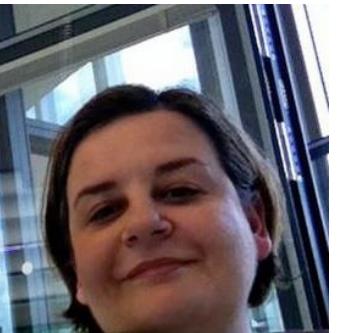




Ken



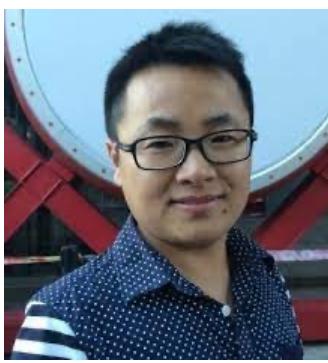
Valia



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

# Outline

- Part A: Glass is half-full
    - Deep nets can do much
  - Part B: Glass is half-empty
    - There is always more work to do
- 
- Make deep nets accessible to masses  
(including non-programmers)
- Advocate combo of Part A with
1. AI Knowledge Representation
  2. Linguistics and Philosophy
- GFT Code,
  - 100s of examples,
  - slides, videos, papers

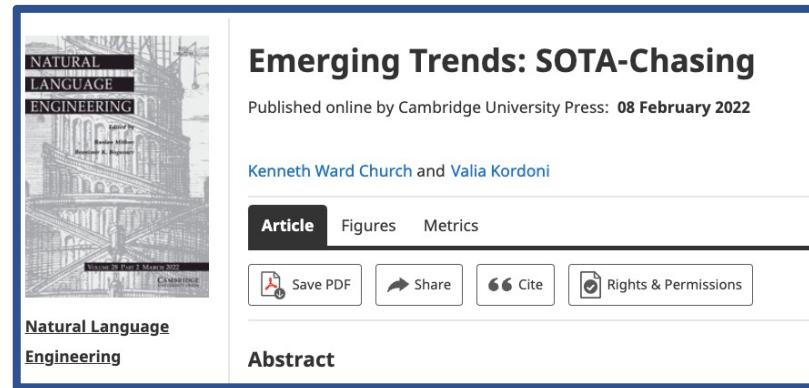
# Agenda: Part A

## ➤ **Strategy**

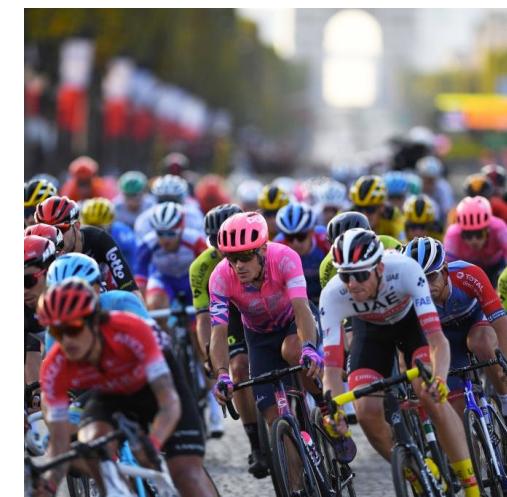
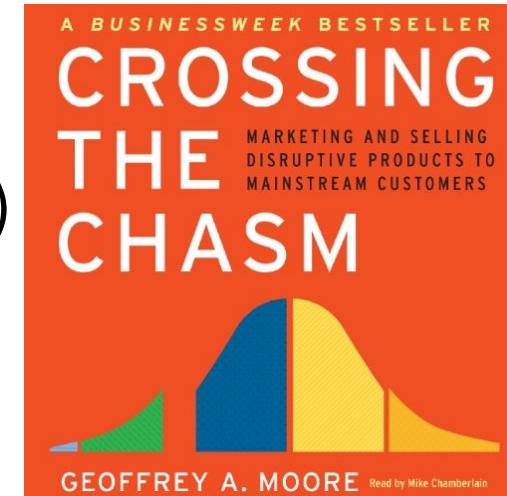
- *gft* Cheatsheet
    - 4-5 Functions + 4 Arguments
    - Amazing how much can be done with so little
  - Seven Simple Examples
- More Tasks: MT, ASR, Vision
  - Two Questions
    - How do we find the good stuff?
    - How do we use the good stuff?
  - Why de-emphasize pre-training?
  - Conclusions

# Strategy: Inclusiveness

- Get your priorities right: Inclusiveness >> Exclusiveness
  - SOTA-Chasing       (Exclusive: mine is bigger than yours)
  - Crossing the Chasm       (Inclusive: Appeal to mainstream users)
  - Hubs: Get big quick       (Inclusive: Usage >> SOTA)

