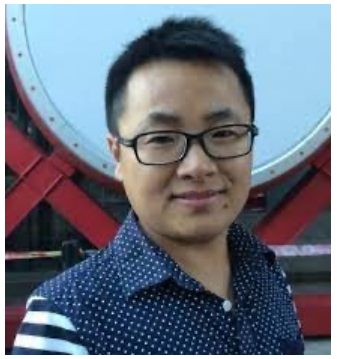
Ken

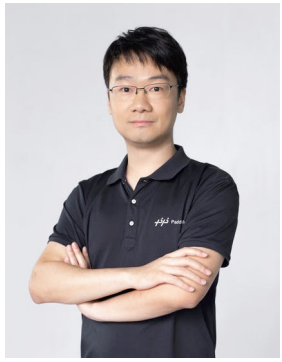
Valia

Gary

Ernie



Yanjun



Zeyu

ACL-2022 Tutorial:
Part A: Simple Uses of Deep Nets

Ken Church, Baidu, USA

Valia Kordoni, Humboldt-Universitaet zu Berlin, Germany

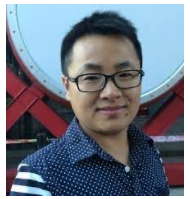
Gary Marcus, Robust.AI

Ernest Davis, NYU

Yanjun MA, Baidu, China

Zeyu Chen, Baidu, China

Part A: Simple Uses of Deep Nets



Yanjun

Zeyu

Schedule

- Overview (15 min)
- Part A: (60 min)
  - Glass is half full
- Break/Q&A (10 min)
- Part B: (60 min)
  - Glass is half empty
  - There is always more work to do
- Conclusions (15 min)
- Break/Q&A (10 min)

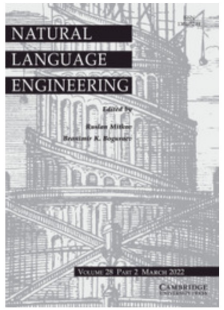
Valia

Gary

Ernie

Part A: Glass is Half Full

Simple/Popular Methods are Powerful



Natural Language Engineering

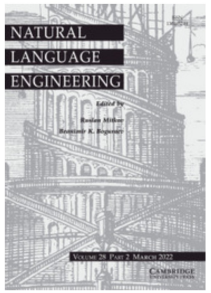
Emerging trends: Deep nets for poets

Published online by Cambridge University Press: **01 September 2021**

Kenneth Ward Church, Xiaopeng Yuan, Sheng Guo, Zewu Wu, Yehua Yang and Zeyu Chen

Article | Figures | Metrics

Save PDF | Share | Cite | Rights & Permissions



Natural Language Engineering

Emerging trends: A gentle introduction to fine-tuning

Published online by Cambridge University Press: **26 October 2021**

Kenneth Ward Church, Zeyu Chen and Yanjun Ma — Show author details

Article | Figures | Metrics

Save PDF | Share | Cite | Rights & Permissions

Standard 3-Step Recipe

Step	gft Support	Description	Time	Hardware
1		Pre-Training	Days/Weeks	Large GPU Cluster
2	*gft_fit*	Fine-Tuning	Hours/Days	1+ GPUs
3	*gft_predict*	Inference	Seconds/Minutes	0+ GPUs

Terminology

Standard Terminology	Proposed Alternatives
~~Base/Foundation/Pre-Trained~~	f_{pre}
~~Fine-Tuning~~	*fit*
~~Inference~~	*predict*

De-emphasize this

Emphasize this

~~Base/Foundation/Pre-Trained Models~~

Recommendation: Download f_{pre} from Hubs

Some Popular Pre-Trained Models: f_{pre}

Pre-trained (foundation) model	Parameters	Training data
ResNet-50 (He *et al.* 2016)	23M	14M images from ImageNet
VT (Wu *et al.* 2020)	11.7–21.9M	14M images from ImageNet
Wav2vec (Baevski *et al.* 2020)	95–317M	960 hours from LibriSpeech
BERT (Devlin *et al.* 2019)	110–340M	3.3B words from Books/Wiki
ERNIE 2.0 (Sun *et al.* 2020)	110–340M	7.9B en + 15B zh tokens
ERNIE 3.0 (Sun *et al.* 2021)	10B	375B tokens of text, as well as knowledge graph
RoBERTa (Liu *et al.* 2019)	110M	160GBs of text
GPT-2 (Radford *et al.* 2019)	1.5B	40GBs of text
GPT-3 (Brown *et al.* 2020)	125M–175B	1TB from Common Crawl, Books and Wikipedia
XLM-RoBERTa (Conneau *et al.* 2020)	12–16B	2.5TBs from many languages

Some Popular Datasets for Pre-Training

Dataset	Description
Billion Word Benchmark (Chelba *et al.* 2013)	A billion words of English
Common Crawl (Buck, Heafield, and van Ooyen 2014)	https://github.com/commoncrawl
Book Corpus (Zhu *et al.* 2015)	Speech with text
ImageNet (Deng *et al.* 2009)	14M images, annotated with 21k classes
LibriSpeech (Panayotov *et al.* 2015)	960 hours of speech with text
LJ Speech	https://keithito.com/LJ-Speech-Dataset/
AISHELL-2 (Du *et al.* 2018)	https://protect-eu.mimecast.com/s/_t4dC0Vqq sjz5YGswPwKH?domain=aishelltech.com

DDI → Don't Do It (yourself)

Terminology

Standard Terminology	Proposed Alternatives
~~Base/Foundation/Pre-Trained~~	f_{pre}
~~Fine-Tuning~~	*fit*
~~Inference~~	*predict*

Emphasize this

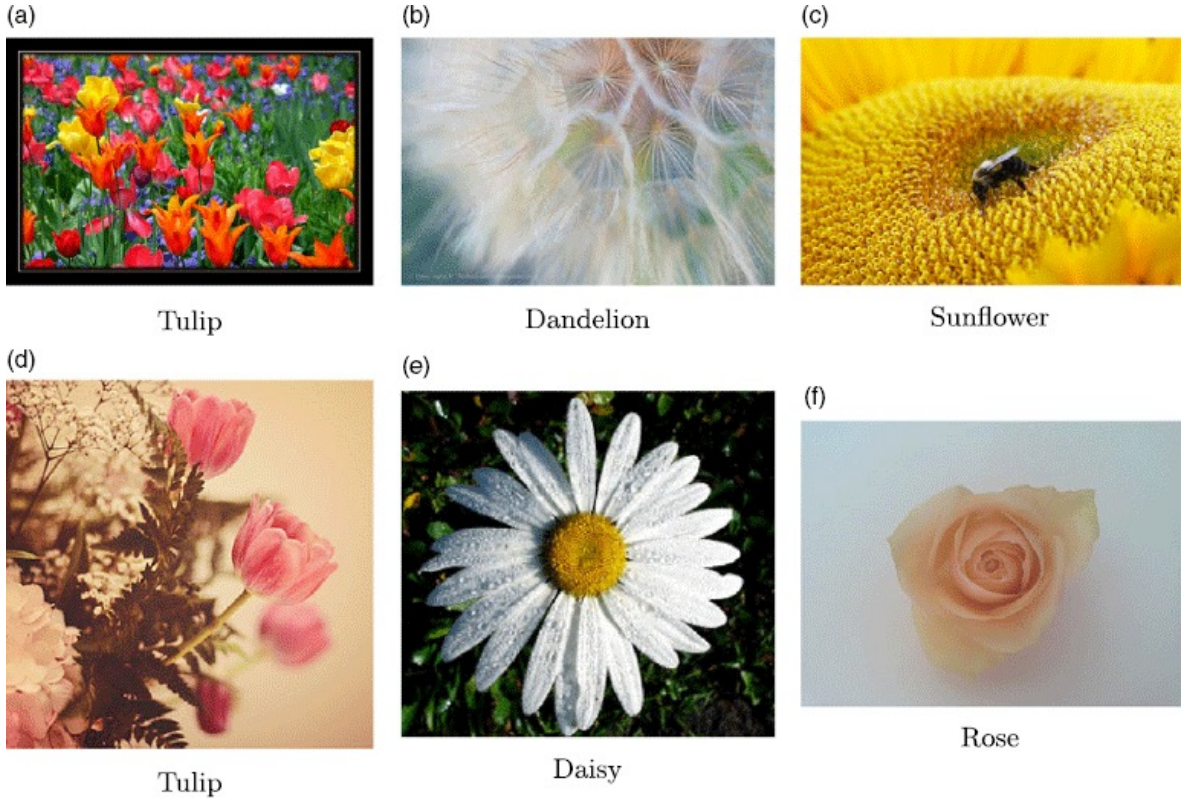
De-emphasize this

We will discuss why we want to change the terminology later

What is ~~fine-tuning~~ *fit*?

- *fit*: $f_{pre} + data \rightarrow f_{post}$
- f_{pre}: ResNet-50
  - Trained on ImageNet
  - Maps images to ImageNet Labels
- $data$: Flowers
  - Maps images to Flower Labels
  - Flower Labels ≠ ImageNet Labels
- f_{post}:
  - maps images to Flower Labels

$data$ → Flowers



f_{pre}: Pre-trained Model

Pre-trained (foundation) model	Parameters	Training data
ResNet-50 (He *et al.* 2016)	23M	14M images from ImageNet
VT (Wu *et al.* 2020)	11.7–21.9M	14M images from ImageNet
Wav2vec (Baevski *et al.* 2020)	95–317M	960 hours from LibriSpeech
BERT (Devlin *et al.* 2019)	110–340M	3.3B words from Books/Wiki
ERNIE 2.0 (Sun *et al.* 2020)	110–340M	7.9B en + 15B zh tokens
ERNIE 3.0 (Sun *et al.* 2021)	10B	375B tokens of text, as well as knowledge graph

Dataset	Description
Billion Word Benchmark (Chelba *et al.* 2013)	A billion words of English
Common Crawl (Buck, Heafield, and van Ooyen 2014)	https://github.com/commoncrawl
Book Corpus (Zhu *et al.* 2015)	Speech with text
ImageNet (Deng *et al.* 2009	14M images, annotated with 21k classes
LibriSpeech (Panayotov *et al.* 2015)	960 hours of speech with text
LJ Speech	https://keithito.com/LJ-Speech-Dataset/

Classify Spans:
Answer is a substring of context

Task: ~~Question-Answering (QA)~~

What is ~~fine-tuning~~ *fit*?

- *fit*: $f_{pre} + data \rightarrow f_{post}$
- f_{pre}: BERT
  - Trained on Text Corpora
- $data$: SQuAD
  - Maps questions + contexts to answers
- f_{post}:
  - Maps questions + contexts to answers

SQuAD

Text with no markup

- Question:
  - *What does AFC stand for*?
- Context:
  - *The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title.*
- Answer:
  - *American Football Conference*

f_{pre}: Pre-trained Model

Pre-trained (foundation) model	rameters	Training data
ResNet-50 (He *et al.* 2016)	23M	14M images from ImageNet
VT (Wu *et al.* 2020)	11.7–21.9M	14M images from ImageNet
Wav2vec (Baevski *et al.* 2020)	95–317M	960 hours from LibriSpeech
BERT (Devlin *et al.* 2019)	110–340M	3.3B words from Books/Wiki
ERNIE 2.0 (Sun *et al.* 2020)	110–340M	7.9B en + 15B zh tokens
ERNIE 3.0 (Sun *et al.* 2021)	10B	375B tokens of text, as well as knowledge graph

Dataset	Description
Billion Word Benchmark (Chelba *et al.* 2013)	A billion words of English
Common Crawl (Buck, Heafield, and van Ooyen 2014)	https://github.com/commoncrawl
Book Corpus (Zhu *et al.* 2015)	Speech with text
ImageNet (Deng *et al.* 2009)	14M images, annotated with 21k classes
LibriSpeech (Panayotov *et al.* 2015)	960 hours of speech with text
LJ Speech	https://keithito.com/LJ-Speech-Dataset/

What is ~~fine-tuning~~ *fit*?

- *fit*: $f_{pre} + data \rightarrow f_{post}$
- f_{pre}:
  - bert-base-cased model
  - from HuggingFace
- $data$:
  - Emotion dataset
  - from HuggingFace
- f_{post}:
  - Maps text to emotion labels

gft Program

```
gft_fit --eqn 'classify: label ~ text' \
    --model H:bert-base-cased \
    --data H:emotion \
    --output_dir $outdir
```

Pre-trained (foundation) model	Parameters	Training data
ResNet-50 (He *et al.* 2016)	23M	14M images from ImageNet
VT (Wu *et al.* 2020)	11.7–21.9M	14M images from ImageNet
Wav2vec (Baevski *et al.* 2020)	95–317M	960 hours from LibriSpeech
BERT (Devlin *et al.* 2019)	110–340M	3.3B words from Books/Wiki
ERNIE 2.0 (Sun *et al.* 2020)	110–340M	7.9B en + 15B zh tokens
ERNIE 3.0 (Sun *et al.* 2021)	10B	375B tokens of text, as well as knowledge graph

Dataset	Description
Billion Word Benchmark (Chelba *et al.* 2013)	A billion words of English
Common Crawl (Buck, Heafield, and van Ooyen 2014)	https://github.com/commoncrawl
Book Corpus (Zhu *et al.* 2015)	Speech with text
ImageNet (Deng *et al.* 2009)	14M images, annotated with 21k classes
LibriSpeech (Panayotov *et al.* 2015)	960 hours of speech with text
LJ Speech	https://keithito.com/LJ-Speech-Dataset/

What is ~~fine-tuning~~ *fit*?

