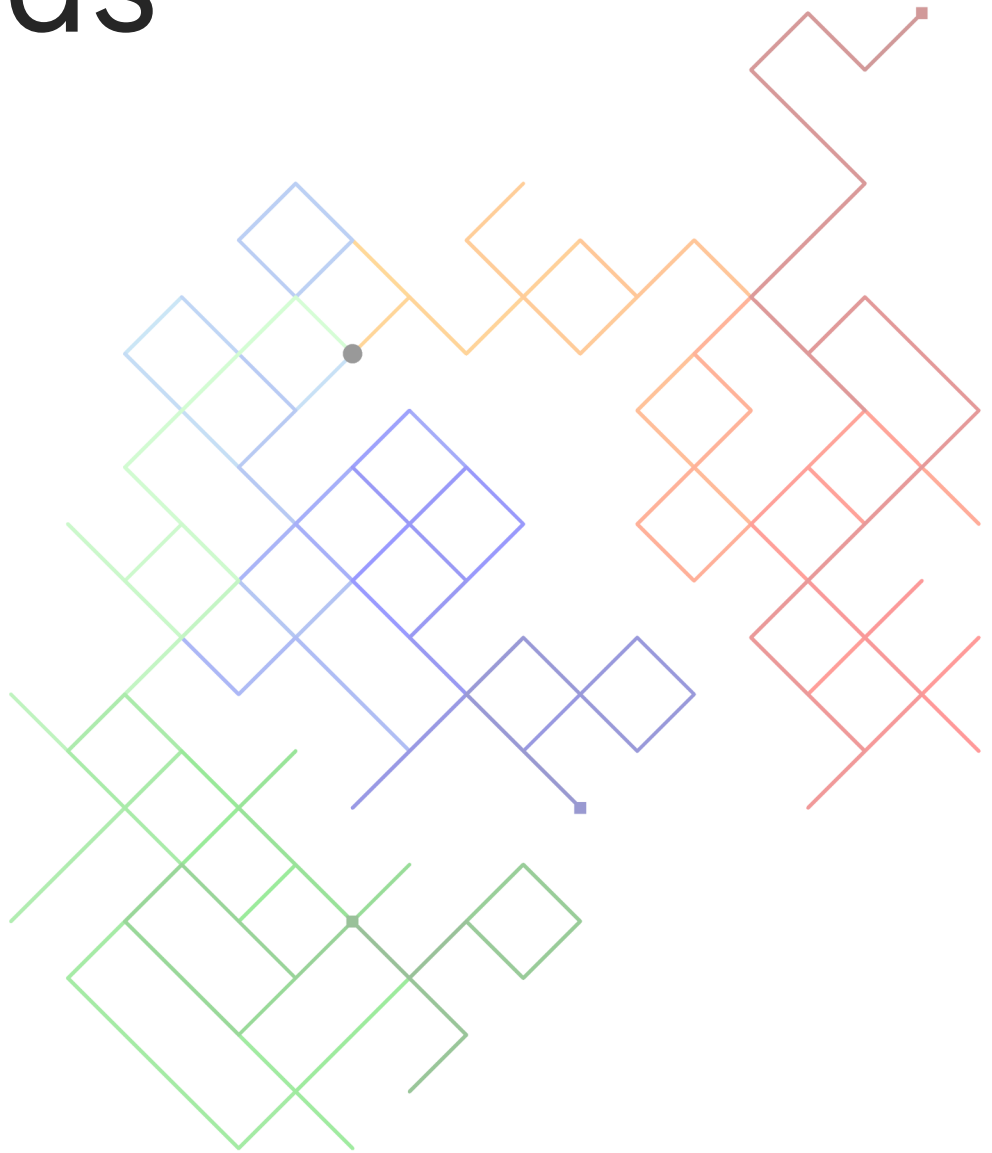
Big Models, Small Pitfalls: My random walk towards building sequence-to-sequence models

Shreya
Khurana



PyCon APAC
2022

This talk is about...

What I've learnt in my journey:

- Data Science @ GoDaddy
- While reading papers, articles, blogs etc.
- While building sequence-to-sequence models

Errors I've faced:

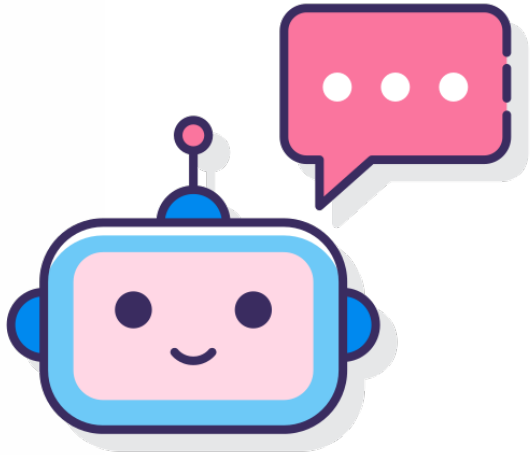
- They're always the stupid ones, aren't they?

