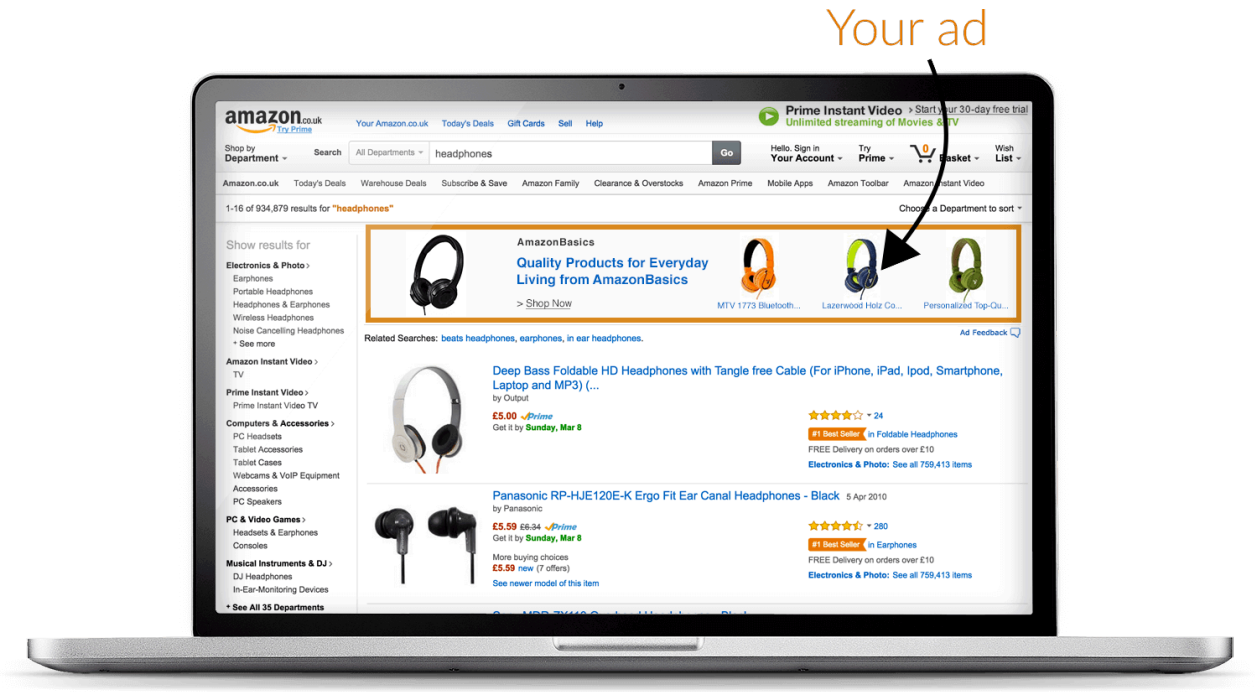
Generating Ad Text

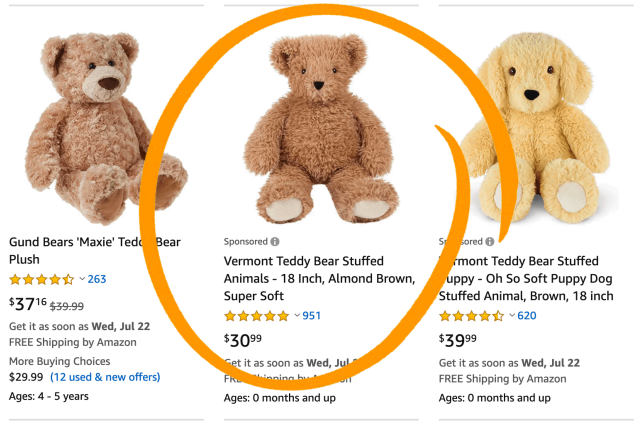


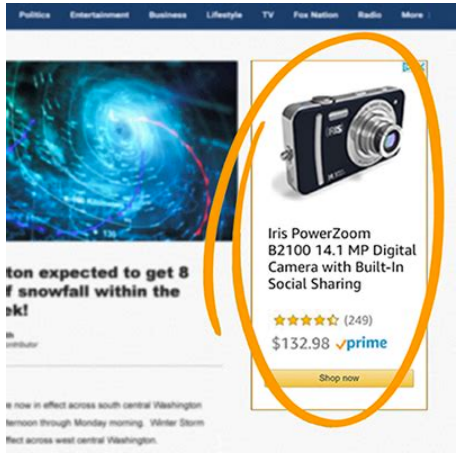
NIRAJ SARAN

MAY 17, 2022

Problem Statement

- Amazon Ads: 6M Sellers
  - Create new creative ads quickly and efficiently, especially for fast moving consumer electronics
- Can we better the state-of-the-art pre-trained models?
- Extend to Google/Meta Ads and other NLG domains?