- *fit*: $f_{pre} + data \rightarrow f_{post}$
- f_{pre}:
  - bert-base-cased model
  - from HuggingFace
- $data$:
  - Emotion dataset
  - from HuggingFace
- f_{post}:
  - Maps text to emotion labels

gft Programs

H → HuggingFace

```
gft_fit --model H:bert-base-cased \
    --data H:emotion \
    --output_dir $1 \
    --eqn 'classify: label ~ text'
```

P → PaddleNLP

```
gft_fit --model P:ernie-tiny \
    --data P:chnsenticorp \
    --output_dir $1 \
    --eqn 'classify: label ~ text'
```

Pre-trained (foundation) model	Parameters	Training data
ResNet-50 (He *et al.* 2016)	23M	14M images from ImageNet
VT (Wu *et al.* 2020)	11.7–21.9M	14M images from ImageNet
Wav2vec (Baevski *et al.* 2020)	95–317M	960 hours from LibriSpeech
BERT (Devlin *et al.* 2019)	110–340M	3.3B words from Books/Wiki
ERNIE 2.0 (Sun *et al.* 2020)	110–340M	7.9B en + 15B zh tokens
ERNIE 3.0 (Sun *et al.* 2021)	10B	375B tokens of text, as well as knowledge graph

Dataset	Description
Billion Word Benchmark (Chelba *et al.* 2013)	A billion words of English
Common Crawl (Buck, Heafield, and van Ooyen 2014)	https://github.com/commoncrawl
Book Corpus (Zhu *et al.* 2015)	Speech with text
ImageNet (Deng *et al.* 2009)	14M images, annotated with 21k classes
LibriSpeech (Panayotov *et al.* 2015)	960 hours of speech with text
LJ Speech	https://keithito.com/LJ-Speech-Dataset/

gft (general fine-tuning):
A Little Language for Deep Nets
(Unix Philosophy: *Less is More*)

Standard 3-Step Recipe

Step	gft Support	Description	Time	Hardware
1		Pre-Training	Days/Weeks	Large GPU Cluster
2	*gft_fit*	Fine-Tuning	Hours/Days	1+ GPUs
3	*gft_predict*	Inference	Seconds/Minutes	0+ GPUs

- Terminology borrowed from *sklearn*:
  - *fit*: $f_{pre} + data \rightarrow f_{post}$
  - *predict*: $f(x) \rightarrow \hat{y}$
- *fit* and *predict* are (almost) all you need
  - *gft* programs are short (1-line)
  - No (not much) programming required
    - No python in this tutorial
    - Examples on hubs are (unnecessarily) long/complicated

Examples of 1-line GFT Programs

Step 2: *gft_fit*

```
gft_fit --eqn 'classify: label ~ text' \
    --model H:bert-base-cased \
    --data H:emotion \
    --output_dir $outdir
```

Data

f_{pre}: Pre-trained Model

f_{post}: Post-trained Model

Step 3: *gft_predict*

```
# text-classification: sentiment analysis
echo 'I love you.' | gft_predict --task text-classification
# I love you.   POSITIVE    0.9998705387115479
```

x $\hat{y}$ Score

Goals (Work in Progress)

- 4+1 functions
  1. *gft_fit* : $f_{pre} + data \rightarrow f_{post}$
     - Fit (almost) any model, f_{pre}
     - with (almost) any (reasonable) dataset, $data$
  2. *gft_ predict*: $f(x) \rightarrow \hat{y}$
     - Apply (almost) any model, f, to almost any input, x
  3. *gft_ summary*:
     - Summarize (almost) anything
     - Find good stuff
  4. *gft_ eval*:
     - Score (almost) any model, f, on (almost) any dataset
  5. *gft_cat_data*:
     - Output (almost) any dataset on *stdout*
- 4 arguments (to 4+1 functions)
  - Datasets, Models, Tasks, Equations
  - --data, --model, --task, --eqn

- Ease of Use
  - Simplicity
  - Consistent Terminology
    - Small set of functions/args cover most cases
  - Generality: Avoid special cases
    - Dataset/Model/Task/Equations
  - Interoperability
    - (Almost) everything works
      - on (almost) any (reasonable) dataset/model/task/equation
      - with (almost) no changes
    - Supplier Agnostic:
      - Everything works everywhere
      - (as much as possible): H/P/C
        - H:glue,cola → Use cola subset of glue from HuggingFace
        - P:glue,cola → Same as above, but from PaddleNLP
        - C:$gft/datasets/syn_ant/tag-$pos-pairs → local filesystem

Portability: Mix and Match Across Suppliers

All HuggingFace (H)

```
gft_fit --pretrained H:bert-base-cased \
    --data H:glue,qnli \
    --metric H:glue,qnli \
    --output_dir $1 \
    --eqn 'classify: label ~ question + sentence' \
    --num_train_epochs 3
```

Model from P; Data from H

```
gft_fit --pretrained P:bert-base-cased \
    --data H:glue,qnli \
    --metric H:glue,qnli \
    --output_dir $1 \
    --eqn 'classify: label ~ question + sentence' \
    --num_train_epochs 3
```

Reality Check

- *gft* takes dependencies on many python packages
  - Lots of moving targets
- Currently not possible to support everything
  - Under a single virtual environment
    - Best attempt to support as much functionality as possible
      - pip install –r $gft/requirements/requirements.txt
    - Special cases
      - pip install –r $gft/requirements/requirements_with_adapters.txt
      - pip install –r $gft/requirements/requirements_without_adapters.txt
      - pip install –r $gft/requirements/requirements_for_paddlespeech.txt

gft Cheat Sheet

Step	gft Support	Description	Time	Hardware
1		Pre-Training	Days/Weeks	Large GPU Cluster
2	*gft_fit*	Fine-Tuning	Hours/Days	1+ GPUs
3	*gft_predict*	Inference	Seconds/Minutes	0+ GPUs

4+1 Functions

1. gft_fit: $f_{pre} \rightarrow f_{post}$ (fine-tuning)
   - 4 Arguments, --output_dir, --metric, --splits
   - (plus most args in most hubs)
2. gft_predict: $f(x) \rightarrow \hat{y}$ (inference)
   - Input: 4 Arguments (x from data or stdin)
   - Output: $\hat{y}$ for each x
3. gft_eval: Score model on dataset
   - Input: 4 Arguments, --split, --metric, …
   - Output: Score
4. gft_summary: Find good stuff
   - Input: 4 Arguments
   - (may include: __*contains*__, __*infer*__)
5. gft_cat_dataset: Output data to *stdout*
   - Input: 4 Arguments (--data, --eqn)

4 Arguments

- --data
- --model
- --eqn $task: y \sim x_1 + x_2$
- --task
   1. classify (text-classification)
   2. classify_tokens (token-classification)
   3. classify_spans (QA, question-answering)
   4. classify_audio (audio-classification)
   5. classify_images (image-classification)
   6. regress
   7. text-generation
   8. MT (translation)
   9. ASR (ctc, automatic-speech-recognition)
   10. fill-mask

Seven Simple Examples

1. Summary
2. Summary with patterns (__*contains*__)
3. Predict: Sentiment Analysis & Text Classification
4. More tasks: token_classification, fill-mask
5. Input from datasets (as opposed to stdin)
6. *gft_predict* → *gft_eval*
7. *gft_fit* (aka fine-tuning)

Seven Simple Examples: 1 of 7
Summarize (almost) anything

```
emodel=H:bhadresh-savani/roberta-base-emotion

# Summarize a dataset and/or model
gft_summary --data H:emotion
gft_summary --model $emodel
gft_summary --data H:emotion --model $emodel
```

Emotion Classes

Summarize both data and model

Seven Simple Examples: 2 of 7

Summary with patterns (__*contains*__)

```
emodel=H:bhadresh-savani/roberta-base-emotion

# Summarize a dataset and/or model
gft_summary --data H:emotion
gft_summary --model $emodel
gft_summary --data H:emotion --model $emodel

# find some popular datasets and models that contain "emotion"
gft_summary --data H:__contains__emotion --topn 5
gft_summary --model H:__contains__emotion --topn 5
```

Find datasets on HuggingFace that contain “emotion”

Find *models* on HuggingFace that contain “emotion”

Seven Simple Examples: 3 of 7

Predict: Sentiment Analysis & Text Classification

```
emodel=H:bhadresh-savani/roberta-base-emotion

# Summarize a dataset and/or model
gft_summary --data H:emotion
gft_summary --model $emodel
gft_summary --data H:emotion --model $emodel

# find some popular datasets and models that contain "emotion"
gft_summary --data H:__contains__emotion --topn 5
gft_summary --model H:__contains__emotion --topn 5

# make predictions on inputs from stdin
echo 'I love you.' | gft_predict --task classify

# The default model (for the classification task) performs sentiment analysis
# The model, $emodel, outputs emotion classes (as opposed to POSITIVE/NEGATIVE)
echo 'I love you.' | gft_predict --task classify --model $emodel
```

Default model → Sentiment Analysis

Emotion Classes

Seven Simple Examples: 4 of 7

More tasks: token_classification, fill-mask

```
emodel=H:bhadresh-savani/roberta-base-emotion

# Summarize a dataset and/or model
gft_summary --data H:emotion
gft_summary --model $emodel
gft_summary --data H:emotion --model $emodel

# find some popular datasets and models that contain "emotion"
gft_summary --data H:__contains__emotion --topn 5
gft_summary --model H:__contains__emotion --topn 5

# make predictions on inputs from stdin
echo 'I love you.' | gft_predict --task classify
```

```
# The default model (for the classification task) performs sentiment analysis
# The model, $emodel, outputs emotion classes (as opposed to POSITIVE/NEGATIVE)
echo 'I love you.' | gft_predict --task classify --model $emodel

# some other tasks (beyond classification)
echo 'I love New York.' | gft_predict --task H:token-classification
echo 'I <mask> you.' | gft_predict --task H:fill-mask
```

~~NER: Named Entity Recognition~~

Cloze (1953)

https://en.wikipedia.org/wiki/Cloze_test

Seven Simple Examples: 5 of 7

Input from datasets (as opposed to stdin)

```
# some other tasks (beyond classification)
echo 'I love New York.' | gft_predict --task H:token-classification
echo 'I <mask> you.' | gft_predict --task H:fill-mask

# make predictions on inputs from a split of a standard dataset
gft_predict --eqn 'classify: label ~ text' --model $emodel --data H:emotion --split test
```

Specify split

Equation specifies task & columns

Read Input from dataset (as opposed to stdin)

gft_cat_data --data H:emotion --eqn 'classify:label ~ text' 2>/dev/null | head

Equation specifies task & columns

text	label
im feeling rather rotten so im not very ambitious right now	0
im updating my blog because i feel shitty	0
i never make her separate from me because i don t ever want her to feel like i m ashamed with her	0
i left with my bouquet of red and yellow tulips under my arm feeling slightly more optimistic than when i arrived	1
i was feeling a little vain when i did this one	0
i cant walk into a shop anywhere where i do not feel uncomfortable	4
i felt anger when at the end of a telephone call	3
i explain why i clung to a relationship with a boy who was in many ways immature and uncommitted despite the excitement i should have been feeling for getting accepted into the masters program at the university of virginia	1
i like to have the same breathless feeling as a reader eager to see what will happen next	1
i jest i feel grumpy tired and pre menstrual which i probably am but then again its only been a week and im about as fit as a walrus on vacation for the summer	3

Seven Simple Examples: 6 of 7
gft_predict → gft_eval

```
# some other tasks (beyond classification)
echo 'I love New York.' | gft_predict --task H:token-classification
echo 'I <mask> you.' | gft_predict --task H:fill-mask

# make predictions on inputs from a split of a standard dataset
gft_predict --eqn 'classify: label ~ text' --model $emodel --data H:emotion --split test

# return a single score (as opposed to a prediction for each input)
gft_eval --eqn 'classify: label ~ text' --model $emodel --data H:emotion --split test
```

Eval returns a single score for the dataset split
(as opposed to a prediction for each example in dataset split)

A Few Simple Examples: 7 of 7
gft_fit (aka fine-tuning)

```
# some other tasks (beyond classification)
echo 'I love New York.' | gft_predict --task H:token-classification
echo 'I <mask> you.' | gft_predict --task H:fill-mask

# make predictions on inputs from a split of a standard dataset
gft_predict --eqn 'classify: label ~ text' --model $emodel --data H:emotion --split test

# return a single score (as opposed to a prediction for each input)
gft_eval --eqn 'classify: label ~ text' --model $emodel --data H:emotion --split test

# Input a pre-trained model (bert) and output a post-trained model
gft_fit --eqn 'classify: label ~ text' \
    --model H:bert-base-cased \
    --data H:emotion \
    --output_dir $outdir
```

Fit returns a post-trained model: f_{post}

Recap of Seven Simple Examples

1

```
# Summarize a dataset and/or model
gft_summary --data H:emotion
gft_summary --model $emodel
gft_summary --data H:emotion --model $emodel

# find some popular datasets and models that contain "emotion"
gft_summary --data H:__contains__emotion --topn 5
gft_summary --model H:__contains__emotion --topn 5

# make predictions on inputs from stdin
echo 'I love you.' | gft_predict --task classify

# The default model (for the classification task) performs sentiment analysis
# The model, $emodel, outputs emotion classes (as opposed to POSITIVE/NEGATIVE)
echo 'I love you.' | gft_predict --task classify --model $emodel
```

2

3

4

```
# some other tasks (beyond classification)
echo 'I love New York.' | gft_predict --task H:token-classification
echo 'I <mask> you.' | gft_predict --task H:fill-mask
```

5

```
# make predictions on inputs from a split of a standard dataset
gft_predict --eqn 'classify: label ~ text' --model $emodel --data H:emotion --split test
```

6

```
# return a single score (as opposed to a prediction for each input)
gft_eval --eqn 'classify: label ~ text' --model $emodel --data H:emotion --split test
```

7

```
# Input a pre-trained model (bert) and output a post-trained model
gft_fit --eqn 'classify: label ~ text' \
    --model H:bert-base-cased \
    --data H:emotion \
    --output_dir $outdir
```

1. Summary
2. Summary with patterns (__*contains*__)
3. Sentiment Analysis & Text Classification
4. More tasks: token_classification, fill-mask
5. Input from datasets (as opposed to stdin)
6. *gft_predict → gft_eval*
7. *gft_fit* (aka fine-tuning)

Hundreds of Examples:
Robustness / Regression Testing

Subdirectories under $gft/examples

1. fit_examples
2. predict_examples
3. eval_examples
4. summary_examples

- Look for *.sh files under these directories
  - `find $gft/examples -name "*.sh"`
- 4 arguments:
  - --data, --model, --eqn, --task

4 Functions

1. gft_fit: $f_{pre} \rightarrow f_{post}$ (fine-tuning)
   - 4 Arguments, --output_dir, --metric, --splits
   - (plus most args in most hubs)
2. gft_predict: $f(x) \rightarrow \hat{y}$ (inference)
   - Input: 4 Arguments (x from data or stdin)
   - Output: $\hat{y}$ for each x
3. gft_eval: Score model on dataset
   - Input: 4 Arguments, --metric, …
   - Output: Score
4. gft_summary: Find good stuff
   - 4 Arguments
   - (may include: __*contains*__, __*infer*__)

Input: *I love you*
Output: Depends on Model (among other things)

- Task: classify
- Models:
  - Sentiment
    - Expected output label: positive
  - Fake News:
    - Expected output label: not fake (real)
  - Spam/Ham:
    - Expected output label: not spam (ham)

```
for model in …
 do
   echo I love you |
   gft_predict \
      --task classify \
      --model $model
 done
```

Scores could be more confident

Many of these models produce reasonable results

I love you is **positive**

Predicted Label	Score	Model	Labels for Model
positive	0.512	SetFit/deberta-v3-large__sst2__train-16-7	negative, positive
POSITIVE	0.871	ayameRushia/roberta-base-indonesian-sentiment-analysis-smsa	POSITIVE, NEUTRAL, NEGATIVE
positive	0.807	SetFit/distilbert-base-uncased__sst2__train-32-2	negative, positive
positive	0.999	AdapterHub/bert-base-uncased-pf-sst2	negative, positive
positive	0.917	SetFit/deberta-v3-large__sst2__train-32-1	negative, positive
positive	0.999	moshew/tiny-bert-aug-sst2-distilled	negative, positive
positive	0.651	rohansingh/autonlp-Fake-news-detection-system-29906863	negative, positive
5 stars	0.872	tomato/sentiment_analysis	1 star, 2 stars, 3 stars, 4 stars, 5 stars
5 stars	0.424	cmarkea/distilcamembert-base-sentiment	1 star, 2 stars, 3 stars, 4 stars, 5 stars
5 stars	0.872	nlptown/bert-base-multilingual-uncased-sentiment	1 star, 2 stars, 3 stars, 4 stars, 5 stars

But..

