

ITMO UNIVERSITY

NLP – Latest Technologies

Google BERT, GPT-2, Transformer XL, and others ...

May 1st, 2019

Outline of this unit

- ✓ Overview of current NLP approaches
 - Short intro to **very recent NLP tools and techniques**: UMLfit, Elmo, BERT, ...
- ✓ **fastai** and an exercise
- ✓ Next week: Liubov will teach

Resources

- ✓ Simple intro to **various pretrained models**:
<https://www.analyticsvidhya.com/blog/2019/03/pretrained-models-get-started-nlp/>
- ✓ Ivan Bilan: Understanding and Applying **Self-Attention for NLP**
<https://www.youtube.com/watch?v=OYygPG4d9H0>
- ✓ **Attention** is all you need:
<https://www.youtube.com/watch?v=iDulhoQ2pro>
- ✓ **BERT** introduction:
https://www.youtube.com/watch?v=0EtD5ybnh_s
- ✓ Slides for Bert intro:
http://redcatlabs.com/2018-11-21_TFandDL_BERT/#/1/1

Resources

- ✓ Visualization of RNNs:
<https://distill.pub/2019/memorization-in-rnns/>
- ✓ <https://github.com/facebookresearch/InferSent>

Starting point: word embeddings like word2vec

- ✓ Word Embeddings only pretrained ingredient 2013-2018
- ✓ Skipgram: predict context from a word
- ✓ CBOW: predict word from a context
- ✓ .. as we have already seen ..
- ✓ Tricks like **negative sampling** and **hierarchical softmax** to optimize training speed

Next step: Skip-thoughts and Quick-thoughts

- ✓ [In my understanding]
- ✓ Like word2vec, but here we don't predict words, but (next) **sentences**.
- ✓ Sentence is encoded with an RNN.
- ✓ Again **negative sampling** is used to help learn which sentences appear together.
- ✓ This task is borrowed in **BERT** (later)

NLP in the last few years: RNN / LSTM / GRU

- ✓ They encode a sentence word by word.
- ✓ Gates in LSTM / GRU decide what is important etc
- ✓ In the end there is sort of a sentence embedding (encoded sentence) “**encoder**”
- ✓ **Decoder** then spits out a sequence (eg. in machine translation) also
- ✓ Problem with LSTM .. long term dependencies, long sentences etc

Step (2018): add Transfer Learning

- ✓ **InferSent:** BiLSTM and max pooling to create a sentence encoding, from FB research:
<https://github.com/facebookresearch/InferSent>
- ✓ **General Purpose Sentence representations:** using 4 supervised tasks → **multi-task learning** – which give better performance on each task
- ✓ Transfer learning on a different task
- ✓ → contextualized word vectors

ELMo (Embeddings from Language Models)

- ✓ **Deep contextualized** word representations
- ✓ 2-stacked BiLSTM and residual NNs, residual helps to make it more stable
- ✓ **Basic usage is quite simple, and has some options**
 - **let's look here:**
<https://tfhub.dev/google/elmo/2>
- ✓ **Show CoLab with ELMo spam classifier example**
 - Code is also on github: https://github.com/gwohlgen/colab/blob/master/Spam_Classification_with_ELMo.ipynb

UMLFiT

- ✓ “Universal Language Model Fine-tuning for Text classification”
- ✓ First popular transfer learning model
- ✓ <https://arxiv.org/abs/1801.06146> (2018, J Howard and S Ruder), “Universal Language Model Fine-tuning for Text Classification”
- ✓ **Idea:** Pretrained model → fine-tune on target corpus → train target classifier (we will see it in the end with the fastai example)
- ✓ Needs only little training examples

UMLFiT (2)

- ✓ UMLFiT uses **some tricks in training**, eg:
- ✓ Different learning rates on different layers
- ✓ Slanted triangular learning rates
- ✓ Gradual unfreezing

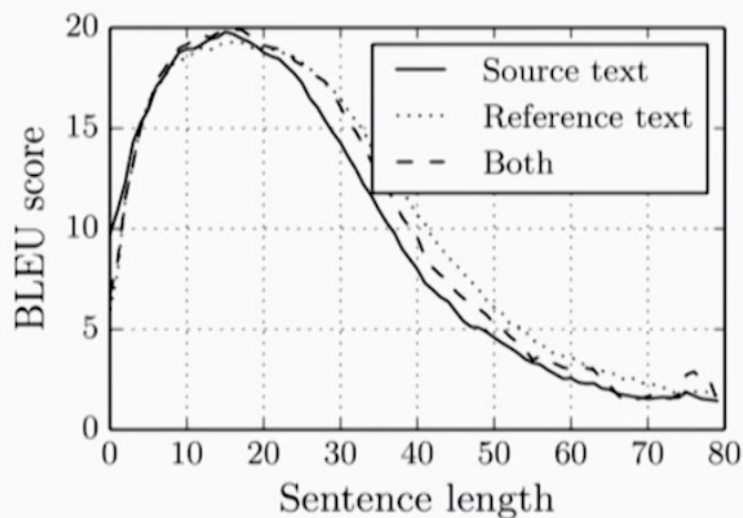
Transformer

- ✓ Uses self-attention
- ✓ Switch away from RNNs to Attention happening in **mid-late 2018**
- ✓ Problem of RNN: sequential processing, long range dependencies, etc.
- ✓ New architectures based on attention: BERT, TransformerXL ...