Breakdown

- Deep Learning models are super resource intensive - how good can our LSTM (Long Short-Term Memory) Recurrent Neural Network (RNN) model really be?
- How do we optimize for compute resources and maintain quality
- Compare with pre-trained Causal Large Language models like OpenAI's GPT-3 and EleutherAI's GPT-J/NEO
- Can we fine-tune these pre-trained models on our ads dataset?
- How well does our fine-tuned model perform? What metrics?
- Can we open-source publish our model for others to fine-tune themselves?

Methodology - RNN LSTM with Embedding layer

1. Why LSTM - good for sequential data, generalize across sequences
2. Tensorflow Keras model
   - Amazon ads dataset (2M rows, one column, 40 categories, 250MB. Not much data cleanup or EDA!
   - Keras Embedding layer to vectorize each word along with semantic information
   - Tokenize, map to IDs, subclassed Keras Sequence object creates sequences for input and next words
   - Self-supervised learning
   - TextDataGenerator - Python generator outputs batches of data (X and y)
   - Predict: vector of size = vocabulary; probability assigned to each word in corpus
3. Google Colab with .py and GPU: 8x performance improvement
4. Comparison with OpenAI's GPT-2 and EleutherAI's gpt-neo-1.3B - zero shot
5. Fine-tuned pre-trained model EleutherAI's gpt-neo-125M with my ads data using Python package aitextgen
6. Determination of most relevant metrics
   - categorical_accuracy and top_k_categorical_accuracy against baseline
7. Published to HuggingFace - 100+ downloads already!

Tokens and their numeric ids

```
['ZAGG', 'InvisibleShield', 'Luxe', 'Screen',
'Protector', 'for', 'Apple', 'Watch', 'Series', '2']
[3393, 963, 9601, 18828, 12556, 10430, 5129, 544,
9532, 9217]
```

Sequence: 3 input words and target

```
Sequence[0] : Zagg Invisible Luxe Screen
Sequence[1] : Screen Protector for Apple
[[ 3393, 963, 9601, 18828],
[18828, 12556, 10430, 5129]]
```

Self-supervised, target word

Model timing and tuning

- Deep Learning is a b**#@!
- Not just CPU, ran out of memory and disk
- Could not use 99.5% of the total data
- For 2M docs, 5 years to train on local machine and 7 months on G Colab!
- At 25k documents - OOM
- 13k rows, chars: 4,520,879, total tokens: 321,785, distinct tokens: 24101, num sequences: 96544, model trainable params: 15,514,789
- 30 min training time on Colab with GPU, 19 min with Embedding layer. 1hr 40m for fine-tuning
- Tuned multiple model params - batch_size, num input nodes, dimension of vectors in embedding, sequence length
- Fun Fact: Output shape is size of vocabulary (24k) - very different from the 2 in a binary classification or 10 in the MNIST image 0-9 digit classification!

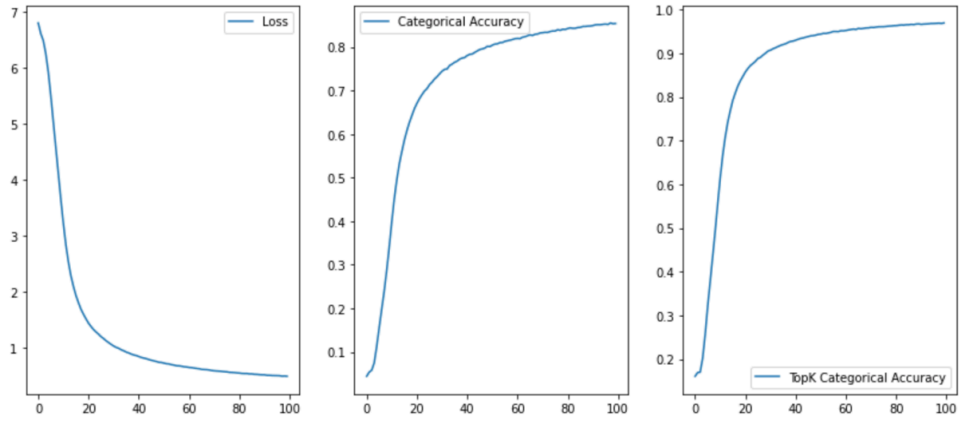
	Local Machine					Google Colab								
	LSTM				Embed	LSTM								Embed
# docs	**10k**	**100k**	**500K**	**2M**	**10K**	**10K**	**10K**	**10K**	**13k**	**15k**	**15k**	**25k**	**25k**	**13k**
Batch Size	4096	4096	4096	4096	4096	1024	2048	4096	**4096**	1024	2048	2048	4096	4096
Distinct tokens	25,561	115,438	300K			25211	25211		**22581**	33020	33020	46576	OOM	
Num sequences	52,014	516,681	26M							77419	77419	129584		
Trainable params	16M					16M				21M	21M	30M		
Steps per epoch	12					50			**23**	75	38	63		
Fit time for 1 epoch	2 min	150 mins	80 hours	800 hours	0.3 mins			0.2 min			Colab kicked me out		**OOM**	
For 60 epochs	80 mins	(150 hours)	**200 days!**	**5 years!**	15m	17m	15m	11m	**23m**	19m	25m	44m		10m
Evaluation Metrics														
categorical_ accuracy	**0.98**				0.92	0.89	0.98	**0.98**	**0.82**	0.78	0.947	0.80		0.81
categorical_ crossentropy	0.1				0.35	0.81	0.10	**0.091**	**0.69**	1.65	0.387	1.28		0.69
top_k_ categorical_ accuracy	0.98				0.96	0.9	0.98	**0.998**	**0.95**	0.83	0.991	0.87		0.95