I love you is **fake news**

Predicted Label	Score	Model	Labels for Model
Fake	0.998	yaoyinnan/bert-base-chinese-covid19	Neutral, Fake, Real
Fake	0.986	yaoyinnan/roberta-fakeddit	Fake, Real
fake	0.958	Qiaozhen/fake-news-detector	real, fake
FAKE	0.959	Narrativaai/fake-news-detection-spanish	REAL, FAKE

I love you is both **spam** and **ham**

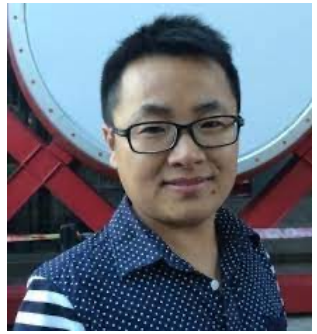
Predicted Label	Score	Model	Labels for Model
spam	0.826	SetFit/distilbert-base-uncased__enron_spam__all-train	ham, spam
not spam	1.000	sureshs/distilbert-large-sms-spam	not spam, spam

A: Ken Church

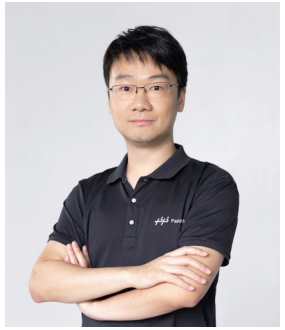
B: Valia Kordoni

B: Gary Marcus

B: Ernie Davis



A: Yanjun Ma



A: Zeyu Chen

fit and *predict* can do much (but not everything)

Part A: Glass is half full

- *gft* is (almost) all you need
  - to cover much of
  - the recent excitement in deep nets
- Amazing how much can be done
  - with so little
- But also demystifies deep nets
  - No one would suggest *fit* & *predict* are AI
- A: Ken Church, Yanjun Ma & Zeyu Chen

Part B: Glass is half empty

- Many issues go beyond *fit* & *predict*
- Traditional views in AI
  - Causality, Truth, Time, Space
  - B: Gary Marcus & Ernie Davis
- Traditional views in Linguistics
  - B: Valia Kordoni

Arguments to *fit, predict, summary, eval*

- Models: --model, --pretrained
  - BERT, ERNIE, wav2vec, distilbert-base-uncased-distilled-squad, etc.
- Datasets: --data
  - glue, super_glue, common_voice, librispeech_asr, wikitext, etc.
- Tasks: --task
  - lhs of equations: *classify, classify_tokens, classify_spans, ctc*
  - pipeline tags: *text-classification, image-classification, token-classification, question-answering, text-generation, translation, automatic-speech-recognition, fill-mask*, etc.

➢ **Equations**: --eqn (Specify task and columns in dataset)

- $classify$: $label \sim question1 + question2$
- $classify\_tokens$: $pos\_tags \sim tokens$
- $classify\_spans$: $answers \sim question + context$
- ctc: $text \sim audio$

Simple Equations Cover Many Cases of Interest
GLUE: A Popular Benchmark

Subtask	Dataset	Equation
COLA	H:glue,cola	$classify: label \sim sentence$
SST2	H:glue,sst2	$classify: label \sim sentence$
WNLI	H:glue,wnli	$classify: label \sim sentence$
MRPC	H:glue,mrpc	$classify: label \sim sentence1 + sentence2$
QNLI	H:glue,qnli	$classify: label \sim sentence1 + sentence2$
QQP	H:glue,qqp	$classify: label \sim question + sentence$
SSTB	H:glue,sstb	$regress: label \sim question1 + question2$
MNLI	H:glue,mnli	$classify: label \sim premise + hypothesis$

Select Column in Dataset

General Fine-Tuning

General Linear Models

Equations in *gft* are borrowed from *glm* in R

Example of regression in R (a statistics package)

Promised no (python) coding

```
d = read.table("data.csv", header=T, sep=",")
g = glm(y ~ x1 + x2, data=d)
```

Variables ($x1$, $x2$, y) refer to columns in data.csv

```
gft_fit --pretrained H:bert-base-cased \
    --data H:glue,qqp \
    --metric H:glue,qqp \
    --output_dir $outdir \
    --eqn 'classify: label ~ question1 + question2'
```

Variables ($label$, $question1$, $question2$) refer to columns in data

Simple Equations Cover Many Cases of Interest

Task	Subtask	Dataset	Equation
GLUE	COLA	H:glue,cola	$classify: label \sim sentence$
SQuAD 1.0		H:squad	$classify\_spans: answers \sim question + context$
SQuAD 2.0		H:squad_v2	$classify\_spans: answers \sim question + context$
CONLL2003	POS	H:conll2003	$classify\_tokens: pos\_tags \sim tokens$
	NER	H:conll2003	$classify\_tokens: ner\_tags \sim tokens$
TIMIT		H:timit_asr	$ctc: text \sim audio$
Amazon Reviews		H:amazon_reviews_multi	$classify: label \sim queston + sentence$
VAD		C:$gft/datasets/VAD/VAD	$regress: Valence + Arousal + Dominance \sim Word$

aka: Question-Answering, QA

aka: Automatic-Speech-Recognition, ASR

Custom

$\mathbb{R}^3$

$\mathbb{R}^3$

$regress: Valence + Arousal + Dominance \sim Word$

https://saifmohammad.com/WebPages/nrc-vad.html

word	Val	Arousal	Dom	Dist
open	0.620	0.480	0.569	0.00
unfold	0.612	0.510	0.520	0.06
reopen	0.656	0.528	0.568	0.06
close	0.292	0.260	0.263	0.50
closed	0.240	0.164	0.318	0.55
undecided	0.286	0.433	0.127	0.56

Table 12: Words above the double line are near *open*. The last column is the Euclidean distance to *open*.

	Antonyms				Sim
	adj	noun	verb	fallows	SimLex
cor	0.55	0.48	0.44	0.52	-0.40

Table 13: VAD distances are positively correlated with antonyms, and negatively correlated with SimLex similarities, though none of these correlations are large.

H → P: Porting from HuggingFace to Paddle (Columns have different names)

Subtask	Dataset	Equation
COLA	P:glue,cola	$classify : labels \sim sentence$
SST2	P:glue,sst-2	$classify : labels \sim sentence$
WNLI	P:glue,wnli	$classify : labels \sim sentence$
MRPC	P:glue,mrpc	$classify : labels \sim sentence1 + sentence2$
QNLI	P:glue,qnli	$classify : labels \sim sentence1 + sentence2$
QQP	P:glue,qqp	$classify : labels \sim sentence1 + sentence2$
SSTB	P:glue,sst-b	$regress : labels \sim sentence1 + sentence2$
MNLI	P:glue,mnli	$classify : labels \sim sentence1 + sentence2$

Select Column in Dataset

Subtask	Dataset	Equation
COLA	H:glue,cola	$classify : label \sim sentence$
SST2	H:glue,sst2	$classify : label \sim sentence$
WNLI	H:glue,wnli	$classify : label \sim sentence$
MRPC	H:glue,mrpc	$classify : label \sim sentence1 + sentence2$
QNLI	H:glue,qnli	$classify : label \sim sentence1 + sentence2$
QQP	H:glue,qqp	$classify : label \sim question + sentence$
SSTB	H:glue,sstb	$regress : label \sim question1 + question2$
MNLI	H:glue,mnli	$classify : label \sim premise + hypothesis$

Select Colu in Datase

Two Questions

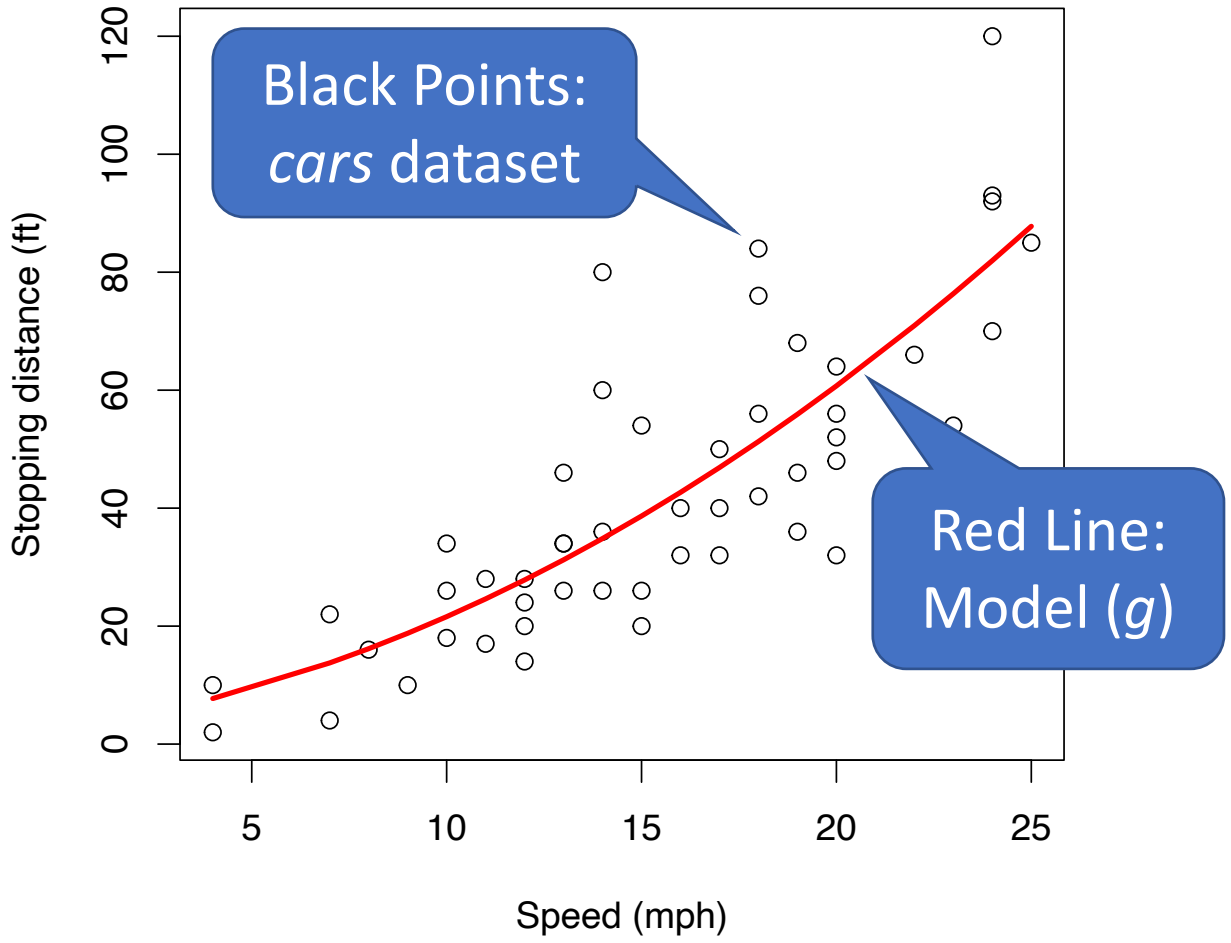
- **How do I find the *good* stuff?**
- And how do I use the *good* stuff?

Embarrassment of Riches

- Hubs have 30k models & 3k datasets (increasing 3x per year)
  - Q: How can I find the good stuff?
  - ➢ **A: *gft_summary***

Regression in *R*:

Summarize (almost) anything




```
> head(cars)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
> summary(cars)
     speed           dist       
 Min.   : 4.0   Min.   :  2.00  
 1st Qu.:12.0   1st Qu.: 26.00  
 Median :15.0   Median : 36.00  
 Mean   :15.4   Mean   : 42.98  
 3rd Qu.:19.0   3rd Qu.: 56.00  
 Max.   :25.0   Max.   :120.00  
>
```

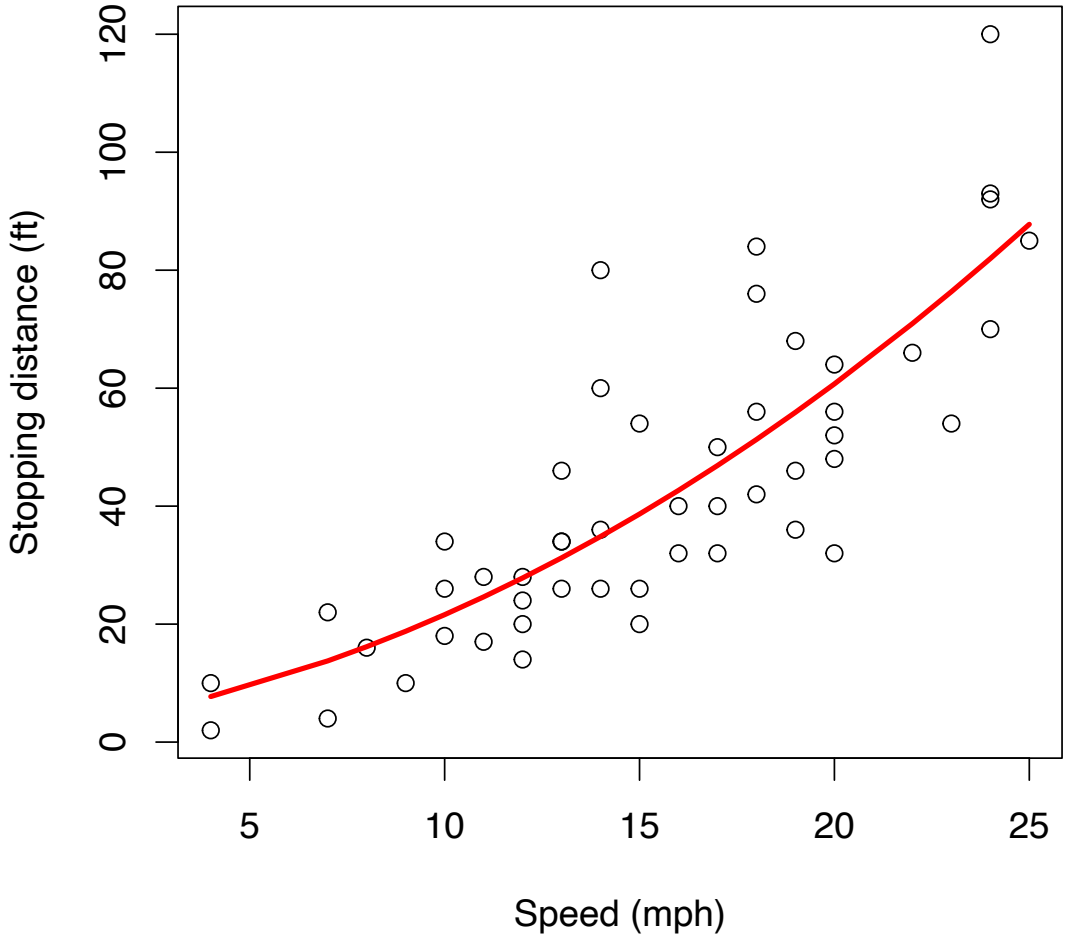
Summarize a dataset (*cars*)