Painful debugging:

- Hours and hours of wondering where I went wrong

Things I've regretted not doing

Sequence-to-sequence



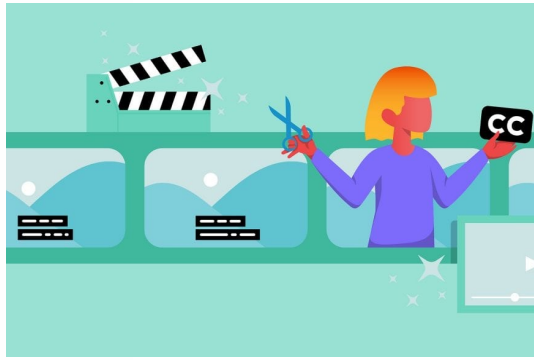
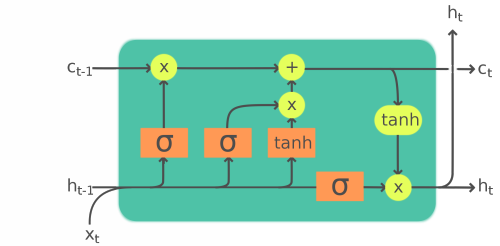


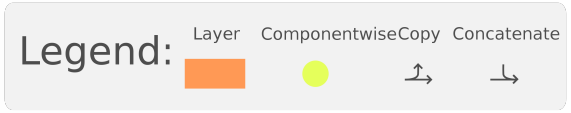
Image Sources: https://www.rev.com/blog/captions-increase-value-of-video-editing-services
https://www.researchgate.net/publication/345943755_Using_Sequence_to_Sequence_Model_to_Transform_Recognized_ASL_Words_to_Understandable_Sentences
https://www.shutterstock.com/search/summarization
https://www.seekpng.com/ima/u2w7u2a9i1r5r5y3/
https://www.flaticon.com/free-icon/chatbot_2040946
https://www.vectorstock.com/royalty-free-vector/young-people-talking-in-different-languages-vector-35052609
https://developer.nvidia.com/blog/how-to-build-domain-specific-automatic-speech-recognition-models-on-gpus/

Evolution

LSTM (Encoder-decoder)

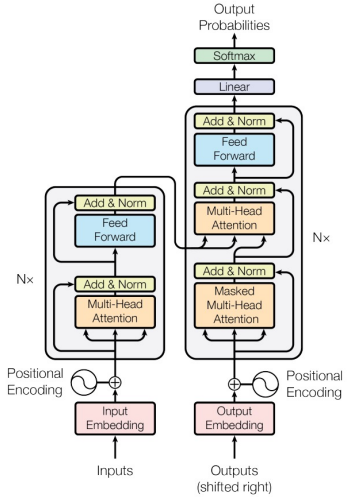
A woman is throwing a frisbee in a park.

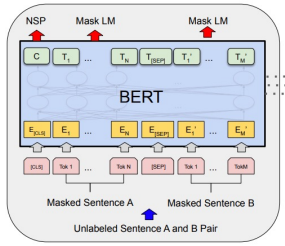




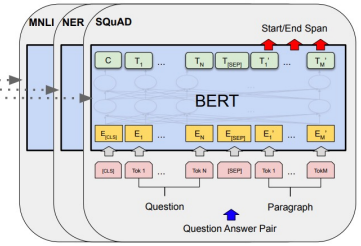
Attention

Transformer





Pre-training



Fine-Tuning

BERT (Bidirectional Transformer)

Pretrained Models

GPT-*n*,
T5,
x-BERT-*x*

https://ruder.io/a-review-of-the-recent-history-of-nlp/index.html#2014sequencetosequencemodels

By Guillaume Chevalier - File:The_LSTM_Cell.svg, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=109362147

Sutskever, I., Vinyals, O., & Le, Q. v. (2014). *Sequence to Sequence Learning with Neural Networks*.
Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*.
Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*.

What we know already…

Experiments with Vocabulary size

For a bigger training data, use a bigger vocabulary

Sizes used in research

*Ding et al (2019)

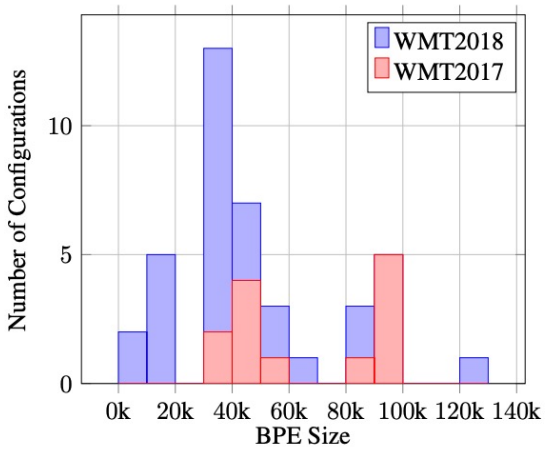


Figure 1: Histogram of BPE merge operations used for in WMT papers from 2017-2018.

LSTM-based architectures

*Denkowski and Neubig (2017)

	WMT			IWSLT	
	DE-EN	EN-FI	RO-EN	EN-FR	CS-EN
Words 50K	31.6	12.6	27.1	33.6	21.0
BPE 32K	**33.5**	**14.7**	**27.8**	34.5	22.6
BPE 16K	33.1	**14.7**	**27.8**	**34.8**	**23.0**

Table 2: BLEU scores for training NMT models with full word and byte pair encoded vocabularies. Full word models limit vocabulary size to 50K. All models are trained with annealing Adam and scores are averaged over 3 optimizer runs.

