4. Transformers, Attention (BERT, GPT-3, etc.) by The AI Epiphany

Saturday, September 26, 2020 11:52 AM

WORD EMBEDDINGS

☑ https://machinelearningmastery.com/what-are-word-embeddings/ <- nice recap of the word embeddings (Word2Vec/GloVe)
☑ https://jalammar.github.io/illustrated-word2vec/ <- (THE BEST EXPLANATION)

Neural language model (2003) - where it all started:
☑ https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

word2vec papers (shallow representations compared to pretrained (on LM task) NLP models)
☑ https://arxiv.org/pdf/1301.3781.pdf
☑ https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
☐ https://nlp.stanford.edu/pubs/glove.pdf <- GloVe

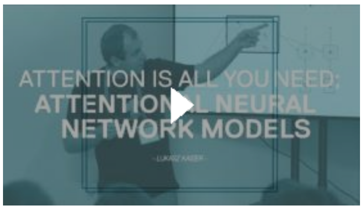
ATTENTION
blogs
☑ https://medium.com/@shashank7.iitd/understanding-attention-mechanism-35ff53fc328e <- high level introduction to the attention mechanism
☑ https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3 <- high level intro to attention mechanism
☑ https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/ <- same (THE BEST EXPLANATION)

papers
☑ https://arxiv.org/abs/1409.0473 (attention mechanism) Neural Machine Translation By Jointly Learning to Align and Translate
☑ https://arxiv.org/abs/1508.04025 Effective Approaches to Attention-based Neural Machine Translation

TOKENIZATION

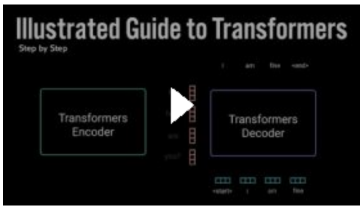
☑ Word-piece
☑ https://blog.floydhub.com/tokenization-nlp/
☐ http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture12-subwords.pdf

SELF-ATTENTION AND TRANSFORMERS
☑ https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a <- high level intro into self-attention mechanism
☑ https://jalammar.github.io/illustrated-transformer/ <- THE BEST EXPLANATION
☑ https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html <- not so useful
☑ https://ai.googleblog.com/2017/06/accelerating-deep-learning-research.html <- not so useful
☑ Attention is all you need; Attentional Neural Network Models | Łukasz Kaiser | Masterclass <- naah not so useful



☑ Transformer Neural Networks - EXPLAINED! (Attention is all you need) <- really nice resource

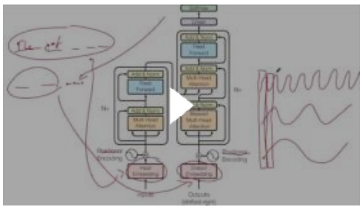
☑ Illustrated Guide to Transformers Neural Network: A step by step explanation<- also very nice



☑ https://medium.com/@Alibaba_Cloud/self-attention-mechanisms-in-natural-language-processing-9f28315ff905
☑ https://arxiv.org/abs/1706.03762 transformer paper again

☑ AI Language Models & Transformers - Computerphile <- super high level on language modeling and transformers

☑ Attention Is All You Need <- reiterates what I learned from the paper



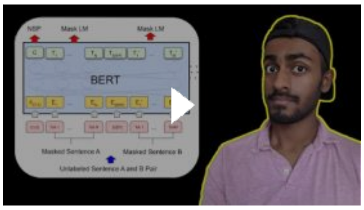
RECAP OF VARIOUS ATTENTIONS
☑ https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html <- introduced me to ideas of neural turing machines, pointer nets and self-attention GANs

BERT

☑ https://jalammar.github.io/illustrated-bert/ (the best resource)

Official Google blogs
☑ https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html
☑ BERT Google blog: https://www.blog.google/products/search/search-language-understanding-bert/

☑ BERT Neural Network - EXPLAINED! <- easy to follow along



Next steps:
☑ https://ruder.io/nlp-imagenet/ <- NLP's ImageNet moment -> ELMo, ULM-FiT, GPT

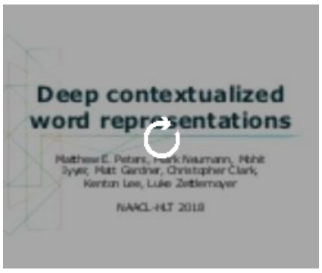
We can do better than Word2Vec and GloVe
“stick” has multiple meanings depending on where it’s used. Why not give it an embedding based on the context it’s used in – to both capture the word meaning in that context as well as other contextual information?”. And so, contextualized word-embeddings were born:
☑ https://techcrunch.com/2018/06/15/machines-learn-language-better-by-using-a-deep-understanding-of-words/
☑ https://arxiv.org/abs/1708.00107 <- Learned in translation (CoVE) (context dependent but only used last layers)
☑ https://arxiv.org/abs/1802.05365 <- deep contextualized word representations (ELMo) (context dependent and used all layers - semantics/syntax information is distributed from higher to lower layers respectively)
☑ https://arxiv.org/abs/1801.06146 <- universal language model ULM-FiT (enabled transfer learning in NLP)
☑ https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf GPT
☑ https://arxiv.org/abs/1511.01432 <- semi-supervised seq learning