Fit a model (*g*) to data (*cars*)

Regression in *R*:

Summarize (almost) anything

```
> plot(cars, xlab="Speed (mph)", ylab="Stopping distance (ft)")
> g = glm(dist ~ poly(speed,2), data=cars)
> o = order(cars$speed)
> lines(cars$speed[o], predict(g,cars)[o], col="red", lwd=3)
```



Regression in *R*:

Summarize (almost) anything

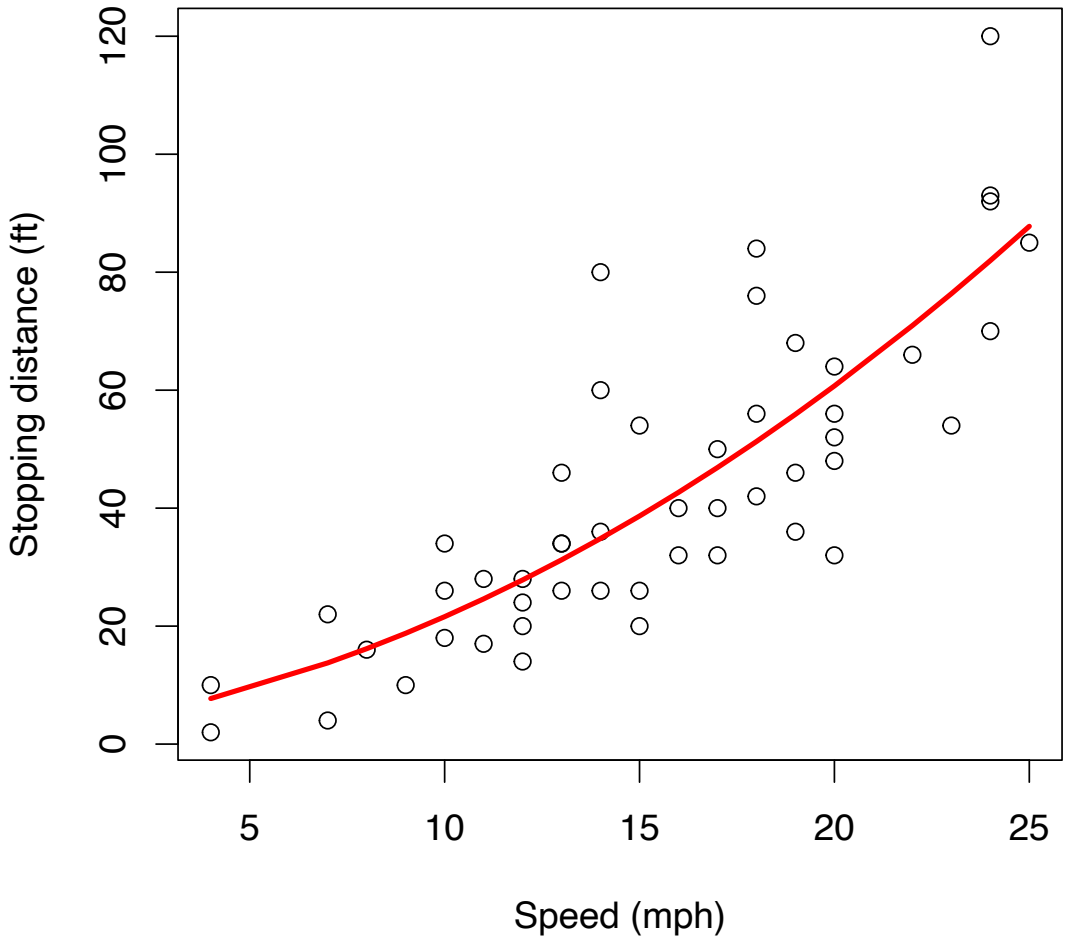
Fit a model (*g*) to data (*cars*)

```
> plot(cars, xlab="Speed (mph)", ylab="Stopping distance (ft)")
> g = glm(dist ~ poly(speed,2), data=cars)
> o = order(cars$speed)
> lines(cars$speed[o], predict(g,cars)[o], col="red", lwd=3)
> summary(g)
```

Summarize a model (*g*)

```
Call:
glm(formula = dist ~ poly(speed, 2), data = cars)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-28.720   -9.184   -3.188    4.628   45.152

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       42.980      2.146  20.026  < 2e-16 ***
poly(speed, 2)1  145.552     15.176   9.591 1.21e-12 ***
poly(speed, 2)2   22.996     15.176   1.515    0.136
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 230.3131)

    Null deviance: 32539  on 49  degrees of freedom
Residual deviance: 10825  on 47  degrees of freedom
AIC: 418.77

Number of Fisher Scoring iterations: 2
```

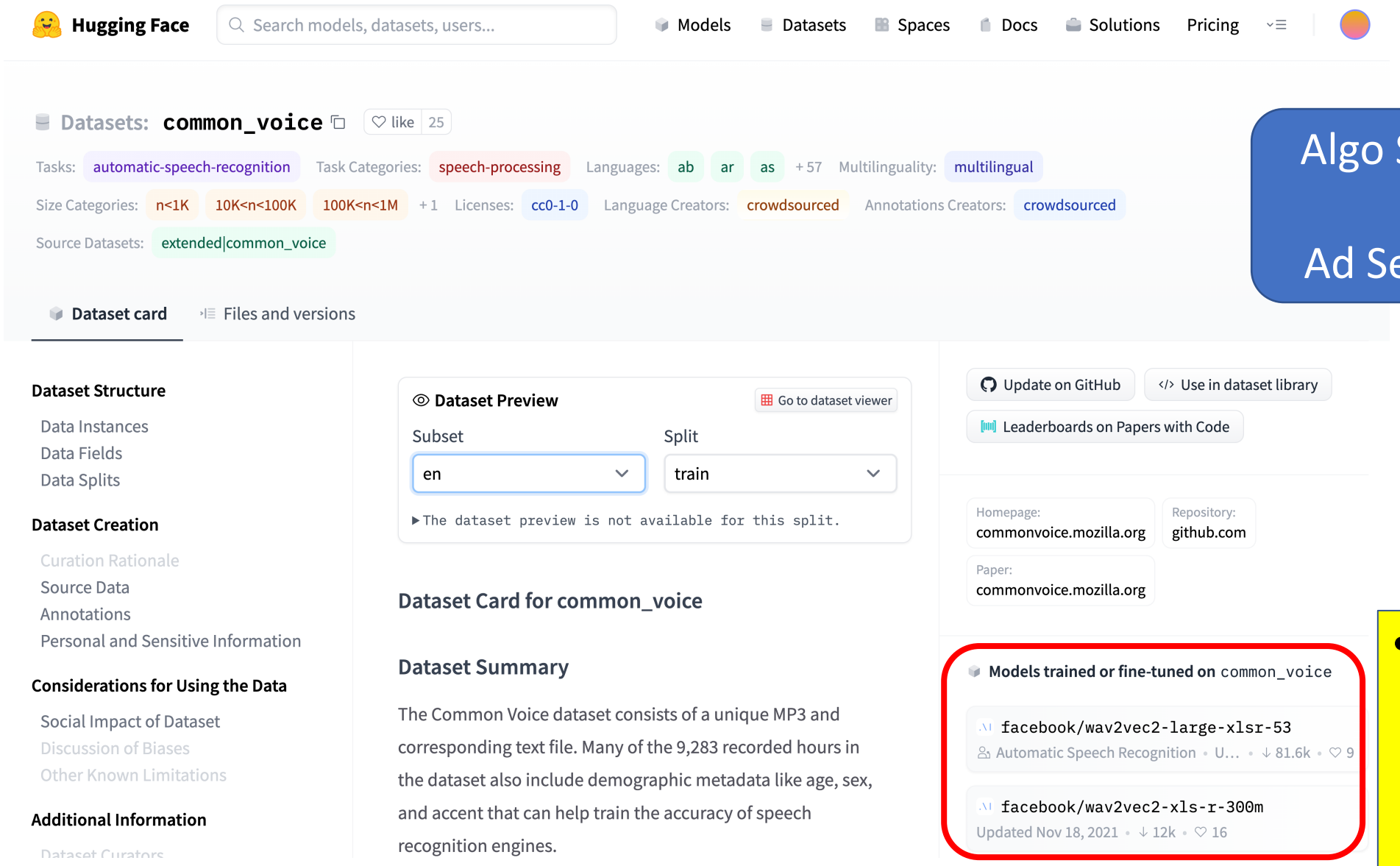
Opportunity to Improve *g*



gft_summary:
Summarize (almost) anything: datasets, models, tasks, etc.

```
# summarize the qqp subtask of glue (from HuggingFace)
gft_summary --data H:glue,qqp 2>/dev/null
# dataset: glue,qqp splits: train: 363846 rows, test: 390965 rows, val: 40430 rows
# dataset: glue,qqp split: train    columns: question1, question2, label, idx
# dataset: glue,qqp labels: not_duplicate, duplicate
```

Metrics of Success: How do we find good stuff?



Algo Search: Left Rail
vs
Ad Search: Right Rail

A kind of search engine

- Score dataset by:
  - # of downloads
  - # of likes
  - # of models
  - # of citations

Embarrassment of Riches

✓Hubs have 30k models & 3k datasets (increasing 3x per year)

- Q: How can I find the good stuff?

➢ **A: *gft_summary***

- ➢ __contains__
  - ➢ Find popular models/datasets that contain substring
- ➢ __infer__
  - ➢ Find popular models that are associated with task/dataset

➢ Examples:

- ➢ gft_summary --data H:__contains__emotion --topn 5
- ➢ gft_summary --model H:__contains__emotion --topn 5
- ➢ gft_summary --task H:classify --model H:__infer__ --topn 5
- ➢ gft_summary --data H:emotion --model H:__infer__ --topn 5

How do I find the good stuff? Popular Datasets? gft_summary –data H:__contains__wiki --topn 20

Dataset	Downloads	Likes	PWC
wikitext	119,031	7	https://paperswithcode.com/dataset/wikitext-2
wikiann	20,021	6	https://paperswithcode.com/dataset/wikiann-1
wiki_snippets	16,550	0	
wikipedia	14,729	8	
wiki_bio	10,135	0	https://paperswithcode.com/dataset/wikibio
wiki_qa	9,670	1	https://paperswithcode.com/dataset/wikiqa
wiki_hop	8,961	0	https://paperswithcode.com/dataset/wikihop
wiki_dpr	5,310	0	
wiki_split	5,260	0	https://paperswithcode.com/dataset/wikisplit
wiki40b	2,618	3	https://paperswithcode.com/dataset/wiki-40b
wikisql	2,135	2	https://paperswithcode.com/dataset/wikisql
wikihow	1,601	0	https://paperswithcode.com/dataset/wikihow
wiki_lingua	1,508	1	https://paperswithcode.com/dataset/wikilingua
kilt_wikipedia	1,421	2	
Tevatron/wikipedia-nq-corpus	1,368	0	
wiki_auto	1,258	0	
wiki_asp	1,192	1	https://paperswithcode.com/dataset/wikiasp
wikicorpus	1,098	0	
wiki_atomic_edits	1,058	3	https://paperswithcode.com/dataset/wikiatomicedits
iapp_wiki_qa_squad	1,033	0	

How do I find the good stuff? Popular Datasets? gft_summary –data H:__contains__ --topn 20

Dataset	Downloads	Likes	PWC
glue	926,082	26	https://paperswithcode.com/dataset/glue
super_glue	306,473	9	https://paperswithcode.com/dataset/superglue
wikitext	119,031	7	https://paperswithcode.com/dataset/wikitext-2
imdb	105,321	8	https://paperswithcode.com/dataset/imdb-movie-reviews
squad	99,954	22	https://paperswithcode.com/dataset/squad
squad_es	72,163	0	https://paperswithcode.com/dataset/squad-es
paws	65,433	1	https://paperswithcode.com/dataset/paws
librispeech_asr	65,325	12	https://paperswithcode.com/dataset/librispeech-1
wmt16	48,479	2	https://paperswithcode.com/dataset/wmt-2016
xnli	41,987	5	https://paperswithcode.com/dataset/xnli
snli	34,045	5	https://paperswithcode.com/dataset/snli
ag_news	30,456	11	https://paperswithcode.com/dataset/ag-news
anli	30,245	3	https://paperswithcode.com/dataset/anli
amazon_polarity	28,584	6	
squad_v2	28,000	3	https://paperswithcode.com/dataset/squad
conll2003	26,012	11	https://paperswithcode.com/dataset/conll-2003
red_caps	25,940	2	https://paperswithcode.com/dataset/redcaps
common_voice	25,033	26	https://paperswithcode.com/dataset/common-voice
stsb_multi_mt	23,866	4	
trec	23,313	3	https://paperswithcode.com/dataset/trecqa

How do I find the good stuff? Popular Models? gft_summary –model H:__contains__snli --topn 20