*Cherry et al. (2018)

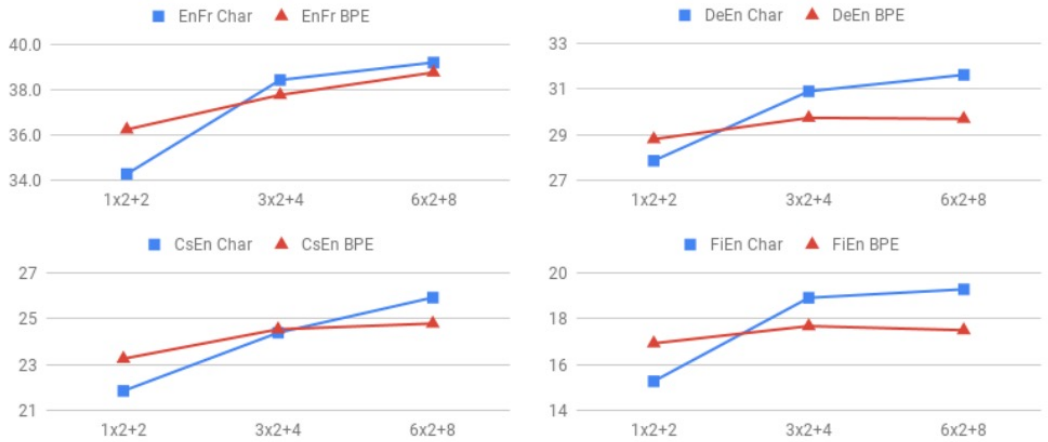


Figure 1: Test BLEU for character and BPE translation as architectures scale from 1 BiLSTM encoder layer and 2 LSTM decoder layers (1×2+2) to our standard 6×2+8. The y-axis spans 6 BLEU points for each language pair.

What we know already…

Experiments with Vocabulary size

*Ding et al (2019)

Transformer-based architectures

		0	0.5k	1k	2k	4k	8k	16k	32k	δ
deep-transformer	ar-en	30.3	30.8	30.6	30.5	30.4	29.8	28	27.5	3.3
	cs-en	24.6	23.3	23.0	22.7	21.2	22.6	20.6	21.0	4.0
	de-en	28.1	28.6	28.0	28.4	27.7	27.5	26.7	25.2	3.4
	fr-en	28.8	29.8	29.6	29.3	28.7	28.5	27.5	26.6	3.2
	en-ar	12.6	13.0	12.1	12.3	11.8	11.3	10.7	10.6	2.4
	en-cs	17.3	17.1	16.7	16.4	16.1	15.6	14.7	13.8	3.5
	en-de	26.1	27.4	27.4	26.1	26.3	26.1	25.8	23.9	3.5
	en-fr	25.2	25.6	25.3	25.5	25.3	24.7	24.1	22.8	2.8
shallow-transformer	ar-en	26.4	27.9	28.7	28.5	28.6	27.7	26.2	25.5	3.2
	cs-en	22.4	22.6	22.3	21.8	21.7	21.1	21.1	20.1	2.5
	de-en	25.5	27.4	27.1	27.3	27.1	25.9	24.6	23.7	3.7
	fr-en	26.3	28.0	28.9	28.0	28.0	27.4	26.1	26.1	2.7
	en-ar	11.7	11.2	11.5	11.0	11.3	10.5	9.5	9.0	2.7
	en-cs	16.4	16.7	16.0	16.2	14.4	14.2	13.9	13.9	2.8
	en-de	23.8	25.7	25.4	25.3	25.2	24.3	24.1	22.1	3.6
	en-fr	23.5	24.7	25.1	24.6	24.5	23.8	22.7	22.1	3.0

Table 2: BLEU score for Transformer architectures with multiple BPE configurations. Each score is color-coded by its rank among scores from different BPE configurations in the same row. δ is the difference between the best and worst BLEU score of each row.

Ding, S., Renduchintala, A., & Duh, K. (2019). *A Call for Prudent Choice of Subword Merge Operations in Neural Machine Translation*.

What we know already…

Experiments with training data

More the data, the better the performance

*Gu et al. (2018)

# of Sentences	0k	**6k**	**13k**	60k	600k
BLEU scores	0	**1.21**	**2.45**	12.49	28.34

Table 1: BLEU scores reported on the test set for Ro-En. The amount of training data effects the translation performance dramatically using a single NMT model.

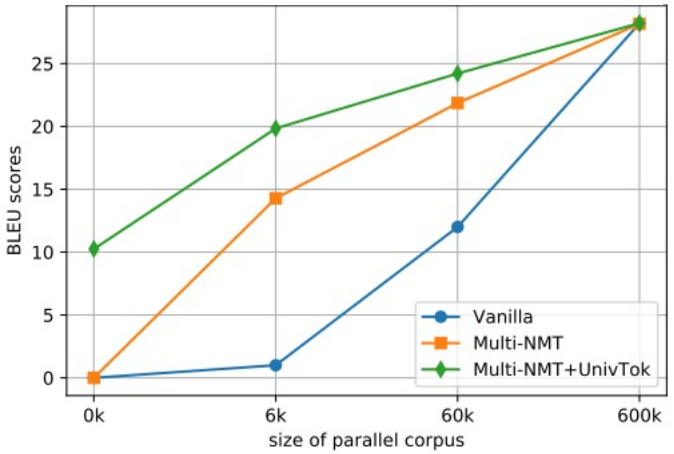


Figure 2: BLEU score vs corpus size

*Duh et al (2020)

	Data Size	Test1: Text SMT	Test1: Text NMT	Test2: Transcripts SMT	Test2: Transcripts NMT
Somali-English baseline	24k	15.1	14.4	7.8	7.7
+ paracrawl	104k	15.7	20.2	8.8	10.5
+ dictionary	50k	15.4	14.3	8.3	7.9
+ dictionary + found-bitext	273k	16.8	24.4	9.4	13.3
+ dictionary + found-bitext + paracrawl	354k	**17.3**	**25.0**	**9.5**	**13.6**
Swahili-English baseline	24k	24.4	24.8	15.4	13.4
+ paracrawl	85k	24.2	26.6	14.5	15.1
+ dictionary	123k	24.6	25.3	15.5	13.1
+ dictionary + found-bitext	312k	25.5	33.3	**16.2**	18.7
+ dictionary + found-bitext + paracrawl	373k	**25.6**	**33.7**	15.9	**20.6**