☑ BERT paper

Next steps:
☑ BERT dependency and BERT
☑ GPT blogs
☑ GPT papers
☑ Verge blog

Pool of resources (later, maybe):

☐ http://web.stanford.edu/~jurafsky/slp3/3.pdf <- read the whole article (I finished the perplexity section)
☐ ELMo slides: A Review of Deep Contextualized Word Representations (Peters+, 2018)


Deep contextualized
word representations
Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer
NAACL-HLT 2018

GPT-family

☑ https://openai.com/blog/language-unsupervised/ <- GPT OpenAI blog

☑ https://openai.com/blog/better-language-models/ <- GPT-2 OpenAI blog (I also read some other blogs on their website and followed different links like the GPT-2 model card and various continuations of the blog on GPT-2)

☑ http://jalammar.github.io/illustrated-gpt2/ <- GPT-2 (AMAZING resource as always!)
☑ https://jalammar.github.io/how-gpt3-works-visualizations-animations/ <- GPT-3
☑ https://openai.com/blog/ai-and-compute/

☑ https://www.theverge.com/21346343/gpt-3-explainer-openai-examples-errors-agi-potential <- verge blog (nice collection of what happened on social media and in general around GPT-3 after it was first published
☑ https://pagestlabs.substack.com/p/gpt-3-and-a-typology-of-hype <- useful around GPT-3 hype
☑ https://minimaxir.com/2020/07/gpt3-expectations/ <- nice overview of the hype also (and model limitations, etc.)
☑ https://lacker.io/ai/2020/07/06/giving-gpt-3-a-turing-test.html <- nice examples of GPT-3's pitfalls (turing test)

PAPERS
☑ https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf <- GPT
☑ https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf <- GPT-2
☑ https://arxiv.org/abs/2005.14165 <- GPT-3 (huge paper 75 pages I read the most important parts)

NLP/transformers overview resources

OVERVIEW BLOGS:
☑ https://medium.com/@Moscow25/the-best-deep-natural-language-papers-you-should-read-bert-gpt-2-and-looking-forward-1647f4438797 <- really nice overview of the SOTA NLP papers and in general nice blog
☐ Transformers Hugging Face: https://medium.com/huggingface/encoder-decoders-in-transformers-a-hybrid-pre-trained-architecture-for-seq2seq-af4d7bf14bb8
☐ https://github.com/keon/awesome-nlp <- useful curated NLP list (after I read everything else potentially find some inspiration here

☑ https://ruder.io/nlp-imagenet/ <- NLP's ImageNet moment

☑ Is there something really popular aside from BERT and GPT families? -> those were the most influential ones

☑ http://www.incompleteideas.net/IncIdeas/BitterLesson.html<- awesome short essay by Rich Sutton

Fun fact: Google and Microsoft Bing are using BERT for search

RESEARCH TIPS:
☑ https://ruder.io/10-tips-for-research-and-a-phd/ <- read this one
☑ https://bigaidream.gitbooks.io/tech-blog/content/2014/de-mystifying-good-research.html <- writing skill is super important for publishing papers
☑ https://karpathy.github.io/2016/09/07/phd/ <- many interesting advices (most of which I knew)
☑ https://www.andreykurenkov.com/writing/life/lessons-learned-from-failures/ <- AI researcher openly talking about the failures and depression, nice
☐ https://ruder.io/requests-for-research/

Later work

☑ https://arxiv.org/abs/1901.02860 <- TransformerXL (Google)
☑ https://arxiv.org/abs/1906.08237 <- XLNet (Google) (uses Transformer XL as a backbone) (512 TPU v3s)

☑ https://arxiv.org/abs/1907.11692 <- RoBERTa (FAIR) (couple of lines of code in PyTorch) (1024 V100s) (strong dependency on BERT and XLNet)
☑ https://arxiv.org/abs/1909.11942 <- ALBERT (Google) (64-512 TPU v3s)

☐ https://nv-adlr.github.io/MegatronLM <- Megatron
☐ https://arxiv.org/abs/1910.10683 <- Google T5 (Very long)

☐ Again read through the rest of result section on GPT-3 (I'd love to make a video on this one)

☑ SparseTransformer <- introduced O(n*sqrt(n)) complexity which is an improvement over O(n^2)
☐ Longformer, Linformer

Maybe read:
☐ https://arxiv.org/abs/1910.13267 <- BPE dropout
☐ https://karpathy.github.io/2015/05/21/rnn-effectiveness/

☐ https://www.gwern.net/GPT-3 <- the most intricate testing of GPT-3
☐ https://openai.com/blog/preparing-for-malicious-uses-of-ai/
☐ https://transformer.huggingface.co/ <- writing with transformers (web app)
☐ https://thegradient.pub/understanding-evaluation-metrics-for-language-models/