Model	Downloads	Likes	Task
ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli	31,354	2	text-classification
textattack/bert-base-uncased-snli	6,651	0	text-classification
boychaboy/SNLI_roberta-base	3,158	0	text-classification
symanto/sn-xlm-roberta-base-snli-mnli-anli-xnli	2,575	1	sentence-similarity
ynie/albert-xxlarge-v2-snli_mnli_fever_anli_R1_R2_R3-nli	2,093	1	text-classification
pritamdeka/S-Biomed-Roberta-snli-multinli-stsb	671	0	sentence-similarity
jegorkitskerkin/bert-base-dutch-cased-snli	528	1	sentence-similarity
ynie/bart-large-snli_mnli_fever_anli_R1_R2_R3-nli	274	0	text-classification
symanto/xlm-roberta-base-snli-mnli-anli-xnli	151	1	text-classification
persiannlp/mt5-base-parsinlu-snli-entailment	115	0	text2text-generation
usc-isi/sbert-roberta-large-anli-mnli-snli	97	0	sentence-similarity
ynie/xlnet-large-cased-snli_mnli_fever_anli_R1_R2_R3-nli	94	0	text-classification
textattack/albert-base-v2-snli	67	0	text-classification
pmthangk09/bert-base-uncased-esnli	53	0	text-classification
boychaboy/SNLI_bert-base-uncased	46	0	text-classification
ynie/electra-large-discriminator-snli_mnli_fever_anli_R1_R2_R3-nli	44	0	text-classification
NDugar/debertav3-mnli-snli-anli	42	2	zero-shot-classification
boychaboy/SNLI_bert-large-cased	39	0	text-classification
persiannlp/mt5-small-parsinlu-snli-entailment	37	0	text2text-generation
boychaboy/SNLI_roberta-large	33	0	text-classification

gft_summary: How do I find the good stuff?
Find models associated with a task (ASR)

task → list of models (by downloads)

```
# How do I find the good stuff?
# Return a list of models for a particular task (sorted by downloads)
gft_summary --task H:ASR --model H:__infer__ 2>/dev/null | head
# task: AutomaticSpeechRecognition --> 1643 models
# task: AutomaticSpeechRecognition  model: facebook/wav2vec2-base-960h  downloads: 1092953  like
# task: AutomaticSpeechRecognition  model: facebook/hubert-large-ls960-ft   downloads: 65234
# task: AutomaticSpeechRecognition  model: facebook/wav2vec2-large-960h-lv60-self   downloads: 6
# task: AutomaticSpeechRecognition  model: facebook/wav2vec2-large-xlsr-53  downloads: 55869
# task: AutomaticSpeechRecognition  model: hf-internal-testing/processor_with_lm    downloads: 2
# task: AutomaticSpeechRecognition  model: pyannote/voice-activity-detection    downloads: 1895
# task: AutomaticSpeechRecognition  model: patrickvonplaten/wavlm-libri-clean-100h-base-plus
# task: AutomaticSpeechRecognition  model: facebook/wav2vec2-large-960h downloads: 13622    like
# task: AutomaticSpeechRecognition  model: facebook/s2t-small-librispeech-asr   downloads: 11808
```

gft_summary: How do I find the good stuff?
Find models associated with a task (ASR)

task → list of models (by downloads)

```
# How do I find the good stuff?
# Return a list of models for a particular task (sorted by downloads)
gft_summary --task H:ASR --model H:__infer__ 2>/dev/null | head
# task: AutomaticSpeechRecognition --> 1643 models
# task: AutomaticSpeechRecognition  model: facebook/wav2vec2-base-960h  downloads: 1092953  like
# task: AutomaticSpeechRecognition  model: facebook/hubert-large-ls960-ft  downloads: 65234
# task: AutomaticSpeechRecognition  model: facebook/wav2vec2-large-960h-lv60-self  downloads:
# task: AutomaticSpeechRecognition  model: facebook/wav2vec2-large-
# task: AutomaticSpeechRecognition  model: hf-internal-testing/proc
# task: AutomaticSpeechRecognition  model: pyannote/voice-activity-
# task: AutomaticSpeechRecognition  model: patrickvonplaten/wavlm-l
# task: AutomaticSpeechRecognition  model: facebook/wav2vec2-large-
# task: AutomaticSpeechRecognition  model: facebook/s2t-small-libri
```

dataset → list of models (by downloads)

```
# How do I find the good stuff?
# Return a list of models for a particular dataset (sorted by downloads)
gft_summary --data H:common_voice,en --model H:__infer__ 2>/dev/null | head
# dataset: common_voice,en  splits: train: 564337 rows, test: 16164 rows, other: 169895 rows, i
# dataset: common_voice,en  split: train  columns: client_id, path, audio, sentence, up_votes,
# dataset: common_voice --> 542 models
# dataset: common_voice model: facebook/wav2vec2-large-xlsr-53  downloads: 55869  likes: 12
# dataset: common_voice model: jonatasgrosman/wav2vec2-large-xlsr-53-english  downloads: 11633
# dataset: common_voice model: facebook/wav2vec2-xls-r-300m downloads: 10156  likes: 16
# dataset: common_voice model: vumichien/wav2vec2-large-xlsr-japanese-hiragana  downloads: 8454
# dataset: common_voice model: jonatasgrosman/wav2vec2-large-xlsr-53-german downloads: 6338 like
# dataset: common_voice model: jonatasgrosman/wav2vec2-large-xlsr-53-russian  downloads: 4689
# dataset: common_voice model: facebook/wav2vec2-large-xlsr-53-german  downloads: 4498 likes:
```

gft_summary: How do I find the good stuff?
Find popular models associated with a dataset

```
# find some models associated with a dataset (and sort by downloads)
gft_summary --data H:emotion --model H:__infer__ 2>/dev/null | head
# dataset: emotion  split: train    columns: text, label
# dataset: emotion  labels: sadness, joy, love, anger, fear, surprise
# dataset: emotion --> 69 models
# dataset: emotion  model: bhadresh-savani/distilbert-base-uncased-emotion  downloads: 505632
# dataset: emotion  model: nateraw/bert-base-uncased-emotion    downloads: 5836 likes: 1
# dataset: emotion  model: mrm8488/t5-base-finetuned-emotion    downloads: 3927 likes: 3
# dataset: emotion  model: mrm8488/t5-small-finetuned-emotion   downloads: 1580 likes: 0
# dataset: emotion  model: bhadresh-savani/roberta-base-emotion downloads: 1432 likes: 0
# dataset: emotion  model: bhadresh-savani/bert-base-uncased-emotion    downloads: 587  likes: 1
# dataset: emotion  model: lewiswatson/distilbert-base-uncased-finetuned-emotion    downloads: 2
```

gft_summary: How do I find the good stuff? Find popular models associated with a task (MT)

```
# ditto, but for machine translation
gft_summary --task H:translation --model H:__infer__ 2>/dev/null | head
# task: Translation --> 1470 models
# task: Translation model: Helsinki-NLP/opus-mt-zh-en   downloads: 7606505  likes: 17
# task: Translation model: t5-small downloads: 986817   likes: 5
# task: Translation model: t5-base  downloads: 812813   likes: 32
# task: Translation model: Helsinki-NLP/opus-mt-en-de   downloads: 721507   likes: 2
# task: Translation model: Helsinki-NLP/opus-mt-en-ROMANCE  downloads: 435634   likes: 1
# task: Translation model: Helsinki-NLP/opus-mt-ar-en   downloads: 185275   likes: 3
# task: Translation model: Helsinki-NLP/opus-mt-es-en   downloads: 175059   likes: 6
# task: Translation model: Helsinki-NLP/opus-mt-de-en   downloads: 144799   likes: 4
# task: Translation model: Helsinki-NLP/opus-mt-fr-en   downloads: 141266   likes: 1
```

Embarrassment of Riches

- ✓Hubs have 30k models & 3k datasets (increasing 3x per year)
  - ✓Q: How can I find the good stuff?
  - ✓A: *gft_summary*
    - ✓ __contains__
      - ✓ Find popular models/datasets that contain substring
    - ✓ __infer__
      - ✓ Find popular models that are associated with task/dataset
  - ✓Examples:
    - ✓gft_summary --data H:__contains__emotion --topn 5
    - ✓gft_summary --model H:__contains__emotion --topn 5
    - ✓gft_summary --task H:classify --model H:__infer__ --topn 5
    - ✓gft_summary --data H:emotion --model H:__infer__ --topn 5

What counts as ``good'' stuff?

- Academic Metrics:
  - Peer Review (Citations)
- Social Media Metrics:
  - Popular (Downloads)
  - Tweets / Blogs
- SOTA: State of the art
  - Papers with Code
  - Leaderboards
- Search Engine Optimization (SEO)
  - Adversarial Game (Arbitrage):
    - Better Web Search → ``Better'' SEOs → Better Web Search

Model	VAcc	Dload
C:Best of Yuchen Bian's models	0.924	
H:textattack/roberta-base-MRPC	0.912	1623
H:textattack/albert-base-v2-MRPC	0.897	175
H:mrm8488/deberta-v3-small-finetuned-mrpc	0.892	30
H:textattack/bert-base-uncased-MRPC	0.877	10,133
H:textattack/distilbert-base-uncased-MRPC	0.858	108
H:ajrae/bert-base-uncased-finetuned-mrpc	0.858	115
C:*gft_fit* example (with default settings)	0.853	
H:textattack/distilbert-base-cased-MRPC	0.784	122

Table 4: *gft* achieves VAcc (accuracy on validation split) close to distilbert (compressed) models. HuggingFace models were selected using *gft_summary* to find popular models by downloads (Dload).

Source	Test Accuracy
GLUE Leaderboard (L)	0.945
Papers with code (PWC)	0.937
Human Baseline (HB)	0.863

Table 5: SOTA (state-of-the-art) for MRPC (GLUE). See footnote 11 for PWC, and 12 for L & HB.

How do I find the good stuff? Popular Models?
gft_summary –model H:__contains__snli --topn 20

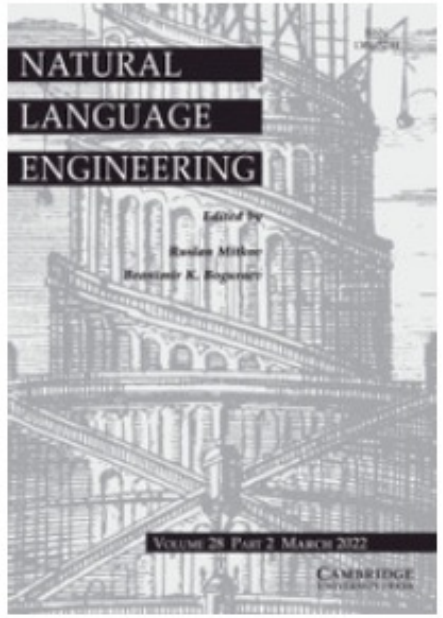
Model	Downloads	Likes	Task
ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli	31,354	2	text-classification
textattack/bert-base-uncased-snli	6,651	0	text-classification
boychaboy/SNLI_roberta-base	3,158	0	text-classification
symanto/sn-xlm-roberta-base-snli-mnli-anli-xnli	2,575	1	sentence-similarity
ynie/albert-xxlarge-v2-snli_mnli_fever_anli_R1_R2_R3-nli	2,093	1	text-classification
pritamdeka/S-Biomed-Roberta-snli-multinli-stsb	671	0	sentence-similarity
jegorkitskerkin/bert-base-dutch-cased-snli	528	1	sentence-similarity
ynie/bart-large-snli_mnli_fever_anli_R1_R2_R3-nli	274	0	text-classification
symanto/xlm-roberta-base-snli-mnli-anli-xnli	151	1	text-classification
persiannlp/mt5-base-parsinlu-snli-entailment	115	0	text2text-generation
usc-isi/sbert-roberta-large-anli-mnli-snli	97	0	sentence-similarity
ynie/xlnet-large-cased-snli_mnli_fever_anli_R1_R2_R3-nli	94	0	text-classification
textattack/albert-base-v2-snli	67	0	text-classification
pmthangk09/bert-base-uncased-esnli	53	0	text-classification
boychaboy/SNLI_bert-base-uncased	46	0	text-classification
ynie/electra-large-discriminator-snli_mnli_fever_anli_R1_R2_R3-nli	44	0	text-classification
NDugar/debertav3-mnli-snli-anli	42	2	zero-shot-classification
boychaboy/SNLI_bert-large-cased	39	0	text-classification
persiannlp/mt5-small-parsinlu-snli-entailment	37	0	text2text-generation
boychaboy/SNLI_roberta-large	33	0	text-classification

Downloads ≠ SOTA

Defaults & Ease-of-Use

SOTA-Chasing: Leaderboards Considered Harmful

SOTA-chasing refers to papers that report SOTA numbers, but contribute little of lasting value to the literature. The point is the pointlessness.



Natural Language Engineering

Emerging Trends: SOTA-Chasing

Published online by Cambridge University Press: **08 February 2022**

Kenneth Ward Church and Valia Kordoni

Article | Figures | Metrics

Save PDF | Share | Cite

Abstract

Most read

This page lists the top ten most read articles for this journal based on the number of full text views and downloads recorded on Cambridge Core over the last 30 days. This list is updated on a daily basis.

GPT-3: What's it good for?
Robert Dale
Published online by Cambridge University Press: 15 December 2020, pp. 113-118
Article | Access | Open access | PDF | HTML | Export citation
View abstract

Word2Vec
KENNETH WARD CHURCH
Published online by Cambridge University Press: 16 December 2016, pp. 155-162
Article | Access | Open access | PDF | HTML | Export citation
View abstract

Emerging Trends: SOTA-Chasing
Kenneth Ward Church, Valia Kordoni
Published online by Cambridge University Press: 08 February 2022, pp. 249-269
Article | Access | Open access | PDF | HTML | Export citation
View abstract

PWC: https://paperswithcode.com/dataset/glue

Search

Browse State-of-the-Art | Datasets | Methods | More | We are hiring!

Aα Texts

GLUE (General Language Understanding Evaluation benchmark)

Edit

Introduced by Wang et al. in GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding

General Language Understanding Evaluation (**GLUE**) benchmark is a collection of nine natural language understanding tasks, including single-sentence tasks CoLA and SST-2, similarity and paraphrasing tasks MRPC, STS-B and QQP, and natural language inference tasks MNLI, QNLI, RTE and WNLI.

Source: Align, Mask and Select: A Simple Method for Incorporating Commonsense Knowledge into Language Representation Models

Homepage

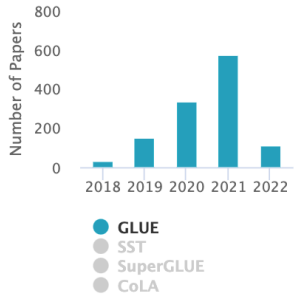
Source: https://gluebenchmark.com/.

Benchmarks

Edit

Trend	Task	Dataset Variant	Best Model	Paper	Code
	Text Classification	GLUE	roberta-base-finetuned-sst2		
	Sentiment Analysis	SST-2 Binary classification	SMART-RoBERTa Large		
	Semantic Textual Similarity	STS Benchmark	SMART-RoBERTa Large		

Usage



PWC ≠ GLUE Leaderboard

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m
1	JDExplore d-team	Vega v1		91.3	73.8	97.9	94.5/92.6	93.5/93.1	76.7/91.1	92.1
2	Microsoft Alexander v-team	Turing NLR v5		91.2	72.6	97.6	93.8/91.7	93.7/93.3	76.4/91.1	92.6
3	DIRL Team	DeBERTa + CLEVER		91.1	74.7	97.6	93.3/91.1	93.4/93.1	76.5/91.0	92.1
4	ERNIE Team - Baidu	ERNIE		91.1	75.5	97.8	93.9/91.8	93.0/92.6	75.2/90.9	92.3
5	AliceMind & DIRL	StructBERT + CLEVER		91.0	75.3	97.7	93.9/91.9	93.5/93.1	75.6/90.8	91.7
6	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4		90.8	71.5	97.5	94.0/92.0	92.9/92.6	76.2/90.8	91.9
7	HFL iFLYTEK	MacALBERT + DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	91.3
8	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6
9	T5 Team - Google	T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2
10	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0
11	Huawei Noah's Ark Lab	NEZHA-Large		89.8	71.7	97.3	93.3/91.0	92.4/91.9	75.2/90.7	91.5
12	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)		89.7	70.5	97.5	93.4/91.2	92.6/92.3	75.4/90.7	91.4
13	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	91.3
14	Shi-Wen Ni	DropSAT-RoBERTa-large		88.8	70.2	96.7	92.6/90.1	92.1/91.8	75.1/90.5	91.1
15	DropAttack Team	DropAK-ELECTRA-large		88.7	70.4	95.8	92.6/90.1	91.2/91.1	75.1/90.5	91.1
16	R-AT Paper	RoBERTa-large + R-AT		88.6	69.0	96.7	92.5/90.0	92.1/91.8	75.0/90.5	91.1
17	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1
18	Junjie Yang	HIRE-RoBERTa		88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7
19	Shiwen Ni	ELECTRA-large-M (bert4keras)		88.3	69.3	95.8	92.2/89.6	91.2/91.1	75.1/90.5	91.1
20	Facebook AI	RoBERTa		88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8

How do I find ``good'' tasks?