Metrics - my model

Model	Baseline Categorical Crossentropy	Categorical Crossentropy (Loss)	Baseline Categorical Accuracy	Categorical Accuracy	Baseline Top K Categorical Accuracy	Top K Categorical Accuracy
LSTM	7.368	0.690	0.051	0.818	0.154	**0.952**
Embed LSTM	7.298	0.695	0.054	0.813	0.152	**0.953**

- Surprisingly, performed quite well!
- Baseline is defined as no training, just one epoch
- Metrics are much better than baseline
- Top K categorical accuracy of 0.95 is really good, though hard to tell if there is overfitting since there is no comparable validation data metrics (Keras doesn't easily support validation data with Sequences)
- Text generated is not too bad ☺

Evaluation of RNN LSTM model



Extending pre-trained models

Few shot learning	Fine Tuning
Model performs the task with only forward passes at test time	Training on a supervised dataset specific to the desired task Involves updating the weights of a pre-trained model
No gradient updates are performed	Trains the model by performing gradient updates after every epoch similar to the training of neural networks Similar to transfer learning, headless server

Zero-shot learning:

Task description:
Convert English to French

Prompt:
cheese =>

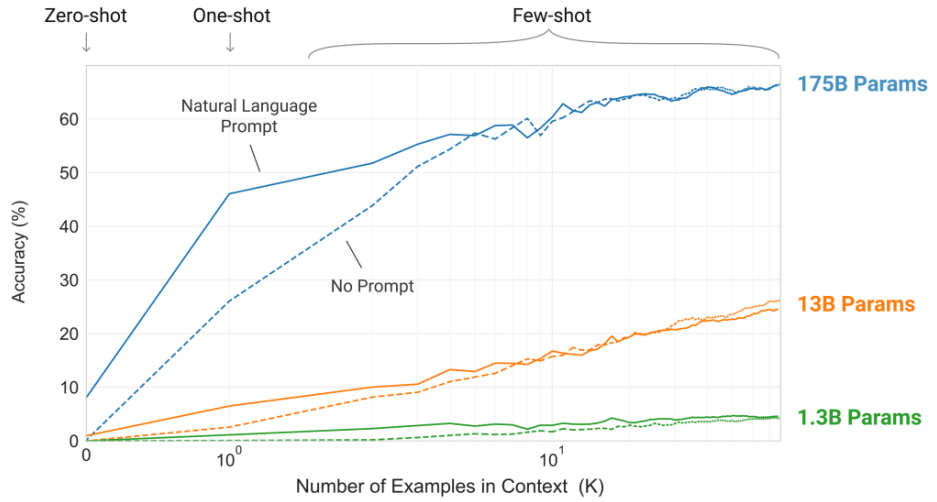
One-shot learning:
Task description:
Convert English to French

Example:
Sea-otter => loutre de maar
Prompt:
cheese =>

Few-shot learning:
Task description:
Convert English to French

Example:
Sea-otter => loutre de maar
Peppermint => menthe poivrée

Prompt:
cheese =>



Comparison with Causal Large Language Models

My model trained on 2MB of data. Had very low expectations

- Market leader: OpenAI's GPT-3 trained on 450 of data! Took $5M and 6 months to train with 100 GPU backends
- Reduced to 2 categories Electronics and Wearables (stopped using clothes, shoes etc). Quality of ad text generation improves significantly (not surprisingly)
- Both the gpt-2 and gpt-neo-1.3B pre-trained models don't do well with zero shot in generating ad text (not surprising)
- Customization helps: Greedy/sampling, temperature, top-k, top-p, repetition_penalty
- Fine tuning: can only use gpt-neo-125M, others (GPT2-355M, 774M, 1558M and GPT-NEO-355M and 1.3B) are crashing the Colab session or give OOM
- **My fine-tuned model does really well, though some hallucinations exist**

