

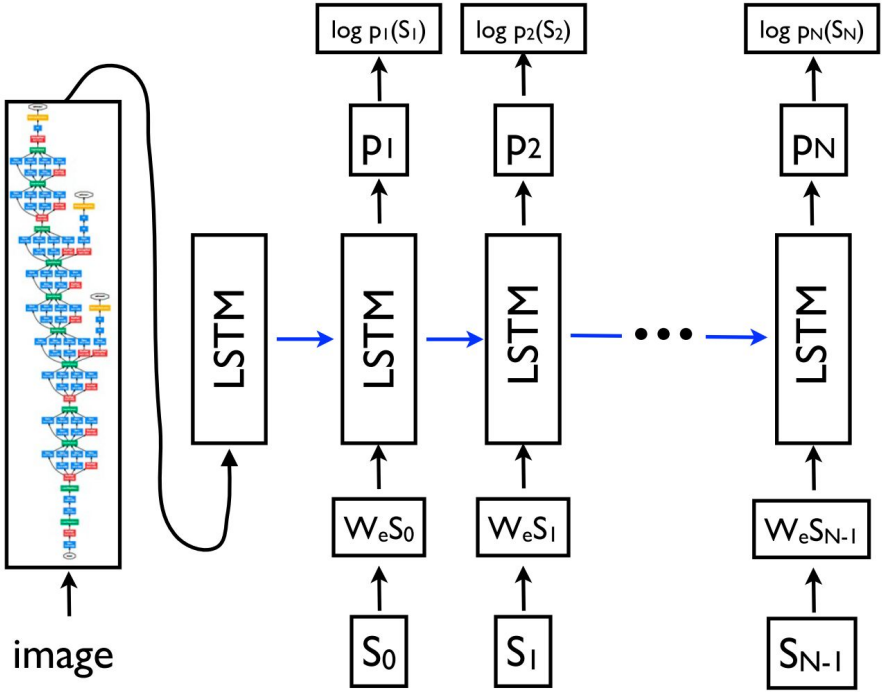
procedure PREPROCESSING(D, $summaries$)
  $y \leftarrow vector()$
  $X \leftarrow tensor()$
  for $doc_i \in D$ **do**
    for $s_j \in doc_i$ **do**
      $y_i \leftarrow salience(s_j, summaries_i)$
      for $word_k \in s_j$ **do**
        $X_{i,j} \leftarrow word2vec(word_k)$
return X, y

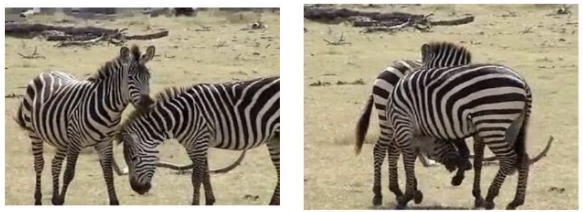
Source: https://leolaugier.github.io/doc/summarization.pdf

Sequence-to-sequence models

A general framework for mapping one sequence to another one using a neural network. In the framework, an encoder neural network process a sentence symbol by symbol and compresses it into a vector representation; a decoder neural network then predicts the output symbol by symbol based on the encoder state, taking as input at every step the previously predicted symbol

Encoders for sequences and decoders are typically based on RNNs but other model types can be used. New architectures mainly emerge from work in MT, which acts as a Petri dish for sequence-to-sequence architectures. Recent models are deep LSTMs (Wu et al., 2016), convolutional encoders (Kalchbrenner et al., 2016; Gehring et al., 2017), the Transformer (Vaswani et al., 2017) and a combination of an LSTM and a Transformer

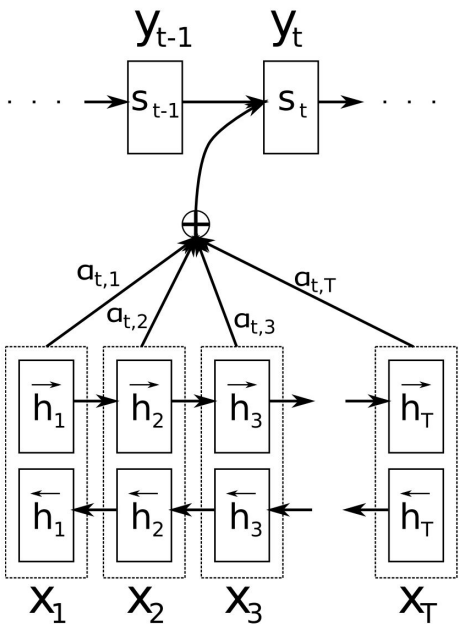




S2VT: A herd of zebras are walking in a field.

2015 – Attention

Attention (Bahdanau et al., 2015) is one of the core innovations in neural MT (NMT) and the key idea that enabled NMT models to outperform classic phrase-based MT systems. The main bottleneck of sequence-to-sequence learning is that it requires to compress the entire content of the source sequence into a fixed-size vector. Attention alleviates this by allowing the decoder to look back at the source sequence hidden states, which are then provided as a weighted average as additional input to the decoder.



Pretrained language models

1. Get To The Point: Summarization with Pointer-Generator Networks

Link: http://www.abigailsee.com/2017/04/16/taming-rnns-for-better-summarization.html

2. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Link: https://github.com/google-research/bert

3. Unified Language Model Pre-training for Natural Language Understanding and Generation

Link: https://github.com/microsoft/unilm

To Get Hands dirty

Transformers: State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch.

Link: https://github.com/huggingface/transformers