- ✓``Good'' tasks are like ``good'' stuff
  - ✓Academic Metrics:
    - ✓Peer Review (Citations)
  - ✓Social Media Metrics:
    - ✓Popular (Downloads)
    - ✓Tweets / Blogs
  - ✓SOTA: State of the art
    - ✓Papers with Code
    - ✓Leaderboards
  - ✓Search Engine Optimization (SEO)
    - ✓Adversarial Game (Arbitrage):
      - ✓ Better Web Search → ``Better'' SEOs → Better Web Search

--task argument → HuggingFace Pipelines

https://huggingface.co/docs/transformers/v4.16.2/en/main_classes/pipelines

```
# text-classification: sentiment analysis
echo 'I love you.' | gft_predict --task text-classification
# I love you.   POSITIVE    0.9998705387115479

# text-classification: emotion classification
model=H:AdapterHub/bert-base-uncased-pf-emotion
echo 'I love you.' | gft_predict --model $model --task text-classification
# I love you.   love    0.6005669236183167

# token-classification: NER (Named Entity Recognition)
echo 'I love New York.' | gft_predict --task token-classification
# I love New York.  New/I-LOC:0.9989    York/I-LOC:0.9974

# fill-mask: guess the masked word
echo 'I <mask> you.' | gft_predict --task fill-mask
# I <mask> you.  salute|0.241    miss|0.177  love|0.147  thank|0.060     applaud|0.047

# text-generation
echo 'I love ' | gft_predict --task text-generation
# I love you and I will never be forgotten and thank you." I was also
# inspired by all of the students who walked onto campus wearing these
#  teddy I love the idea that you can be anything people ask for you

# translation
echo 'I love you.' | gft_predict --task translation --model H:Helsinki-NLP/opus-mt-en-fr
# I love you.   Je t'aime.
echo 'I love you.' | gft_predict --task translation --model H:Helsinki-NLP/opus-mt-en-zh
I love you. 我爱你
```

task (`str`) — The task defining which pipeline will be returned. Currently accepted tasks are:

- `"audio-classification"`: will return a AudioClassificationPipeline.
- `"automatic-speech-recognition"`: will return a AutomaticSpeechRecognitionPipeline.
- `"conversational"`: will return a ConversationalPipeline.
- `"feature-extraction"`: will return a FeatureExtractionPipeline.
- `"fill-mask"`: will return a FillMaskPipeline:.
- `"image-classification"`: will return a ImageClassificationPipeline.
- `"question-answering"`: will return a QuestionAnsweringPipeline.
- `"table-question-answering"`: will return a TableQuestionAnsweringPipeline.
- `"text2text-generation"`: will return a Text2TextGenerationPipeline.
- `"text-classification"` (alias `"sentiment-analysis"` available): will return a TextClassificationPipeline.
- `"text-generation"`: will return a TextGenerationPipeline:.
- `"token-classification"` (alias `"ner"` available): will return a TokenClassificationPipeline.
- `"translation"`: will return a TranslationPipeline.
- `"translation_xx_to_yy"`: will return a TranslationPipeline.
- `"summarization"`: will return a SummarizationPipeline.
- `"zero-shot-classification"`: will return a ZeroShotClassificationPipeline.

Equation Keywords ≈ Pipeline Tasks

Task	Subtask	Dataset	Equation
GLUE	COLA	H:glue,cola	$classify: label \sim sentence$
SQuAD 1.0		H:squad	$classify\_spans: answers \sim question + context$
SQuAD 2.0		H:squad_v2	$classify\_spans: answers \sim question + context$
CONLL2003	POS	H:conll2003	$classify\_tokens: pos\_tags \sim tokens$
	NER	H:conll2003	$classify\_tokens: ner\_tags \sim tokens$
TIMIT		H:timit_asr	$ctc: text \sim audio$
Amazon Reviews		H:amazon_reviews_multi	$classify: label \sim queston + sentence$
VAD		C:$gft/datasets/VAD/VAD	$regress: Valence + Arousal + Dominance \sim Word$

What are the most popular tasks?
30k Models → Pipeline Tasks

(computed from $gft/gft_internals/huggingface_hub/huggingface_datasets.txt)

Models	Pipeline Task
9537	None
3949	text-generation
3143	text-classification (*classify*)
2390	fill-mask
2253	text2text-generation
1508	automatic-speech-recognition (*ctc*)
1460	translation
1340	token-classification (*classify_tokens*)
1042	conversational
983	question-answering (*classify_spans*)
982	feature-extraction
338	sentence-similarity
320	summarization
261	image-classification

Models	Pipeline Task
160	text-to-speech
68	zero-shot-classification
55	audio-to-audio
54	audio-classification
26	table-question-answering
12	object-detection
11	image-segmentation
9	text-to-image
4	structured-data-classification
4	image-to-text
2	voice-activity-detection
1	zero-shot-image-classification
1	speech-segmentation
1	protein-folding
1	pipeline_tag

Popular Text Classification Models
gft_summary --task H:text-classification
Input: Task → Output: Popular Models

Model	Downloads	Likes
cross-encoder/ms-marco-MiniLM-L-12-v2	11,542,439	4
distilbert-base-uncased-finetuned-sst-2-english	4,188,846	38
facebook/bart-large-mnli	1,253,261	89
cross-encoder/nli-distilroberta-base	581,821	6
cardiffnlp/twitter-roberta-base-sentiment	542,530	40
nlptown/bert-base-multilingual-uncased-sentiment	541,492	30
roberta-large-mnli	520,103	12
finiteautomata/beto-sentiment-analysis	496,725	6
j-hartmann/emotion-english-distilroberta-base	273,134	16
cardiffnlp/twitter-roberta-base-emotion	259,369	5

Summarize Tasks $gft/examples/summary_examples/task

- gft_summary --task H:audio-classification
- gft_summary --task H:audio-to-audio
- gft_summary --task H:automatic-speech-recognition
- gft_summary --task H:conversational
- gft_summary --task H:feature-extraction
- gft_summary --task H:fill-mask
- gft_summary --task H:image-classification
- gft_summary --task H:image-segmentation
- gft_summary --task H:image-to-text
- gft_summary --task H:object-detection
- gft_summary --task H:protein-folding
- gft_summary --task H:question-answering
- gft_summary --task H:sentence-similarity
- gft_summary --task H:speech-segmentation
- gft_summary --task H:structured-data-classification
- gft_summary --task H:summarization
- gft_summary --task H:table-question-answering
- gft_summary --task H:text2text-generation
- **gft_summary --task H:text-classification**
- gft_summary --task H:text-generation
- gft_summary --task H:text-to-image
- gft_summary --task H:text-to-speech
- gft_summary --task H:token-classification
- gft_summary --task H:translation
- gft_summary --task H:voice-activity-detection
- gft_summary --task H:zero-shot-classification
- gft_summary --task H:zero-shot-image-classification

30k Models → Pipeline Tasks

(computed from $gft/gft_internals/huggingface_hub/huggingface_datasets.txt)

Models	**Pipeline Task**
9537	None
3949	text-generation
3143	text-classification (*classify*)
2390	fill-mask
2253	text2text-generation
1508	automatic-speech-recognitic
1460	translation
1340	token-classification (*classify_*
1042	conversational
983	question-answering (*classify*
982	feature-extraction
338	sentence-similarity
320	summarization
261	image-classification

# of Models	Task	Models
3949	text-generation	distilgpt2, gpt2, EleutherAI/gpt-neo-1.3B
3143	text-classification	cross-encoder/ms-marco-MiniLM-L-12-v2, distilbert-base-uncased-finetuned-sst-2-eng
2390	fill-mask bert-base-uncased	distilbert-base-uncased, roberta-base
2253	text2text-generation	facebook/m2m100_418M, facebook/mbart-large-50-one-to-many-mmt, google/mt5-base
1508	ctc	facebook/wav2vec2-base-960h, facebook/hubert-large-ls960-ft, facebook/wav2vec2-lar
1460	translation	Helsinki-NLP/opus-mt-zh-en, t5-small, t5-base
1340	classify	xlm-roberta-large-finetuned-conll03-english, classla/bcms-bertic-ner, dslim/bert-base-N
1042	conversational	microsoft/DialoGPT-small, microsoft/DialoGPT-medium, facebook/blenderbot_small-90M
983	classify_spans	deepset/roberta-base-squad2, distilbert-base-cased-distilled-squad, bert-large-uncased
982	feature-extraction	feature-extraction, openai/clip-vit-base-patch32, facebook/bart-base, monsoon-nlp/hind
338	sentence-similarity	sentence-transformers/multi-qa-MiniLM-L6-cos-v1, sentence-transformers/paraphrase-M L6-v2

Two Questions

✓How do I ***find*** the *good* stuff?

➢**And how do I *use* the *good* stuff?**

➢**A: gft_predict**

Question: How do I ***use*** the good stuff?
Answer: *gft_predict*

$f(x) \rightarrow \hat{y}$

No model specified

```
# text-classification: sentiment analysis
echo 'I love you.' | gft_predict --task text-classification
# I love you.   POSITIVE    0.9998705387115479
```

x $\hat{y}$ Score

Inference: *gft_predict*

- gft_predict
  - Input from a dataset or *stdin*
  - Output to *stdout*
- Arguments (many are optional)
  - --data, --split, --model, --eqn, --task
  - --eqn $task$: $lhs \sim rhs$
    - Variables in lhs and rhs
    - refer to columns in dataset
- Since the terminology for tasks is currently in a state of flux,
  - gft supports a number of aliases
  - (shown in parentheses)

- Tasks (aliases in parenthesis)
  1. classify (text-classification)
  2. classify_tokens
     - (token-classification)
  3. classify_spans
     - (QA, question-answering)
  4. classify_audio (audio-classification)
  5. classify_images (image-classification)
  6. regress
  7. text-generation
  8. MT (translation)
  9. ASR
     - (ctc, automatic-speech-recognition)
  10. fill-mask

Specifying –model argument

```
# text-classification: sentiment analysis
echo 'I love you.' | gft_predict --task text-classification
# I love you.   POSITIVE    0.9998705387115479

# text-classification: emotion classification
model=H:AdapterHub/bert-base-uncased-pf-emotion
echo 'I love you.' | gft_predict --model $model --task text-classification
# I love you.   love    0.6005669236183167
```

Default Model → Sentiment Classes

A Model with Emotion Classes

```
# text-classification: sentiment analysis
echo 'I love you.' | gft_predict --task text-classification
# I love you.   POSITIVE    0.9998705387115479

# text-classification: emotion classification
model=H:AdapterHub/bert-base-uncased-pf-emotion
echo 'I love you.' | gft_predict --model $model --task text-classification
# I love you.   love    0.6005669236183167

# token-classification: NER (Named Entity Recognition)
echo 'I love New York.' | gft_predict --task token-classification
# I love New York.  New/I-LOC:0.9989    York/I-LOC:0.9974
```

More tasks (1 of 4)

✓Text Classification

➢**Token Classification**

- Fill Mask
- Text Generation
- Machine Translation

```
# text-classification: sentiment analysis
echo 'I love you.' | gft_predict --task text-classification
# I love you.   POSITIVE    0.9998705387115479

# text-classification: emotion classification
model=H:AdapterHub/bert-base-uncased-pf-emotion
echo 'I love you.' | gft_predict --model $model --task text-classification
# I love you.   love    0.6005669236183167

# token-classification: NER (Named Entity Recognition)
echo 'I love New York.' | gft_predict --task token-classification
# I love New York.  New/I-LOC:0.9989    York/I-LOC:0.9974

# fill-mask: guess the masked word
echo 'I <mask> you.' | gft_predict --task fill-mask
# I <mask> you.  salute|0.241    miss|0.177  love|0.147  thank|0.060     applaud|0.047
```

More tasks (2 of 4)

- ✓Text Classification
- ✓Token Classification
- ➢**Fill Mask**
- Text Generation
- Machine Translation

More tasks (3 of 4)

```
# text-classification: sentiment analysis
echo 'I love you.' | gft_predict --task text-classification
# I love you.   POSITIVE    0.9998705387115479

# text-classification: emotion classification
model=H:AdapterHub/bert-base-uncased-pf-emotion
echo 'I love you.' | gft_predict --model $model --task text-classification
# I love you.   love    0.6005669236183167

# token-classification: NER (Named Entity Recognition)
echo 'I love New York.' | gft_predict --task token-classification
# I love New York.  New/I-LOC:0.9989    York/I-LOC:0.9974

# fill-mask: guess the masked word
echo 'I <mask> you.' | gft_predict --task fill-mask
# I <mask> you.  salute|0.241    miss|0.177  love|0.147  thank|0.060     applaud|0.047

# text-generation
echo 'I love ' | gft_predict --task text-generation
# I love you and I will never be forgotten and thank you." I was also
# inspired by all of the students who walked onto campus wearing these
#  teddy I love the idea that you can be anything people ask for you
```

- ✓Text Classification
- ✓Token Classification
- ✓Fill Mask
- ➢**Text Generation**
- Machine Translation

Output is non-deterministic

More tasks (4 of 4)

```
# text-classification: sentiment analysis
echo 'I love you.' | gft_predict --task text-classification
# I love you.   POSITIVE    0.9998705387115479

# text-classification: emotion classification
model=H:AdapterHub/bert-base-uncased-pf-emotion
echo 'I love you.' | gft_predict --model $model --task text-classification
# I love you.   love    0.6005669236183167

# token-classification: NER (Named Entity Recognition)
echo 'I love New York.' | gft_predict --task token-classification
# I love New York.  New/I-LOC:0.9989    York/I-LOC:0.9974

# fill-mask: guess the masked word
echo 'I <mask> you.' | gft_predict --task fill-mask
# I <mask> you.  salute|0.241    miss|0.177  love|0.147  thank|0.060     applaud|0.047

# text-generation
echo 'I love ' | gft_predict --task text-generation
# I love you and I will never be forgotten and thank you." I was also
# inspired by all of the students who walked onto campus wearing these
#  teddy I love the idea that you can be anything people ask for you

# translation
echo 'I love you.' | gft_predict --task translation --model H:Helsinki-NLP/opus-mt-en-fr
# I love you.   Je t'aime.
echo 'I love you.' | gft_predict --task translation --model H:Helsinki-NLP/opus-mt-en-zh
I love you. 我爱你
```

- ✓Text Classification
- ✓Token Classification
- ✓Fill Mask
- ✓Text Generation
- ➢**Machine Translation**

More tasks (4 of 4)

```
# text-classification: sentiment analysis
echo 'I love you.' | gft_predict --task text-classification
# I love you.   POSITIVE    0.9998705387115479

# text-classification: emotion classification
model=H:AdapterHub/bert-base-uncased-pf-emotion
echo 'I love you.' | gft_predict --model $model --task text-classification
# I love you.   love    0.6005669236183167

# token-classification: NER (Named Entity Recognition)
echo 'I love New York.' | gft_predict --task token-classification
# I love New York.  New/I-LOC:0.9989    York/I-LOC:0.9974

# fill-mask: guess the masked word
echo 'I <mask> you.' | gft_predict --task fill-mask
# I <mask> you.  salute|0.241    miss|0.177  love|0.147  thank|0.060     applaud|0.047

# text-generation
echo 'I love ' | gft_predict --task text-generation
# I love you and I will never be forgotten and thank you." I was also
# inspired by all of the students who walked onto campus wearing these
#  teddy I love the idea that you can be anything people ask for you

# translation
echo 'I love you.' | gft_predict --task translation --model H:Helsinki-NLP/opus-mt-en-fr
# I love you.   Je t'aime.
echo 'I love you.' | gft_predict --task translation --model H:Helsinki-NLP/opus-mt-en-zh
I love you. 我爱你
```

- ✓Text Classification
- ✓Token Classification
- ✓Fill Mask
- ✓Text Generation
- ➢**Machine Translation**
  - ➢**English → French**
  - ➢**English → Chinese**

A Model for: English → French

A Model for: English → Chinese

gft_predict –task image-classification

Funny Cat	Cat Chonk
	

Default Model → 1000 Labels

```
url="https://images.all-free-download.com/images/graphicwebp/funny_cat_194619.webp"
echo $url | gft_predict --task H:image-classification
https://images.all-free-download.com/images/graphicwebp/funny_cat_194619.webp   Egyptian cat|0.736  tiger cat|0.039 tabby,
```

The results are more reasonable if we replace the default model with a more appropriate model:

A Model with Cat/Dog Labels

```
url="https://images.all-free-download.com/images/graphicwebp/funny_cat_194619.webp"
echo $url | gft_predict --task H:image-classification --model H:nateraw/vit-base-cats-vs-dogs
https://images.all-free-download.com/images/graphicwebp/funny_cat_194619.webp   cat|1.000   dog|0.000
```

Asirra: a **CAPTCHA** that **exploits interest-aligned manual image categorization**.

