COMP3220 — Document Processing and the Semantic Web

Week 05 Lecture 1: Processing Text Sequences

Diego Mollá

COMP3220 2021H1

Abstract

This lecture focuses on some of the key aspects of text that make it different, and difficult, from other unstructured data from the point of machine learning, and how deep learning handles them. The first solution is to map words into continuous representations called embeddings. Using embeddings usually improves the quality of most text processing tasks. The second solution is the use of recurrent neural networks that can handle sequences of text.

Update March 21, 2021

Contents

Reading

- Deep Learning book, chapter 6.
- Understanding LSTM Networks, `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

Additional Reading

- Jurafsky & Martin, Chapter 9 (9.4 will be introduced in week 6)

1 Challenges of Text for Machine Learning

Words as Arbitrary Symbols

- Words are encoded as arbitrary symbols.
- Within one language there is no clear correspondence between a word symbol and its meaning.
  - "dig" vs. "dog"
  - "car" vs. "automobile"
- Different languages may use different representations of the same word.

https://en.wikipedia.org/wiki/File:Hello_in_different_languages_word_cloud.jpeg

Ambiguities Everywhere

Language features ambiguity at multiple levels.

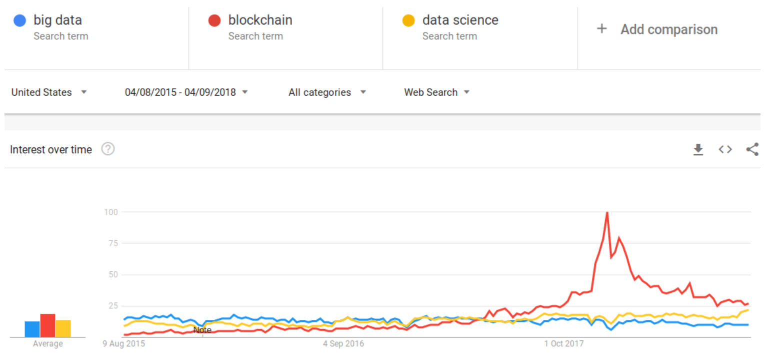
Lexical Ambiguity

Example from Google's dictionary:

- bank (n): the land alongside or sloping down a river or lake.
- bank (n): financial establishment that uses money deposited by customers for investment, ...
- bank (v): form in to a mass or mound.
- bank (v): build (a road, railway, or sports track) higher at the outer edge of a bend to facilitate fast cornering.
- ...

So many words!

- Any language features a large number of distinct words.
- New words are coined.
- Words change their use in time.
- There are also names, numbers, dates... an infinite number.



● big data Search term
● blockchain Search term
● data science Search term

United States · 04/08/2015 - 04/09/2018 · All categories · Web Search

https://trends.google.com

The plot shows the evolution of frequency of use of several terms. You can see that popularity of words can vary widely. At some point in time, new words are coined, and old words become no longer relevant.

Long-distance Dependencies

- Sentences are sequences of words.
- Words close in the sentence are often related.
- But sometimes there are relations between words far apart.

grammatical: "The man living upstairs . . . is very cheerful" "The people living upstairs . . . are very cheerful"

knowledge: "I was born in France and I speak fluent . . . French"

reference: "I bought a book from the shopkeeper and I liked it"

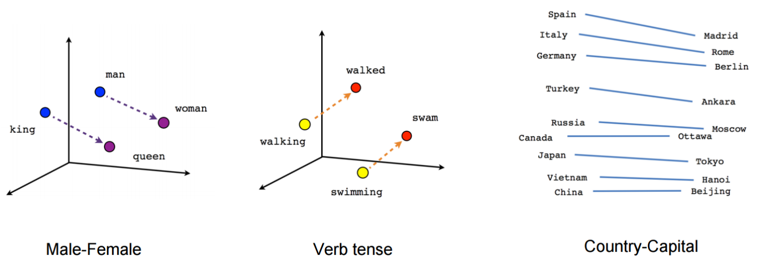
From the two examples above we can see that words that may be far from the gap determine what is the best word that fills the gap.

The third example shows that the reference of a pronoun can be fairly far from the pronoun itself.