Table 3: The effect of additional resource types for SMT and NMT. We show BLEU scores on the text and transcripts test sets. Data Size shows the number of segments used for training. The NMT BLEU scores correspond to those "chosen" on the validation set, and is a fair comparison with the SMT numbers. The best BLEU score in each column is boldfaced. The baselines are trained on the MATERIAL training data, taken from Table 2. Observe that adding paracrawl, dictionary and found-bitext to baseline tends to improve performance for both SMT and NMT, with NMT gaining significant benefits.

*Cherry et al. (2018)

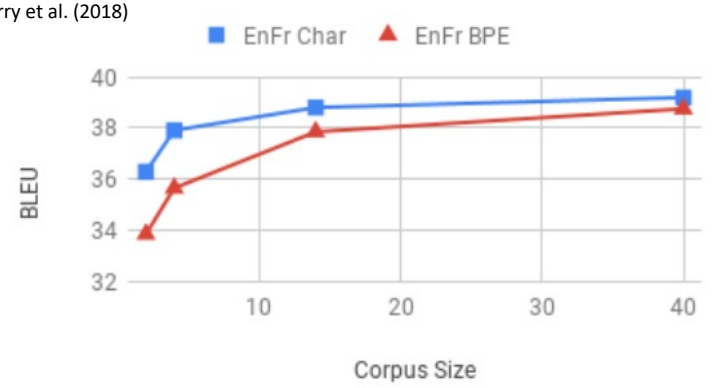


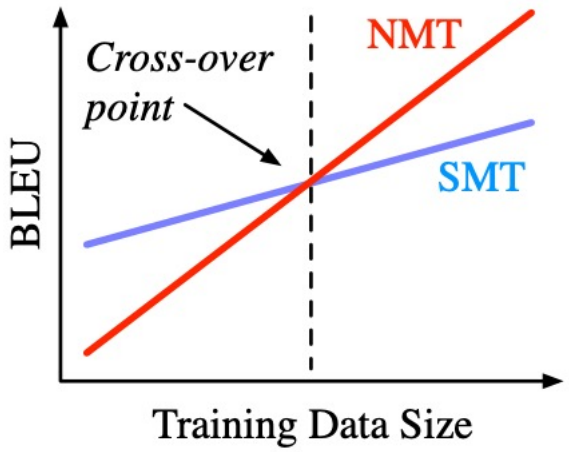
Figure 2: BLEU versus training corpus size in millions of sentence pairs, for the EnFr language-pair.

Let's go step by step: Data size

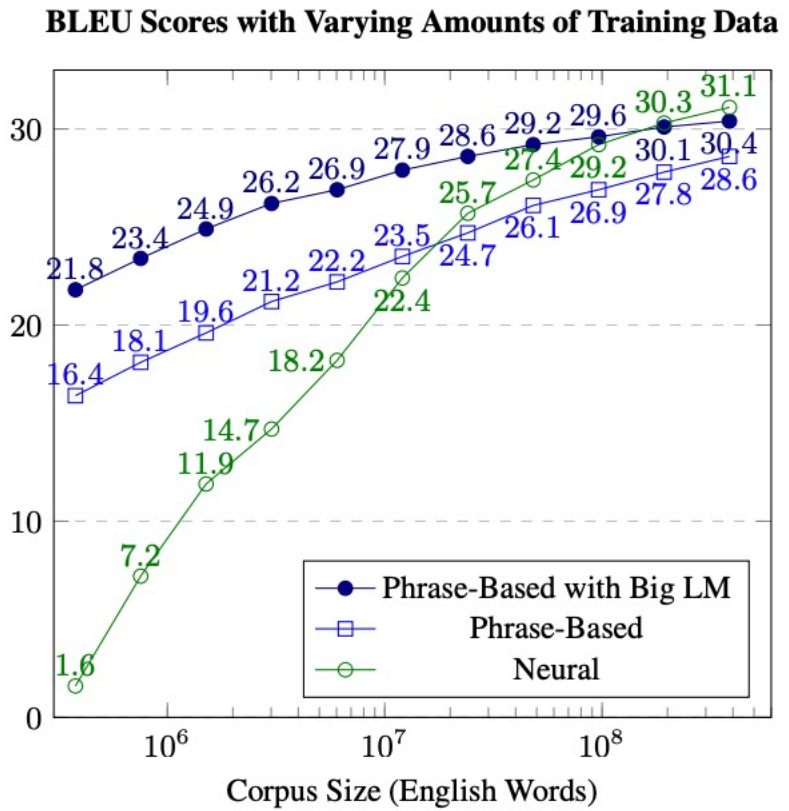
Not "enough" data

How much data is enough data?

*Duh et al (2020)



*Koehn and Knowles (2017)



Let’s go step by step: Data size

What can we do with small datasets?

Add more data!

Augment original data with

- Artificially-generated data
- Open-source datasets:
  - Google Dataset Search
  - Kaggle
  - Datasets from industry/academia research: Eg: No Language Left Behind (Meta)

Change the black box!