J Elson, JR Douceur, J Howell, J Saul - CCS, 2007 - dl.acm.org

History behind Cat/Dog Task

… **Asirra** (Figure 1), a **CAPTCHA** that asks users to identify cats out of a set of 12 photographs of both cats and dogs. **Asirra** is easy … **Asirra**'s **image** database is provided by a novel, mutually …

☆ Save Cite Cited by 581 Related articles All 8 versions

Figure 1: An Asirra challenge. The user selects each of the 12 images that depict cats. As the mouse is hovered over each thumbnail, a larger image and "Adopt me" link appear. "Adopt me" first invalidates the challenge, then takes the user to that animal's page on Petfinder.com.

Figure 6: The differences between cats and dogs are immediately obvious to humans. In many cases, species look similar, with only subtle cues to distinguish them. This makes it a hard vision problem.

Inputs can also come from files (as well as URLs)
There are a few images (and audio files) under $gft/doc/objects

PetImages/0.jpg	PetImages/10000.jpg	PetImages/10001.jpg
		
beans/healthy_test.20.jpg	beans/healthy_test.21.jpg	beans/healthy_test.22.jpg
		

```
model=H:nateraw/vit-base-cats-vs-dogs
ls $gft/doc/objects/images/PetImages/* | sed 3q |
   gft_predict --task H:image-classification --model $model 2>/dev/null
# /mnt/home/kwc/gft7/gft/doc/objects/images/PetImages/0.jpg cat|0.999   dog|0.001
# /mnt/home/kwc/gft7/gft/doc/objects/images/PetImages/10000.jpg cat|1.000   dog|0.000
# /mnt/home/kwc/gft7/gft/doc/objects/images/PetImages/10001.jpg cat|1.000   dog|0.000

model=H:nickmuchi/vit-base-beans
ls $gft/doc/objects/images/beans/* | sed 3q |
   gft_predict --task H:image-classification --model $model 2>/dev/null
# images/beans/healthy_test.20.jpg  healthy|0.996   angular_leaf_spot|0.002 bean_rust|0.002
# images/beans/healthy_test.21.jpg  healthy|0.996   angular_leaf_spot|0.002 bean_rust|0.002
# images/beans/healthy_test.22.jpg  healthy|0.996   angular_leaf_spot|0.002 bean_rust|0.002
```

Task: Speech Recognition (ASR, ctc)

Input from stdin

```
cd $gft/doc/objects/wav/TIMIT;
ls *.WAV | sed 3q | gft_predict --task ASR 2>/dev/null
```

filename	prediction (yhat)
SA1.WAV	SHE HAD YOUR DARK SUIT AND GREASY WASHWATER ALL YEAR
SA2.WAV	DON'T ASK ME TO CARRY AN OILY RAG LIKE THAT
SI1129.WAV	THIS GROUP IS SECULARIST AND THEIR PROGRAMM TENDS TO BE TECHNOLOGICAL

Input from dataset

```
gft_predict --eqn 'ASR:text~file' \
        --data H:timit_asr \
        --split test 2>/dev/null |
    awk -F/ '{print $NF}' | sed 3q
```

filename	gold	prediction (yhat)
SX139.WAV	The bungalow was pleasantly situated near the shore.	THE BUNGALOW WAS PLEASANTLY SITUATED NEAR THE SHORE
SA2.WAV	Don't ask me to carry an oily rag like that.	DON'T ASK ME TO CARRY AN OILY RAG LIKE THAT
SX229.WAV	Are you looking for employment?	ARE YOU LOOKING FOR EMPLOYMENT

Wrapping up Part A

- ✓Two Questions
  - ✓How do I find the *good* stuff?
  - ✓And how do I use the *good* stuff?
    - ✓A: gft_predict
- ➢**Cheat Sheet**
- Guide to Hundreds of Examples
- Motivations for GFT
- Why change terminology?
- Part B

gft Cheat Sheet

Step	gft Support	Description	Time	Hardware
1		Pre-Training	Days/Weeks	Large GPU Cluster
2	*gft_fit*	Fine-Tuning	Hours/Days	1+ GPUs
3	*gft_predict*	Inference	Seconds/Minutes	0+ GPUs

4 Functions

- gft_fit: $f_{pre} \rightarrow f_{post}$ (fine-tuning)
  - 4 Arguments, --output_dir, --metric, --splits
  - (plus most args in most hubs)
- gft_predict: $f(x) \rightarrow \hat{y}$ (inference)
  - Input: 4 Arguments (x from data or stdin)
  - Output: $\hat{y}$ for each x
- gft_eval: Score model on dataset
  - Input: 4 Arguments, --split, --metric, …
  - Output: Score
- gft_summary: Find good stuff
  - 4 Arguments
  - (may include: __*contains*__, __*infer*__)

4 Arguments

- --data
- --model
- --eqn $task: y \sim x_1 + x_2$
- --task
  1. classify (text-classification)
  2. classify_tokens (token-classification)
  3. classify_spans (QA, question-answering)
  4. classify_audio (audio-classification)
  5. classify_images (image-classification)
  6. regress
  7. text-generation
  8. MT (translation)
  9. ASR (ctc, automatic-speech-recognition)
  10. fill-mask

Guide to Hundreds of Examples

```
(gft-adapters) kwc@asimov-7:~/gft7/gft/examples$ find . -name '*sh' | sed 50q
./eval_examples/model.HuggingFace/language/data.HuggingFace/Yuchen_models/glue/mrpc.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/Yuchen_models/glue/sst2.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/Yuchen_models/glue/qnli.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/Yuchen_models/glue/rte.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/Yuchen_models/glue/stsb.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/paraphrase/paws_labeled_final.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/paraphrase/paws_unlabeled_final.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/paraphrase/paws_labeled_swap.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/text_classification/banking77.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/text_classification/snli.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/text_classification/ag_news.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/text_classification/tweets_hate_speech_detection.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/text_classification/emotion.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/cola.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/mrpc.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/sst2.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/mnli_mismatched.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/qnli.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/qqp.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/rte.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/mnli_matched.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/stsb.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/cola.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/anli_r3.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/mrpc.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/scicite.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/sst2.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/duorc_s.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/winogrande.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/art.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/drop.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/conll2000.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/sick.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/comqa.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/ud_pos.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/mit_movie_trivia.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/squad.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/snli.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/wnut_17.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/wikihop.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/copa.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/emo.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/pmb_sem_tagging.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/boolq.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/rotten_tomatoes.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/quoref.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/trec.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/conll2003_pos.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/wic.sh
./eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/AdapterHub/hotpotqa.sh
```

Naming Conventions

- Models
  - model.HuggingFace/model.Paddle
  - our_models/their_models
- Data
  - data.HuggingFace/data.Paddle

Cola subset of Glue, with pretrained models and data from HuggingFace

$gft/examples/eval_examples/model.HuggingFace/language/data.HuggingFace/their_models/glue/cola.sh

Input

```
for model in textattack/distilbert-base-uncased-CoLA textattack/bert-base-uncased-CoLA textattack/distilbert-base-cased-CoLA textattack/roberta-base-CoLA cointegrated/roberta-large-cola-krishna2020 textattack/albert-base-v2-CoLA gchhablani/bert-base-cased-finetuned-cola yoshitomo-matsubara/bert-base-uncased-cola howey/roberta-large-cola textattack/xlnet-base-cased-CoLA isakbos/Q8BERT_COLA_L_512 EhsanAghazadeh/bert-large-uncased-CoLA_A EhsanAghazadeh/bert-large-uncased-CoLA_B 09panesara/distilbert-base-uncased-finetuned-cola yoshitomo-matsubara/bert-large-uncased-cola  textattack/xlnet-large-cased-CoLA Alireza1044/albert-base-v2-cola pmthangk09/bert-base-uncased-glue-cola kamivao/autonlp-cola_gram-208681 mrm8488/deberta-v3-small-finetuned-cola 123abhiALFLKFO/distilbert-base-uncased-finetuned-cola Ahren09/distilbert-base-uncased-finetuned-cola

do

gft_eval --data H:glue,cola  --split val  \

  --eqn 'classify: label ~ sentence'  --model H:$model \

 --metric H:glue,cola  --figure_of_merit matthews_correlation

done
```

Popular models containing "cola" (not SOTA)

Output (Sorted by score; 4 errors removed)

Model	Score	Seconds
howey/roberta-large-cola	0.65	3.93
textattack/roberta-base-CoLA	0.64	1.6
yoshitomo-matsubara/bert-large-uncased-cola	0.63	5.12
mrm8488/deberta-v3-small-finetuned-cola	0.63	1.62
EhsanAghazadeh/bert-large-uncased-CoLA_A	0.61	4.05
yoshitomo-matsubara/bert-base-uncased-cola	0.61	1.58
EhsanAghazadeh/bert-large-uncased-CoLA_B	0.60	4.13
gchhablani/bert-base-cased-finetuned-cola	0.60	1.57
pmthangk09/bert-base-uncased-glue-cola	0.58	1.53
textattack/albert-base-v2-CoLA	0.58	1.59
textattack/distilbert-base-uncased-CoLA	0.57	1.04
kamivao/autonlp-cola_gram-208681	0.56	1.52
Alireza1044/albert-base-v2-cola	0.55	1.71
09panesara/distilbert-base-uncased-finetuned-cola	0.54	0.87
textattack/bert-base-uncased-CoLA	0.53	2.35
123abhiALFLKFO/distilbert-base-uncased-finetuned-cola	0.53	0.9
textattack/xlnet-base-cased-CoLA	0.50	2.23
textattack/distilbert-base-cased-CoLA	0.46	0.88
isakbos/Q8BERT_COLA_L_512	0.00	1.77
textattack/xlnet-large-cased-CoLA	0.00	5.08
cointegrated/roberta-large-cola-krishna2020	-0.65	4.31

Wrapping up Part A

- ✓Two Questions
  - ✓How do I find the *good* stuff?
  - ✓And how do I use the *good* stuff?
    - ✓A: gft_predict
- ✓**Cheat Sheet**
- ✓Guide to Hundreds of Examples
- ➢**Motivations for GFT**
- Why change terminology?
- Part B

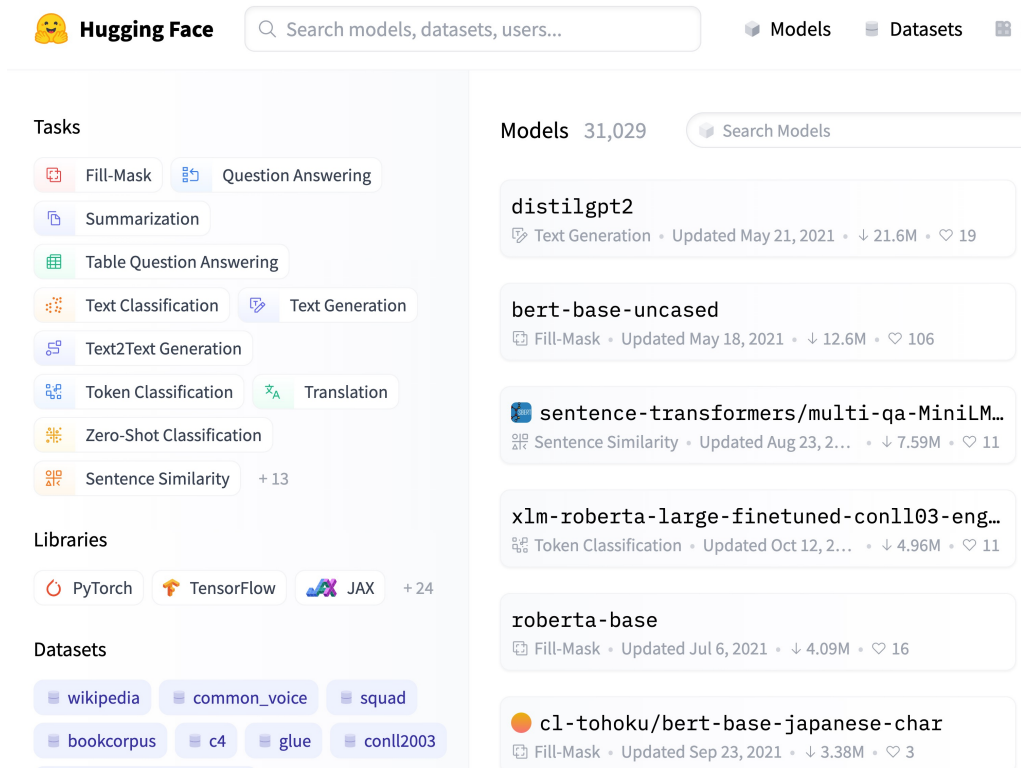
Hubs: Data, Models, Examples (with $\overset{too\ much}{\wedge}$ code)