2 Word Embeddings

Word Embeddings

- First introduced in 2013, nowadays is one of the most common ingredients in text processing systems.
- Word embeddings squarely aim at addressing the issue of representing words as continuous vectors of integers.
- Words with similar context are mapped to similar vectors.
- Embeddings are learnt using large, unlabelled training data.



https://www.tensorflow.org/tutorials/representation/word2vec

One-hot vs. word embeddings

One-hot

- Sparse
- Binary values (typically)
- High-dimensional
- Hard-coded

Word embeddings

- Dense
- Continous values
- Lower-dimensional
- Learned from data

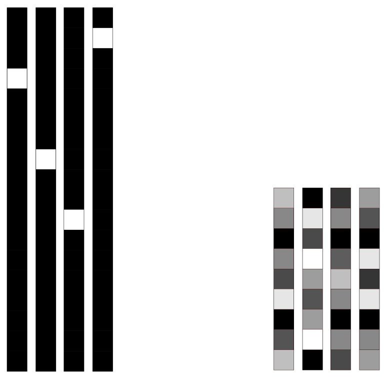


Image from Chollet (2018) "Dee Learning with Python:", Manning. Figure 6.2, page 184.

Two Ways to Obtain Word Embeddings

1. Learn the word embeddings jointly with the task you care about (e.g. document classification).
2. Use pre-trained word embeddings.

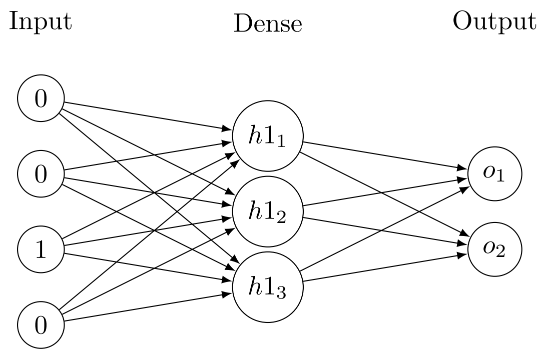
Learning Word Embeddings

- You can add a dense layer as the first layer of your network and let the system learn the optimal weights.
- This approach is so useful and common that many deep learning frameworks define an "embedding" layer that facilitates this.
- The input to the "embedding" layer is the word index.
- The output is the word embedding.

Embedding Layer as a Dense Layer

The input of the dense layer is the one-hot encoding of the word

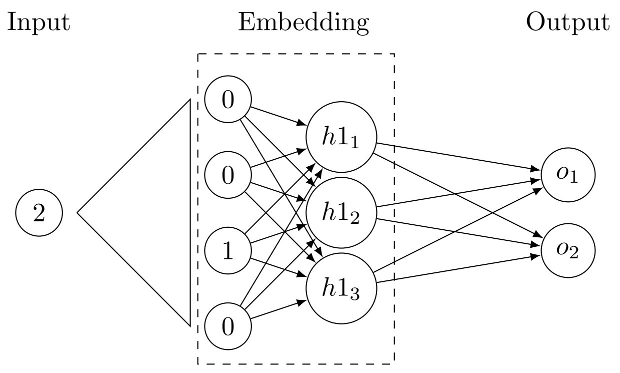
A Dense Layer



Embedding Layer in Keras

The input of a Keras embedding layer is a sequence of word indices which will be internally converted into their one-hot representations, followed by the dense layer.

A Keras Embedding Layer (for one word)



Processing Sequences of Words in Keras

- The input of a Keras embedding layers is a sequence of words.
- The output is a sequence of word embeddings.
- Since the layer will process a batch of samples at a time, each batch must have sequences with the same numbers of words.
- Keras provides a way to trim sequences of words or pad them to adjust the sequence length: pad_sequences.

Using pre-trained word embeddings

The Problem: Data Sparsity

- Sometimes we have so little training data that many words are poorly represented.
- Often, words in the training data do not occur in the test data.
- For these unseen words we would not be able to learn the embeddings.

A Solution: Pre-training

- Several people have computed word embeddings for large vocabularies using large data sets.
- We can then use these pre-trained embeddings to map from the word index to the word embedding.

Using Word Embeddings 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`
  - Using word embeddings.
- The notebook illustrates how you can use an embeddings layer for text classification, and how to load pre-trained word embeddings.
- This notebook is important because it also illustrates Keras' text tokenisation techniques.

3 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.

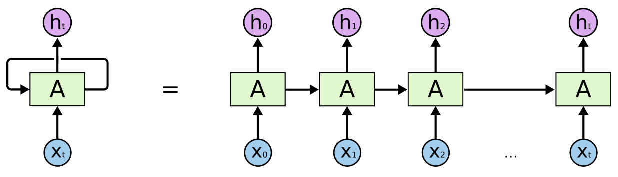
Why context matters

"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"