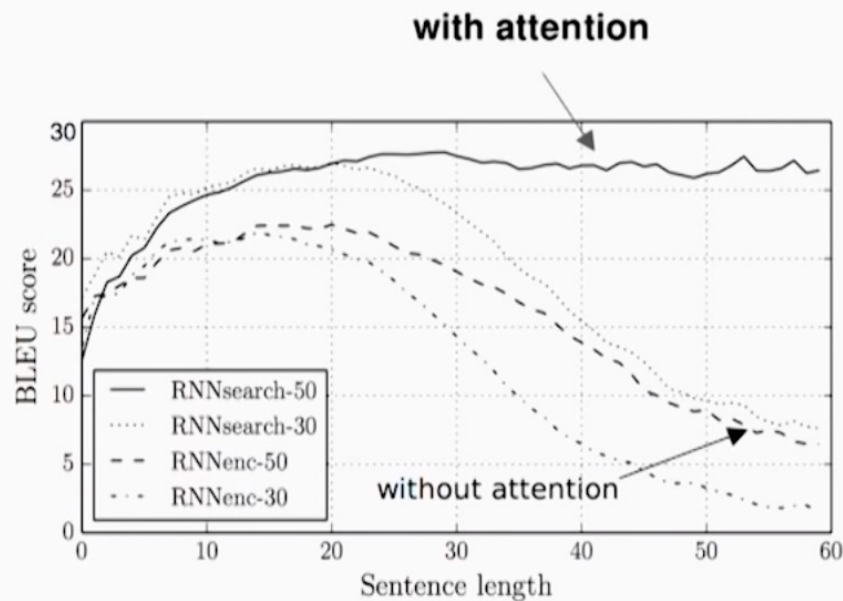
(Self-)Attention

- ✓ “Attention is all you need” – <https://arxiv.org/abs/1706.03762>
- ✓ Attention can be used to improve LSTMs or CNNs
- ✓ Attention to focus on specific parts of the text (like a human)
- ✓ Looks at the whole sentence at once
- ✓ Shown that in NMT attention helps to translate long sentences (with LSTM)
- ✓ Attention can help also in text classification (attend to the sentences which are important for the classification)
- ✓ Also possible to just skip LSTM/CNN, and just use attention layers

NMT with LSTMs



NMT with LSTMs + attention



Bahdanau et al. 2014
Slide: Text generation with attention, GTC 2017, Valentin Malykh (2017)

(Self-)Attention

- ✓ With Attention, the **decoder** can directly look at (hidden) states of specific words – in contrast to an RNN where you only have the one decoder state
- ✓ In every decoder step you do attention over everything
- ✓ Input: word embeddings + **pos. encodings**
- ✓ On the input sentences – doing attention basically means on which words to look at more and less strongly – result is one encoded representation
- ✓ Same on output sentence, put current state (eg 2 words) into it – like for input sentence about
- ✓ Then another attention block to combine input and output representations – 3 connections: **keys, values, queries**
- ✓ All with multi-head attention, so different things to be looked at (focused) in input sequences

(Self-)Attention

- ✓ In K,V,Q block: dot-prod of keys and queries. See their angle, do they align?
- ✓ Each Key Vector has a value.
- ✓ The softmax (dot-prod) basically selects one of the keys. Softmax is basically a differentiable max() function – so it basically selects a values (it's like an indexing scheme)
- ✓ The values are the interesting things, and the keys are ways to indexed those. And the queries are ways to select those values via the keys.
- ✓ **Summary:** attention reduces path-length
- ✓ **See eg also:**
<https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04>

Google BERT – Introduction

- ✓ BERT (Bidirectional Encoder Representations from Transformers)
- ✓ Bidirectional training with transformers, using **attention-based** mechanisms.
- ✓ **No** RNN / LSTM sequential input (left-to-right, right-to-left)
- ✓ **Look at:** http://redcatlabs.com/2018-11-21_TFandDL_BERT/#/1/1

Google BERT – Usage (Idea)

- 1) Use the existing pretrained model
- 2) Add only **one** (or few) additional layers
- 3) Train the model for the given task
("Finetuning")

Google BERT – ... things ..

- ✓ If you want to use BERT for a classification task, you can use the random [CLS] token at the beginning
- ✓ Not sure if multi-label classification is possible with BERT

Google BERT – Training

- ✓ BERT reads the whole input sentence **at once** (by adding positional embeddings).