Use traditional machine learning algorithms

Pretrained models + transfer learning

- useful for high/low-resource mix distribution
- Example NMT: Universal representation for all languages for each word to help in zero/few shot learning [Gu et al., 2018]
- BERT, Universal Sentence Encoder, BART, T5

https://www.kaggle.com/datasets
https://datasetsearch.research.google.com/
https://github.com/facebookresearch/fairseq/blob/nllb/examples/nllb/data/README.md

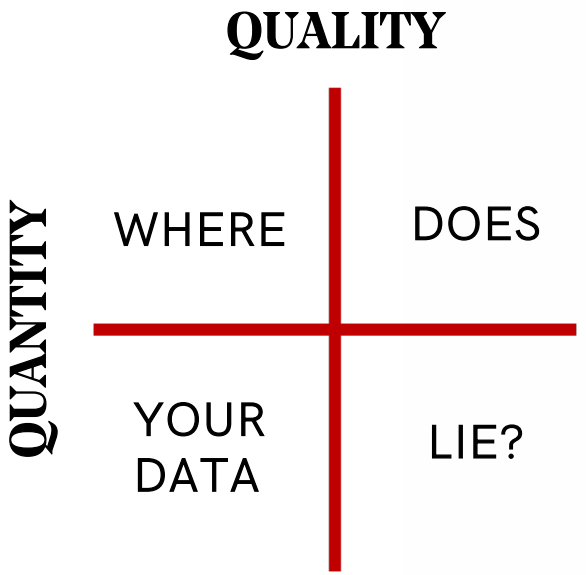
Let’s go step by step: Data Quality

When we have enough data, how much does quality matter?

Q. Is a bigger model affordable?

- Use model complexity to get to a better performance, without any information loss

Model size ⟷ Computation Resources
Inference time
Maintainability



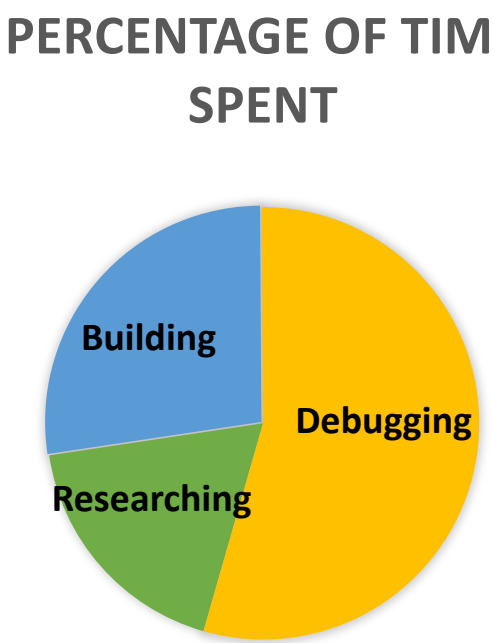
Q. Is majority data domain-specific?

- Get to as much domain-specific data as possible*
- Use pretrained models if needed