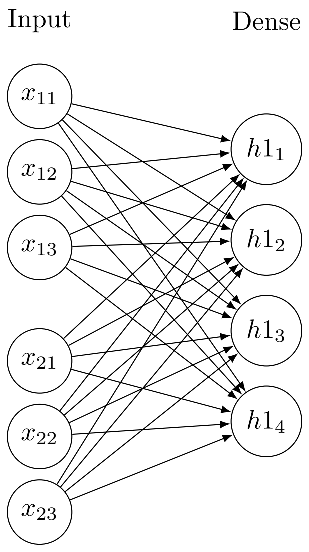
Recurrent Neural Networks

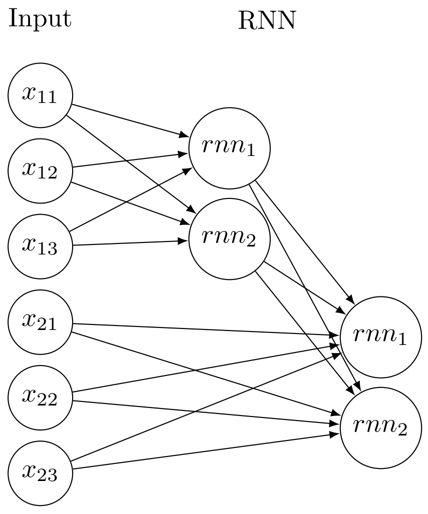
- A Recurrent Neural Network (RNN) is designed to process sequences.
- A RNN is a neural network that is composed of RNN cells.
- Each RNN cell takes as input two pieces of information:
  1. A vector representing an item x_i in the sequence.
  2. The state resulting from processing the previous items.
- 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 chain 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.

- The dense layer has 4 cells.
- The RNN layer is a simple dense layer (there are other possibilities).

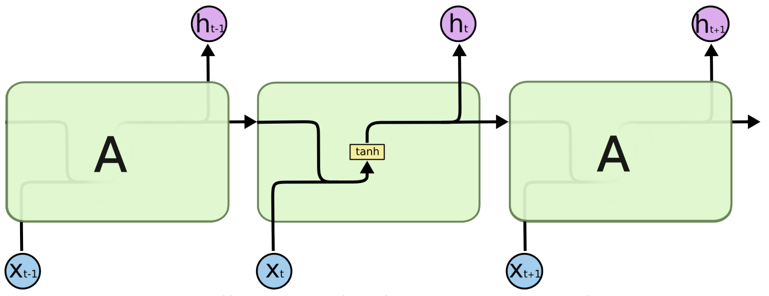
The RNN layer has many less parameters than the dense layer because:

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.



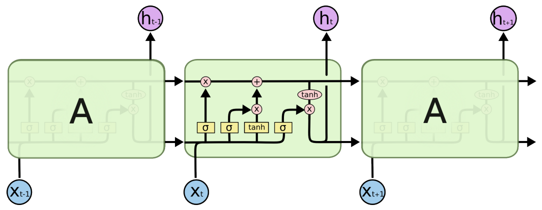
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

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.
- You often have the option to use the state of the last cell, or the state of all cells.

Recurrent Neural Networks in Keras

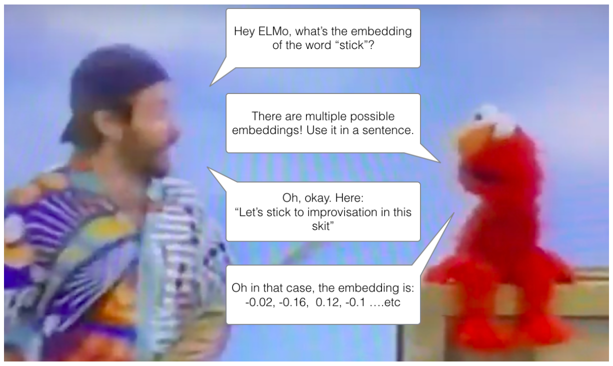
The following notebook is based on the jupyter notebooks provided by the Deep Learning book: `https://github.com/fchollet/deep-learning-with-python-notebooks`

- Understanding Recurrent Neural Networks.

The notebook illustrates how you can use an embeddings layer for text classification, and how to load pre-trained word embeddings.

Final Note: Contextualised Word Embeddings!

Recent research deviced a way to combine RNN and word embeddings to produce context-dependent word embeddings. The resulting systems are beating state of the art in many applications!


Hey ELMo, what's the embedding of the word "stick"?
There are multiple possible embeddings! Use it in a sentence.
Oh, okay. Here: "Let's stick to improvisation in this skit"
Oh in that case, the embedding is: -0.02, -0.16, 0.12, -0.1 ....etc

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

Take-home Messages

1. Explain some of the fundamental challenges that plain text represents to machine learning.
2. Apply word embeddings in deep learning.
3. Use recurrent neural networks for text classification.

What's Next

Week 6

- Advanced topics in deep learning
- Reading: Deep Learning book, chapter 8.1
- Additional reading: Jurafsky & Martin, Chapter 9