Embarrassment of Riches: Overwhelming

Goal for *gft*: Accessibility

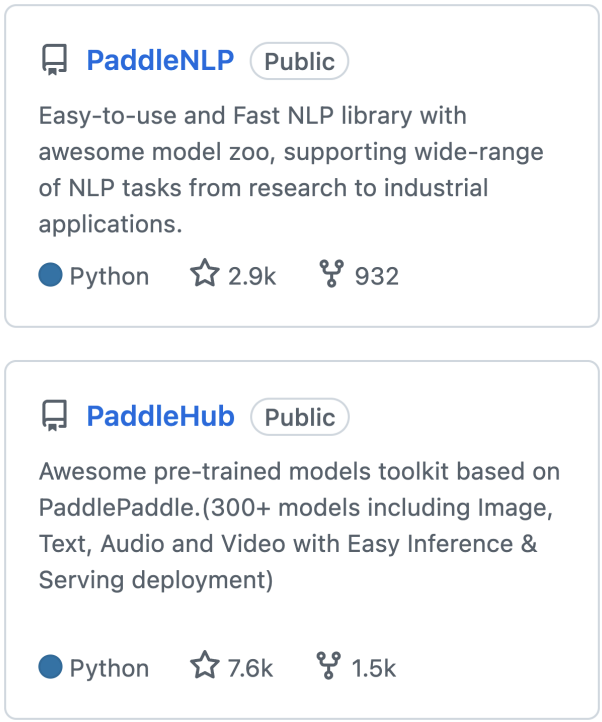
HuggingFace

https://huggingface.co/



PaddleNLP and PaddleHub

https://github.com/PaddlePaddle



Advantages of Higher-Level Little Languages

- Ease of use
  - Accessible to broader audience
  - Deep Nets for Poets (see next slide)
- Hide complexity
  - 1-line of gft ≪ 800+ lines of python
  - More transparency; fewer bugs
- Avoid special cases
  - Standard examples on hubs
    - expect users to modify python
    - different programs for different datasets/models/tasks
  - gft encourages code re-use
    - gft uses similar code/algorithms as examples
    - with similar performance and computational costs
    - but generalizes over more datasets/models/tasks
- Portability across suppliers
  - Mix and Match Across Hubs
    - (HuggingFace, PaddleNLP)

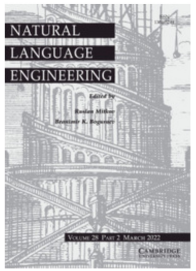
(Sensible) Priority for Startups

Hub Strategy: Usage >> SOTA

(Mistaken) Priority Research

- Startup Strategy:
  - Get big quick
- SOTA is like sprinting (hard work)
  - Avoid temptation to sprint to front
    - (except when it is worth the effort)
  - Hubs spend most of the race
    - in the peloton drafting researchers
  - Make it easy for 3rd parties
    - to post their best models on your hub
- Use of 3rd parties → Wide range in quality
  - Best are amazing: best in class, industrial strength…
  - But some don't work well, and may not even load…
  - Help users find good stuff

Unix for Poets → Deep Nets for Poets



Natural Language Engineering

Emerging trends: A gentle introduction to fine-tuning

Published online by Cambridge University Press: **26 October 2021**

Kenneth Ward Church, Zeyu Chen and Yanjun Ma

Show author details

Article | Figures | Metrics

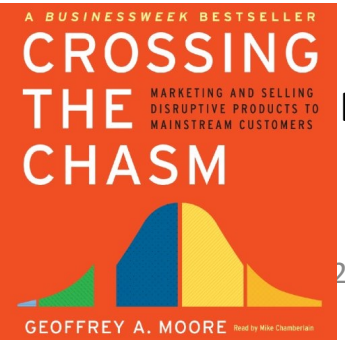
Save PDF | Share | Cite | Rights & Permissions

This tutorial → Community-building

- Unix for Poets: Slides from 1990s
  - When empiricism was just taking off
  - Too much low hanging fruit to pick ourselves
- Revival of 1950s Empiricism in 1990s
  - Community-building exercises
    - Unix for Poets
    - EMNLP (Empirical Methods in NLP)
    - Linguistic Data Consortium (LDC)
- Deep nets in 2020s
  - Community-building exercises
    - Deep Nets for Poets
    - (target audience includes non-programmers)

Deep Nets will benefit

- if we can increase demand for what we do

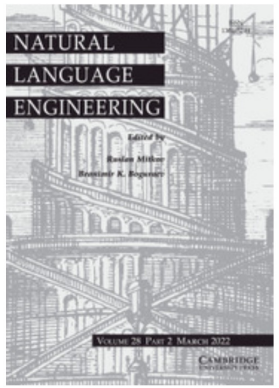


Slides for a Linguistic Summer School

Unix™ for Poets

Kenneth Ward Church
AT&T Bell Laboratories
kwc@research.att.com

- Text is available like never before
  - Dictionaries, corpora, etc.
  - Data Collection Efforts:
    ACL/DCI, BNC, CLR, ECI, EDR, ICAME, LDC



Natural Language Engineering

Emerging trends: Deep nets for poets

Published online by Cambridge University Press: **01 September 2021**

Kenneth Ward Church, Xiaopeng Yuan, Sheng Guo, Zewu Wu, Yehua Yang and Zeyu Chen

Article | Figures | Metrics

Save PDF | Share | Cite | Rights & Permissions

Wrapping up Part A

- ✓Two Questions
  - ✓How do I find the *good* stuff?
  - ✓And how do I use the *good* stuff?
    - ✓A: gft_predict
- ✓**Cheat Sheet**
- ✓Guide to Hundreds of Examples
- ✓Motivations for GFT
- ➢**Why change terminology?**
- Part B

Why Change Terminology?

More opportunity for academics & Computational Linguistics

~~Standard Terminology~~	Proposed Alternat
~~Base/Foundation/Pre-Trained~~	f_{pre}
~~Fine-Tuning~~	*fit*
~~Inference~~	*predict*

Not this

Emphasize this

Industry has an unfair advantage over academia

- Pre-training requires $$$$, Big Data & Logistics:
- Big Investments → Risk Adverse → Less Creativity

https://www.youtube.com/watch?v=dG628PEN1fY

VLSI History

Perspective from MIT and beyond:
IBM, Silicon Valley, Mich, LGBTQ, etc.

http://www.eecs.mit.edu/docs/newsletter/VLSI.pdf
http://ai.eecs.umich.edu/people/conway/Memoirs/MIT/MIT_Reminiscences.pdf

The VLSI revolution at MIT *by Paul Penfield, Jr.*

Remember the VLSI revolution? If you are over 50, you might. If not, you have probably heard about it. In the late 1970s the VLSI (Very Large Scale Integration) revolution opened up the design of integrated circuits to people who did not know anything about device physics or semiconductor processing.

This was first demonstrated here at MIT. Below is the story of the two people, Lynn Conway (photo lower right) and Professor Jonathan Allen (photo right), who made it happen. Ideally, these two should tell the story, and Conway has written a compelling account from her perspective, "MIT Reminiscences: Student years to VLSI revolution", http://ai.eecs.umich.edu/people/conway/Memoirs/MIT/MIT_Reminiscences.pdf. Sadly, Allen died in 2000 but I will try to tell the story from the MIT EECS perspective here. Please read both accounts.

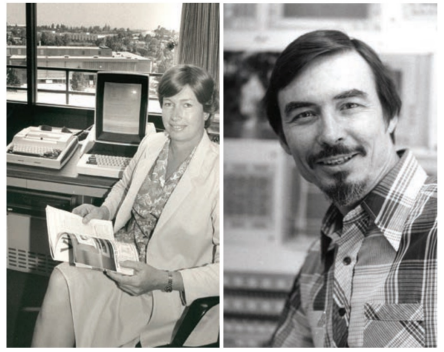
Although in the 1950s and 1960s our department (named EE at the time) was innovative in using physical device models to teach electronics, we consciously decided not to expand research in silicon devices and circuits because we thought industry could do it better.

The rise of computer science made us rethink that decision. In the early 1970s we decided not to split into separate EE and CS departments but stay united, under the new name EECS. Then we realized that our research in computer architecture and digital systems was hindered: testing new ideas in the form of hardware required LSI (large-scale integration) that was available only in industry. Allen knew this both from his own research in speech technology and through serving as Associate Director of the Research Laboratory of Electronics. He wanted fabrication as a service, not something he and other digital hardware designers would have to know how to do.

Meanwhile a revolution was brewing in California. Professor Carver Mead at Caltech had used industry connections to get student projects fabricated and the students loved it. A group centered around Lynn Conway at Xerox PARC (Palo Alto Research Center) set out to make designing integrated systems so simple that lots more people could do it. Both Stanford and Berkeley (who had continued their research in silicon devices) were interested.

In 1977 we decided we had to get on board. Several decisions followed. To keep true to our decision to stay as one department, we needed relevant research in both EE and CS. That meant: a silicon fabrication facility where research on devices and processes was informed by the needs of digital systems;

Professor Jonathan Allen was the sixth Director of the Research Laboratory of Electronics, RLE, and a member of the EECS faculty since 1968. Photo courtesy of RLE.



Lynn Conway (left) in her office at Xerox PARC (1983). On Lynn's desk are the Alto computer she used to write the VLSI textbook, and the TI terminal she used to interact with PARC while at MIT. Photo by Margaret Moulton, courtesy Lynn Conway

- Think of f_{pre} like Intel CPU chips
- Universities can afford
  - to program CPUs and models (*fit/predict*)
  - but not VLSI fabrication (f_{pre})
- Jon Allen (my thesis advisor at MIT)
  - Invested big $$$$ in VLSI fabrication
  - Big $$$$ → (Too) Much Responsibility
    - Creativity requires willingness to take risks
- Industry can afford to work on projects
  - with industrial challenges/rewards:
    - Capital, ROI, Scale, Logistics
  - Academics have other priorities:
    - Bottom line: $fit/predict \gg f_{pre}$

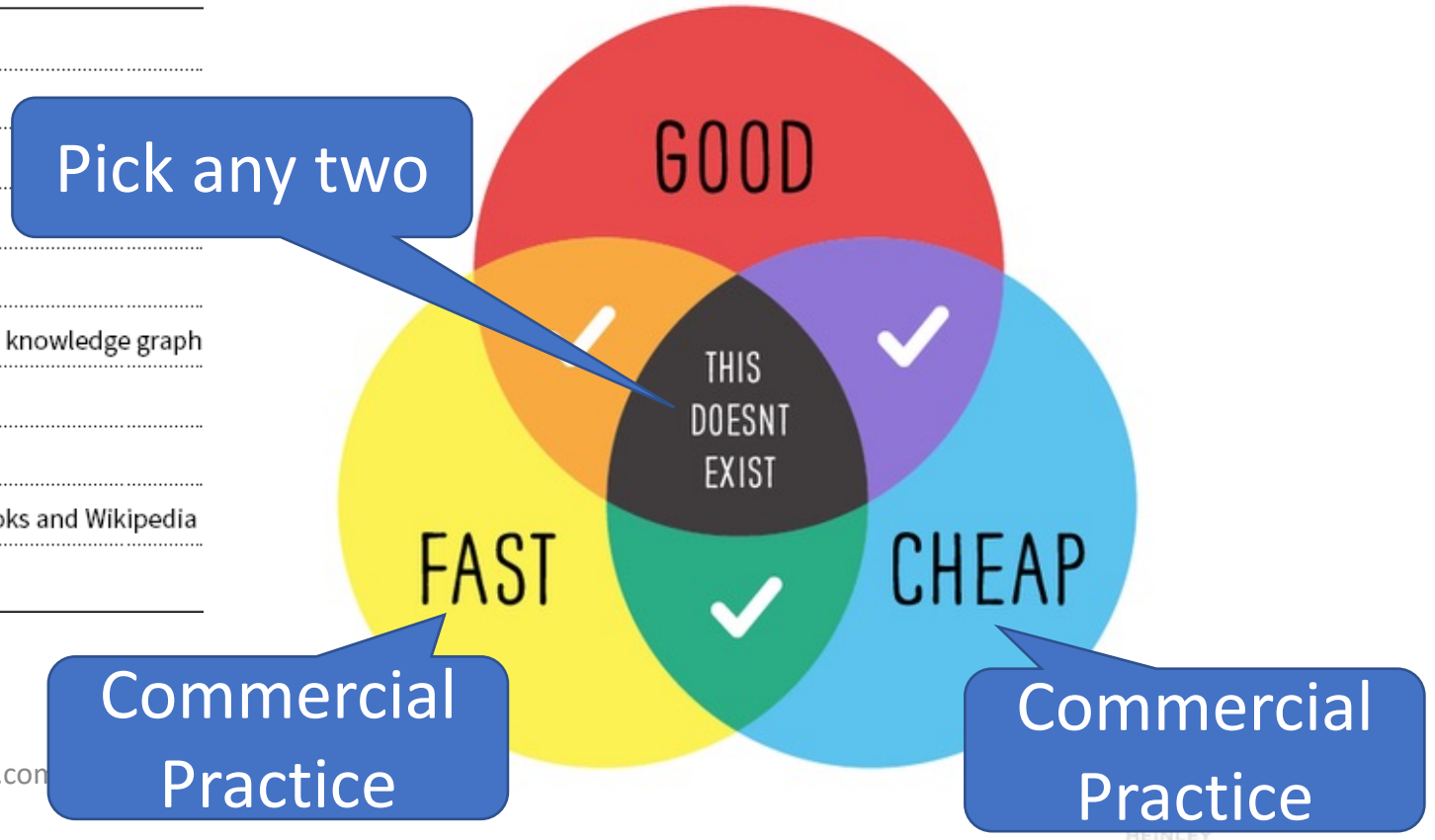
Model Compression

Popular Choices on Hubs

Pre-trained (foundation) model	Parameters	Training data
ResNet-50 (He *et al.* 2016)	23M	14M images from ImageNet
VT (Wu *et al.* 2020)	11.7–21.9M	14M images from ImageNet
Wav2vec (Baevski *et al.* 2020)	95–317M	960 hours from LibriSpeech
BERT (Devlin *et al.* 2019)	110–340M	3.3B words from Books/Wiki
ERNIE 2.0 (Sun *et al.* 2020)	110–340M	7.9B en + 15B zh tokens
ERNIE 3.0 (Sun *et al.* 2021)	10B	375B tokens of text, as well as knowledge graph
RoBERTa (Liu *et al.* 2019)	110M	160GBs of text
GPT-2 (Radford *et al.* 2019)	1.5B	40GBs of text
GPT-3 (Brown *et al.* 2020)	125M–175B	1TB from Common Crawl, Books and Wikipedia
XLM-RoBERTa (Conneau *et al.* 2020)	12–16B	2.5TBs from many languages

Model Compression:

Trade-offs: Space, Time, Accuracy



Bottleneck for Global Warming:
$cost(transportation) \gg 5\ cars$
$cost(smart\ speakers) \gg 5\ cars$

Commercial Practice:

$cost(inference) \gg cost(training)$

- Variable Costs: Recurring costs
  - Inference Costs ≫ 5 cars
    - Deploy a model to 1B people
    - Who use it every day (for years)
- If the application goes viral (success)
  - $variable\ costs \gg fixed\ costs$
  - $cost(inference) \gg cost(training)$

Wrapping up Part A: Glass is Half Full
Amazing how much can be done with so little (*gft*)

- ✓Two Questions
  - ✓How do I find the *good* stuff?
  - ✓And how do I use the *good* stuff?
    - ✓A: gft_predict
- ✓Cheat Sheet
- ✓Guide to Hundreds of Examples
- ✓Motivations for GFT
- ✓Why change terminology?
- ➢**Part B: Glass is Half Empty**
  - ➢**No one would suggest that fit/predict are AI**