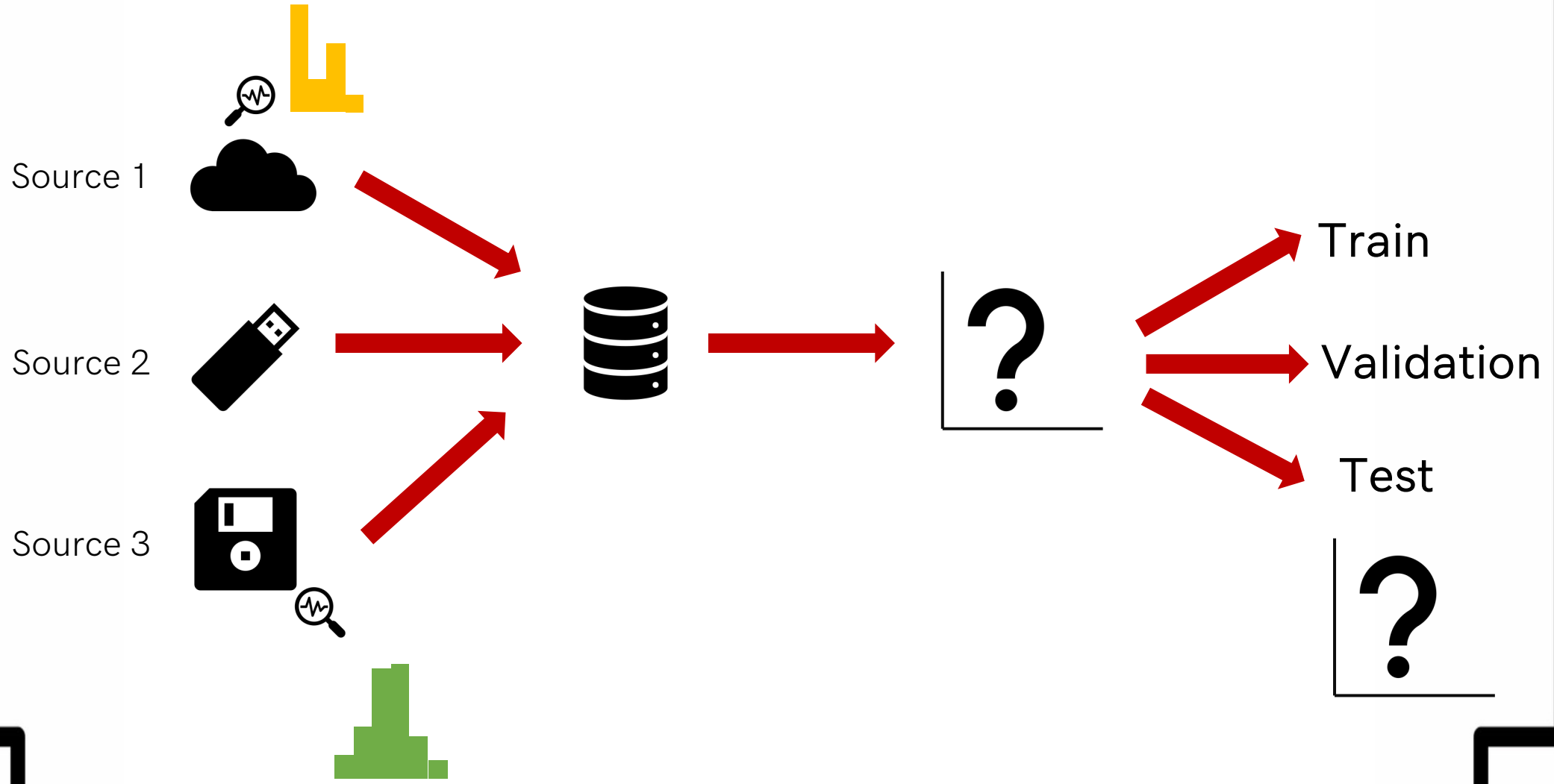
*https://arxiv.org/pdf/1706.03872.pdf

And now, the fun part!

Real-world data science in action



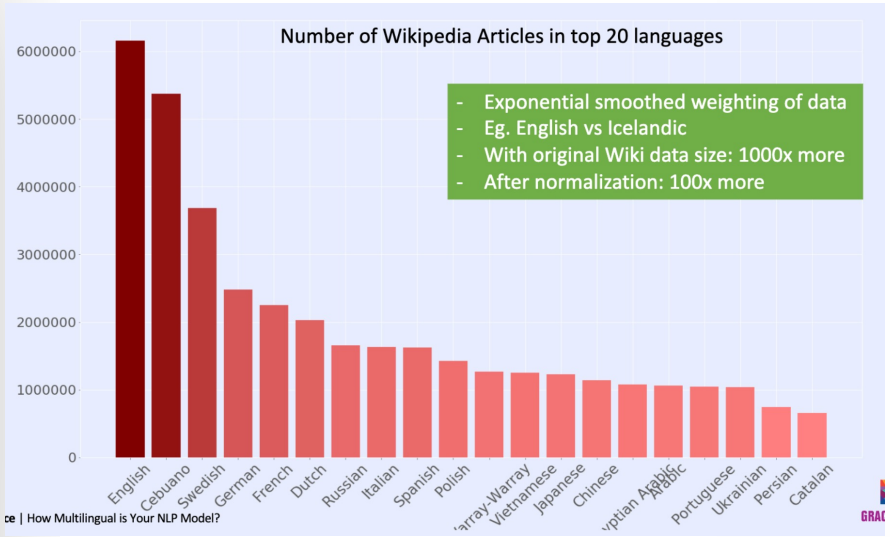
Data blunders: Varying distributions



Data blunders: Varying distributions

"Choose dev and test sets to reflect data you expect to get in the future and want to do well on." - Andrew Ng (Machine Learning Yearning)

```python
import pandas as pd
from pandas.plotting import scatter_matrix

data = pd.read_csv('...')

data.hist()

data.describe()

data.summary()

scatter_matrix(data, diagonal='kde')
```



https://machinelearningmastery.com/quick-and-dirty-data-analysis-with-pandas/

My talk at GHC 2020: https://github.com/ShreyaKhurana/ghc2020/blob/master/how-multilingual-is-your-nlp-model.slides.pdf

Data Blunders: Preprocessing components

- **Tedious bugs**
  - Have to go ALL the way back if not found early
- **Do obvious things**
  - Ensure same pipeline during training, testing and inference
  - Same cleaning modules, same tokenizer and vocabulary
  - Single source of truth for config
- **Test very obvious things**
  - Are you using the correct data?
  - Are your input/output being parsed correctly?
  - Did you generate a valid vocabulary?
  - Write unit tests
- **Code reviews are awesome! Have them ☺**

> "The world is full of obvious things which nobody by any chance ever observes."
>
> - Sherlock Holmes

Data Blunders: Preprocessing components

```python
import unittest
from config import SOURCE_LANG, TARGET_LANG, DEFAULT_BEAM, MODEL_NAME, MODEL_SUBDIR, USE_BPE, BPE_CODES, VOCAB
import tensorflow as tf


class TestObvious(unittest.TestCase):
      def test_model(self):

        model = load_model(MODEL_SUBDIR,
                           vocab=VOCAB,
                           use_bpe=USE_BPE,
                           bpe_codes=BPE_CODES
                           beam=DEFAULT_BEAM)

        # if using Keras
        self.assertIsInstance(model, tf.keras.Model)

        # if using a wrapper library/toolkit
        self.assertIsInstance(model, TransformerModel)

        # test correct files/tools
        self.assertIsNotNone(model.__dict__['bpe'])
        self.assertEqual(model.__dict__['cfg']['model'].bpe_codes, BPE_CODES)
        self.assertEqual(model.__dict__['cfg']['model'].bpe, 'subword_nmt')
        self.assertEqual(model.__dict__['cfg']['task']['data'], VOCAB)

        # test correct tokenizer
        self.assertIsInstance(model.__dict__['tokenizer'], TokenizerClass)

        # test hyperparameters
        self.assertEqual(model.__dict__['cfg']['model'].encoder_embed_dim, 32)
        self.assertEqual(model.__dict__['cfg']['model'].decoder_embed_dim, 32)
        self.assertEqual(model.__dict__['cfg']['model'].encoder_attention_heads, 2)
        self.assertEqual(model.__dict__['cfg']['model'].decoder_attention_heads, 2)

        self.assertEqual(model.__dict__['cfg']['model'].source_lang, SOURCE_LANG)
        self.assertEqual(model.__dict__['cfg']['model'].target_lang, TARGET_LANG)
        self.assertEqual(model.__dict__['cfg']['model'].beam, DEFAULT_BEAM)
```