HuggingFace ('GitHub of Machine Learning') makes it super easy to use industry leading models

```
1. from transformers import pipeline
2. gpt_neo_generator = pipeline('text-generation', model='EleutherAI/gpt-neo-1.3B')
3. gpt_neo_generator("Best deals on Samsung")
```

How did we do - Inference

Actual ads from Amazon dataset	Sony DVPSR510H DVD Player, with HDMI port (Upscaling)
	Apple Watch Series 3 (GPS), 42mm Space Gray Aluminum Case with Black Sport Band - MQL12LL/A

Model	Generated output from model	Remarks
My model – from scratch	**Apple watch**, Charger Cable, Wireless Charging Pad, Compatible with Max, VESA,, 38mm, for Apple Watch Series 4 44mm, Full Coverage MilitaryShield Screen Protector **Sony TV** Universal Remote Control Replacement for Sony Bravia TV KDL- 26L5000 KDL- 32L5000 Zone KDL- 32S5100 H- 32XBR9	Some glitches. Locally coherent Mixes Charger Cable with Screen protector
My model Embedding	**Sony TV**, 4K HD LED TV Antenna Support 1080P, with HDMI, GANA 1080P HDMI to HDMI to HDMI Projector, AV Audio **Apple Watch** Band 38mm 40mm 42mm 44mm,, Fashion Bling Crystal Diamond Rhinestone Replacement Strap for Fitbit Charge	Watch band should be for Apple, not Fitbit charge
GPT-2	**Sony TV**, which has a $2.7 billion budget and is expected to become the first cable network in U.-S., with 10 million subscribers by 2016.[3]\n\n\n…and that's just for starters: **Apple Watch** 2.0, Apple\'s new watch OS for iOS and Android that will run on the latest version of Google Now or any other supported smartphone operating system until January 1st 2014	Coherent, pleasing sentence. Not related to ads 2014 data
gpt-neo-1.3B	'**Sony TV** will take "the world" by storm.\n\nIn my opinion, Sony is the greatest company to have invested in televisions and video technology. They are one of the most powerful producers and marketers of televisions because they have **Apple Watch**\n\nApple Watch is an intelligent wearable watch that is designed to be a replacement for the Apple iPhone and Apple Watch.	Not related to ads
My fine-tuned model	**Apple Watch** Band 38mm 40mm, Stainless Steel Metal Replacement Strap for iWatch Series 4/3/2/1 **Sony TV** Stick Remote for Roku Streaming Player Chromecast Ultra/Audio and Den LAN Network **Last minute deals Christmas deals on Apple Watch** Charging Stand for Any Smartphone	**Much better!!**

Now for some fun…
Your turn, for a live demo

https://huggingface.co/nirajsaran/AdTextGenerator

Conclusion

- The RNN LSTM model built from scratch works reasonably well even with only 13k documents
- The Large Language Models like GPT-2 and GPT_NEO etc do really well on the text they're trained on, with zero-shot learning
- They are not suitable for my use case - generating text for Amazon advertisements
- They do have "few-shot learning", where you can provide prompts in a line or two. Learns from this minimal training data to generate similar output content. I have not found it generate ad text, especially for Amazon ads
- Fine-tuning these pre-trained models with supervised dataset specific to the desired task works best

Next Steps

- **Ad text quality is really good with fine-tuned model. Good to deploy!**
- Expand to additional categories and also beyond Amazon Ads to Google, Facebook and LinkedIn ads
- Calculate Perplexity metrics
- Can improve performance by using TPU
  - Error during model.fit on Google Colab **with TPU**:
    - Failed to connect to all addresses.. This is a known blocking issue with TPUs - they don't support PyFunction. Details https://github.com/tensorflow/tensorflow/issues/38762, #34346, #39099
- Known Issues:
  - Don't have validation data to check for overfit. Running into error when using validation_data during fit
    - InvalidArgumentError: Graph execution error: Specified a list with shape [4096,22555] from a tensor with shape [4096,5570]

References

- Word-level text generation using GPT-2, LSTM and Markov Chain | Towards Data Science
- nlg-text-generation/lstm.ipynb at main · klaudia-nazarko/nlg-text-generation (github.com)
- aitextgen — Train a GPT-2 (or GPT Neo) Text-Generating Model w/ GPU - Colaboratory (google.com)
- https://huggingface.co/docs/hub/model-repos#when-sharing-a-model-what-should-i-add-to-my-model-card

Question & Answer