Things I should know before jumping into GNNs

☐ What exactly is BLEU (machine translation metric)?
☑ What exactly is perplexity (language model metric)?
☑ https://huggingface.co/transformers/perplexity.html <- go through this (ppl)
☑ What exactly is F1 (metric)?
☑ What exactly is beam search
☑ What exactly is word-piece, byte-pair encodings
☑ What is the N-grams language model

☐ https://thegradient.pub/transformers-are-graph-neural-networks/
☐ How to apply attention to imagery (self attention GAN)
☐ https://openreview.net/pdf?id=YicbFdNTTy <- Vision Transformer

CODE

TUTORIALS
☑ https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/ <- playing with BERT (gave me understanding of how easy it is to use transformers lib from HuggingFace)
☑ (2017) https://pytorch.org/tutorials/beginner/transformer_tutorial.html(reading PyTorch docs to understand it + some blogs to understand torchtext lib, jupyter notebook + debugging in PyCharm) < it's an LM and not NMT model
☑ https://github.com/pytorch/pytorch/blob/187e23397c075ec2f6e89ea75d24371e3fbf9efa/torch/nn/modules/transformer.pyOfficial PyTorch implementation
☑ (2018) https://towardsdatascience.com/how-to-code-the-transformer-in-pytorch-24db27c8f9ec <- how to build a transformer tutorial (naah not the best resource out there)
☑ (2018) http://nlp.seas.harvard.edu/2018/04/03/attention.html<- transformer code in PyTorch coupled with theory! (The Annotated transformer)

☐ Read BLEU paper from 2002 https://www.aclweb.org/anthology/P02-1040.pdf, https://machinelearningmastery.com/calculate-bleu-score-for-text-python/

☐ Take a look at Tensor2Tensor implementation of the transformerTake a look at Tensor2Tensor impleme

Get some familiarity with Hugging Face's transformers library
Go through their notebooks
☐ https://github.com/huggingface/transformers<- (Hugging face) definitely the place to start with code
☐ https://medium.com/huggingface/distilbert-8cf3380435b5 <- Distil BERT by HuggingFace
☐ https://colab.research.google.com/github/huggingface/blog/blob/master/notebooks/02_how_to_generate.ipynb#scrollTo=KxLvv6UaPa33
☐ https://colab.research.google.com/github/huggingface/transformers/blob/master/notebooks/02-transformers.ipynb#scrollTo=t7UatPZzbeDR
☐ https://colab.research.google.com/github/huggingface/blog/blob/master/notebooks/01_how_to_train.ipynb
☐ https://colab.research.google.com/github/huggingface/blog/blob/master/notebooks/trainer/01_text_classification.ipynb
☐ https://colab.research.google.com/github/ViktorAlm/notebooks/blob/master/MPC_GPU_Demo_for_TF_and_PT.ipynb

Final set of relevant resources:

1. Annotated transformer
2. Illustrated transformer
3. PyTorch tutorial + PyTorch transformer
4. paper

If I get stuck start consulting other blogs/repos I found (probably won't happen)

torchtext
☑ https://mlexplained.com/2018/02/08/a-comprehensive-tutorial-to-torchtext/ <- helped me a bit with torchtext
☐ https://towardsdatascience.com/deep-learning-for-nlp-with-pytorch-and-torchtext-4f92d69052f (may help with torchtext)

Understanding multiheadattention details:
https://github.com/pytorch/pytorch/blob/master/torch/nn/functional.py#L4011
https://pytorch.org/docs/stable/_modules/torch/nn/modules/activation.html#MultiheadAttention

TensorFlow resources (maybe go through them)
☐ https://colab.research.google.com/github/tensorflow/tpu/blob/master/tools/colab/bert_finetuning_with_cloud_tpus.ipynb
☐ https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb<- another potentially useful resource

☑ https://tunz.kr/post/4 <- useful blog

Torch text did not do an excelent job at documenting I had to dig deeply into their TranslationDataset/Dataset/Field/Example source code

☑ https://pytorch.org/text/_modules/torchtext/vocab.html <- how vocab is constructed

TIME STATS:

RTX 2080, G2E, IWSLT, 5 EPOCHS, 1500 TOKEN/BATCH -> 66 MINUTES -> **13.2 min/epoch**

K-80, G2E, IWSLT, 1500 TOKEN/BATCH -> ~**34 min/epoch**
K-80 E2G, WMT-14, 1500 TOKEN/BATCH -> **16h and 20 min/epoch**

Note: I had some bug so I had to use only 1.5k instead of 3.5k on K80 GPU

DATASETS STATS:

WMT-14
Train: E=128M tokens, G=132M tokens
(~4.5 M sentence pairs I verified)
Validation: E=76k G=81K tokens

IWSLT:
Train: E=3.9M tokens, G = 3.6M tokens
Validation: E=21k G=19.5K

Train dataset is ~x34 times bigger for WMT-14!

It seems they made around 19 epochs on the WMT-14,
Or 100k steps using 25.000 tokens/batch (that's for the baseline model!)