https://docs.python.org/3/library/unittest.html

Data Blunders: Vocabulary generating tools

Idea of subword units: "...make the NMT model capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units." – Sennrich et al. (2015)

solar system (English) ←→ Sonnensystem (Sonne + System) (German)
solar system (English) ←→ Naprendszer (Nap + Rendszer) (Hungarian)

FastBPE
(C++ API to subword-nmt)

```
./fast
usage: fastbpe <command> <args>

The commands supported by fastBPE are:

getvocab input1 [input2]            extract the vocabulary from one or two text files
learnbpe nCodes input1 [input2]     learn BPE codes from one or two text files
applybpe output input codes [vocab] apply BPE codes to a text file
applybpe_stream codes [vocab]       apply BPE codes to stdin and outputs to stdout
```

Subword-nmt

To apply byte pair encoding to word segmentation, invoke these commands:

```
subword-nmt learn-bpe -s {num_operations} < {train_file} > {codes_file}
subword-nmt apply-bpe -c {codes_file} < {test_file} > {out_file}
```

https://github.com/glample/fastBPE
https://github.com/rsennrich/subword-nmt

Sennrich, R., Haddow, B., & Birch, A. (2015). *Neural Machine Translation of Rare Words with Subword Units.*

Data Blunders: Vocabulary generating tools

Concatenate training data for input and output sequences IF they share common alphabet

```
cat {train_file}.L1 {train_file}.L2 | subword-nmt learn-bpe -s {num_operations} -o {codes_file}
subword-nmt apply-bpe -c {codes_file} < {train_file}.L1 | subword-nmt get-vocab > {vocab_file}.L1
subword-nmt apply-bpe -c {codes_file} < {train_file}.L2 | subword-nmt get-vocab > {vocab_file}.L2

subword-nmt apply-bpe -c {codes_file} --vocabulary {vocab_file}.L1 --vocabulary-threshold 50 < {train_file}.L1 >
{train_file}.BPE.L1

subword-nmt apply-bpe -c {codes_file} --vocabulary {vocab_file}.L2 --vocabulary-threshold 50 < {train_file}.L2 >
{train_file}.BPE.L2
```

Some segmentations may be only L1/L2 specific
Use vocabulary threshold to only allow symbols present in that language vocabulary

https://github.com/rsennrich/subword-nmt
https://github.com/rsennrich/subword-nmt/issues/67

Data Blunders: Vocabulary generating tools

Perform sanity checks on vocab files

```
NUM_ALLOWED_CHARS = 26 + 1 # eg. alphabet characters + space
BPE_OPERATIONS = 32000

with open('vocab_file.txt') as f:
    vocab = f.readlines()

vocab_set = set([v.lower().strip() for v in vocab])

for word in most_common_words:
    assert word in vocab_set

assert len(vocab_set) < 2*NUM_ALLOWED_CHARS + BPE_OPERATIONS
assert len(vocab_set) > ...
```

Making Practical Modeling Choices

1) Neural net training is a leaky abstraction

2) Neural net training fails silently

Andrej Karpathy, A Recipe for Training Neural Networks. Apr 25, 2019.

Writing Your First Neural Net in Less Than 30 Lines of Code with Keras

How to build a simple neural network in 9 lines of Python code

Deep Learning in 7 lines of code

How to build a Deep Learning model in 10 lines

A Guide to quickly immerse yourself in Deep Learning

Source: Various Blog Titles

http://karpathy.github.io/2019/04/25/recipe/

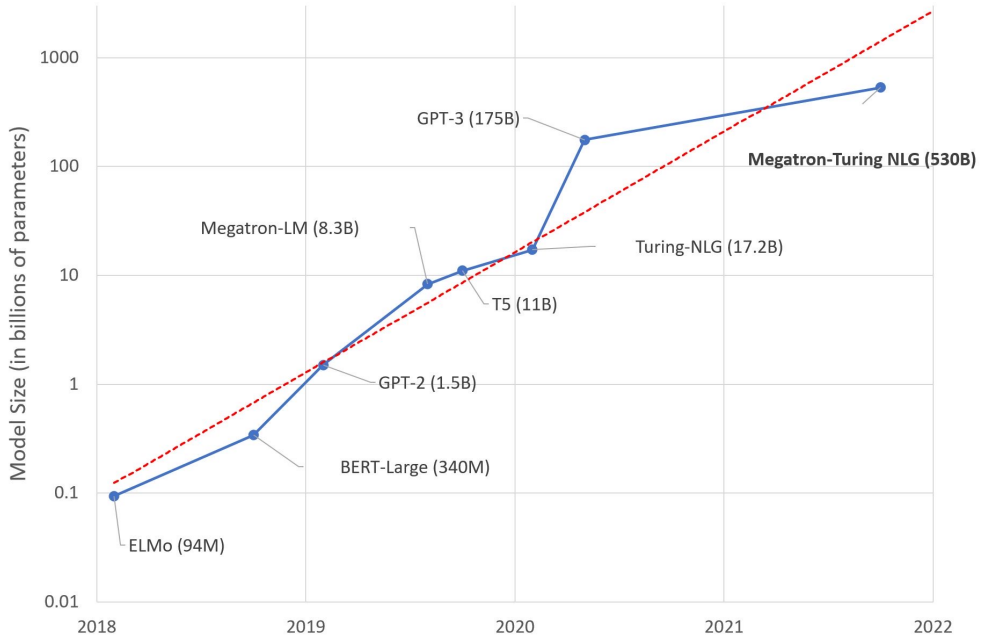
Initial Model Architecture

"First get a model large enough that it can overfit (i.e. focus on training loss) and then regularize it appropriately (give up some training loss to improve the validation loss)"