References for BERT:

- <https://arxiv.org/pdf/1810.04805.pdf>
- <https://github.com/google-research/bert>
- <https://arxiv.org/abs/1905.05950> (shows that BERT emulates classic NLP pipeline)
- and others

Character Embeddings

- **Why:** OOV words, infrequent words, misspelled words, emoticons, small amount of vectors(!), etc.
- **Have a look here:**
<https://towardsdatascience.com/besides-word-embedding-why-you-need-to-know-character-embedding-6096a34a3b10>
- Basic Character-CNN, with 70 characters
- How? See link above

BERT and CoLab

- **Have a look here:**

<https://towardsdatascience.com/how-to-do-text-binary-classification-with-bert-f1348a25d905>

- **BERT CoLab notebook:**

https://colab.research.google.com/github/google-research/bert/blob/master/predicting_movie_reviews_with_bert_on_tf_hub.ipynb#scrollTo=xiYrZKaHwV81

- **GitHub:**

- https://github.com/wshuyi/demo-text-binary-classification-with-bert/blob/master/bert_text_classification.ipynb

GPT-2



- GPT-2 only released the **small model (117M parameters)**, results with it are not too great
- But maybe the model would be good input for fine-tuning on some language task
- Explore: <https://gpt2.apps.allenai.org/?text=Joel%20is>

GPT-2

- Reference:
<https://openai.com/blog/better-language-models/>
- The model, called GPT-2 (a successor to GPT), was trained simply to **predict the next word** in 40GB of Internet text.
- GPT-2 is a large **transformer-based** language model with 1.5 billion parameters.

GPT-2

- In the playground, the model get's some **input** and produces some continuation. The input can be Q/A pairs or whatever.
- GPT-2 only released the **small model (117M parameters)**, results with it are not too great
- But maybe the model would be good input for fine-tuning on some language task

 gpt-2-playground_ipynb 
File Edit View Insert Runtime Tools Help
+ CODE + TEXT ↑ CELL ↓ CELL COPY TO DRIVE RAM Disk
Model prompt >>> The best Vodka in Russia is made out of stones.
...
===== SAMPLE 1 =====
I only drink vodka with vodka. So I've seen some Russian vodka, but there are no stones. I don't like drinking vodka with vodka. I like vodka.
There's another Russian vodka made out of stones. I haven't tried it yet. In my opinion, Russian vodka is better than vodka made from ice.
Vodka from the Ural Mountains
Vodka from the Ural Mountains is quite different from vodka. It has a very strong flavor and a strong flavor is very important. It's made out
There are few things that have to be done in preparation for Vodka from the Ural Mountains. We make vodka using stone, we use vodka in the var
Vodka from the Ural Mountains is a very strong, very strong and very cold spirit. It's made out of stones!
Vodka is made from stones!
Vodka is made out of stones and cold!
They are made out of cold.
You can see that there are several stones that are used to make vodka from stones. You can see all these stones.

GPT-2 .. Trying out

- Playground .. open in Google CoLab:
https://colab.research.google.com/github/ilopezfr/gpt-2/blob/master/gpt-2-playground_.ipynb
- It's quite easy to use, you can play around in interactive mode to see the text generation capabilities.
- **Show on local machine:**

```
wohlg@wohlg-XPS:~/itmo/gpt-2$ python  
src/interactive_conditional_samples.py
```

GPT-2 – Fine-Tuning (Chat Messages)

- Example 1:
<https://svilentodorov.xyz/blog/gpt-finetune>
- This guy used all his Facebook chat messages (14M) to fine-tune GPT-2
- After that, the system created more or less realistic new chats with new persons
- Example code provided (jupyter/CoLab notebook)

GPT-2 – Fine-Tuning (Poetry)

- Example 2: <https://www.gwern.net/GPT-2>
- Fine-tuning for poetry generation – on public domain Project Gutenberg poetry corpus
- Very nice article with lots of ideas for future work in the field

Transformer XL (2019) - Resources

- Blog article:
<https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html>
- Paper: <https://arxiv.org/abs/1901.02860>
- Another blog:
<https://medium.com/dair-ai/a-light-introduction-to-transformer-xl-be5737feb13>

Transformer XL – Background

✓ Limitations of BERT etc:

- Limited window size / text needs to be split in segments
- The algorithm is not able to model dependencies that are longer than a fixed length (eg 512 words).
- The segments usually do not respect the sentence boundaries, resulting in context fragmentation which leads to inefficient optimization.

✓ Transformer-XL enables natural language understanding beyond a fixed-length context.

Transformer XL – Background (2)

- We want larger context!
- Segment-level Recurrence
 - largest possible dependency length by N times, where N is the depth of the network
 - Look at: <https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html>
- Relative positional embeddings
 - In order to capture the positions in different input segments

Transformer XL – Usage

- Seems like it is mostly used in **language modelling** currently
- Did not see any pretrained models (except the ones included in the paper) yet
- Also didn't see any experiences yet (May 2019) on how to fine-tune those models or train on new tasks

FastAI

- <https://www.fast.ai/>
- They try to make deep learning easy and accessible
- Very good online course – if you want to get deeper into AI (incl NLP)
- Look at notebook on colab (and github) cooking multilabel example!

FastAI - Exercise

- We use a the spam classification task again – binary text classification (positive, negative)
- About 5000 examples of spam / ham – unbalanced classes
- Create a classifier with **fastai**

FastAI – Exercise - Steps

1) Download the datasets from

<https://raw.githubusercontent.com/gwohlgen/misc/master/spam.csv>

2) Based on the example jupyter notebook:

- Finetune a language model
- Learn a classifier