"Find the most related paper and copy paste their simplest architecture that achieves good performance"

- **Andrej Karpathy, A Recipe for Training Neural Networks. Apr 25, 2019.**

- **Research smaller versions like DistilBERT (Sanh et al., 2019), DistilBART (Shleifer and Rush, 2020)**



Source: https://huggingface.co/blog/large-language-models

http://karpathy.github.io/2019/04/25/recipe/

Iterating on the Model Architecture

Make one change at a time

- Don't mix data and model changes (if you can)
- Add one feature at a time
- Change data timeframe by leaving out/adding smaller timeframes

Define Optimizing and Satisficing metrics

- Acceptable running time
- Refer papers, third-party libraries (Huggingface), hardware benchmarks (NVIDIA/Google Cloud/AWS) for baselines
- just be "good enough" on this metric, in the sense that it should take at most 100ms.
- Then optimize the major metric like BLEU/perplexity/validation loss

[Andrew Ng. (2018). *Machine Learning Yearning*.]

https://github.com/ajaymache/machine-learning-yearning

Model Description	Threads	QPS	Average Latency (p50)	P90 Latency	P95 Latency	P99 Latency
ONNX converted + dynamic-quantized PyTorch	8	54.06	18.5	18.72	18.86	19.31
ONNX converted + dynamic-quantized PyTorch	4	35.81	27.92	28.48	28.67	28.76
Dynamic-quantized TorchScript	8	32.60	30.67	36.01	36.94	38.02
ONNX converted PyTorch	8	31.29	31.96	33.05	33.44	33.82
Dynamic-quantized TorchScript	4	30.12	33.21	34.64	35.32	36.13
Vanilla TensorFlow	4	10.58	94.56	98.00	99.46	101.39
Vanilla TensorFlow	8	8.34	119.87	130.93	135.58	138.63
Vanilla TensorFlow	2	7.55	132.52	136.78	137.35	138.65

Source: https://blog.twitter.com/engineering/en_us/topics/insights/2021/speeding-up-transformer-cpu-inference-in-google-cloud

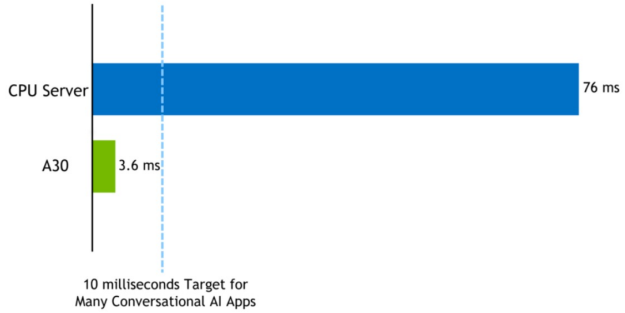


Figure 4. Compute latency in milliseconds for executing BERT-large on an NVIDIA A30 GPU vs. a CPU-only server

Source: https://developer.nvidia.com/blog/real-time-nlp-with-bert-using-tensorrt-updated/

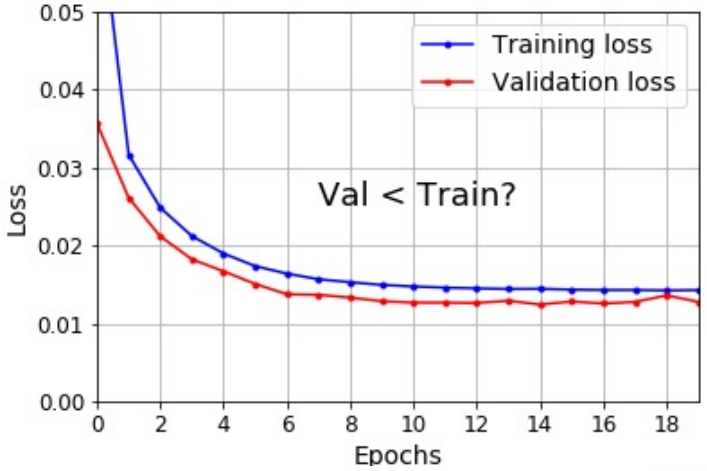
Training and Validation Loss

In general, prediction on training set is easier than prediction on the unseen validation set.

Validation loss > Training loss

BUT...

- **Data leak from training set to validation set**
- **Validation distribution different than training distribution. May be easier to predict on.**
- **Regularization (Dropout during training and is not validation/testing)**



Source: Aurélien Geron.
https://twitter.com/aureliengeron/status/1110839223878184960.

→

Build a strong validation set

Check sampling distribution

Add more data to the validation set

Regularization is okay!

Source Twitter thread:
https://twitter.com/svpino/status/1423569964112429060?s=09

Thank you!

Research results

Easy ways to fall in practical scenarios

How to avoid them

Acknowledgements:

Team @ GoDaddy

Team @ PyCon APAC