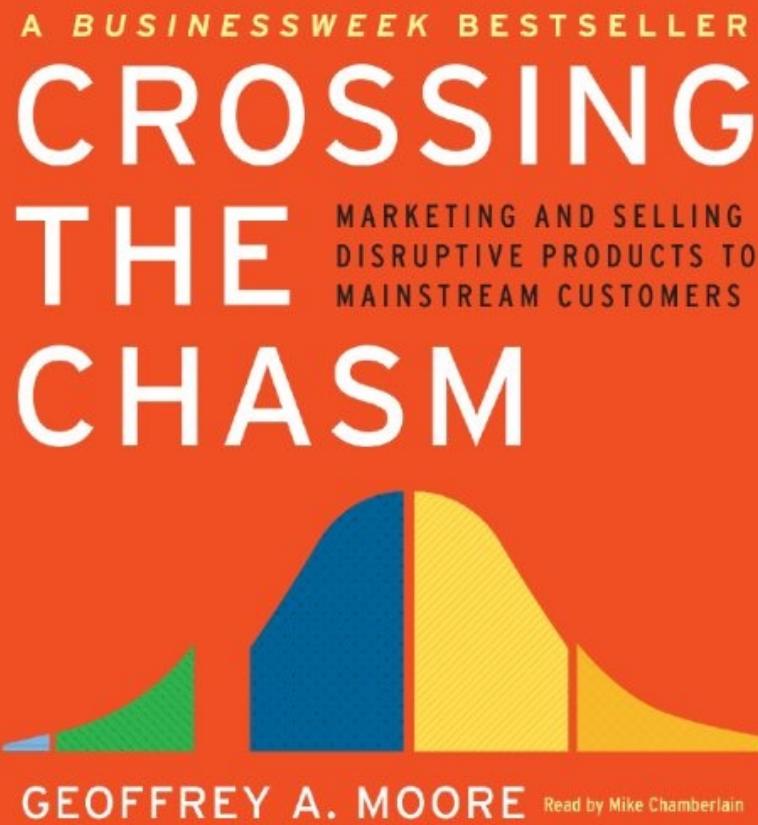
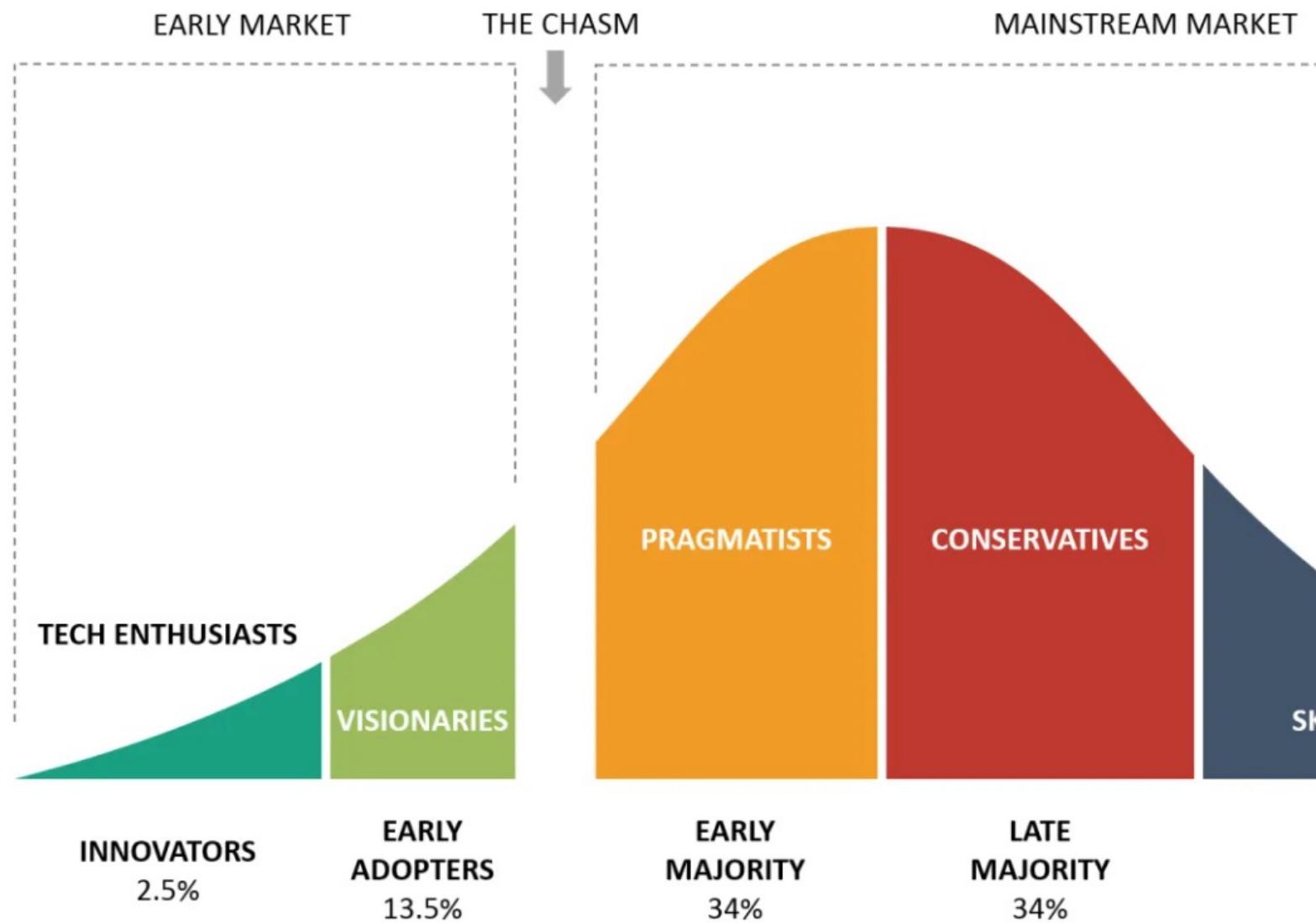


The screenshot shows a digital journal cover for 'Natural Language Engineering'. The title 'NATURAL LANGUAGE ENGINEERING' is at the top, followed by 'Volume 29 Part 1 March 2022'. Below the title is a black and white photograph of a classical building facade. At the bottom, it says 'Natural Language Engineering'. To the right of the cover, there's a header with 'Emerging Trends: SOTA-Chasing' and 'Published online by Cambridge University Press: 08 February 2022'. Below the header are sections for 'Article', 'Figures', and 'Metrics'. At the bottom are buttons for 'Save PDF', 'Share', 'Cite', and 'Rights & Permissions'.



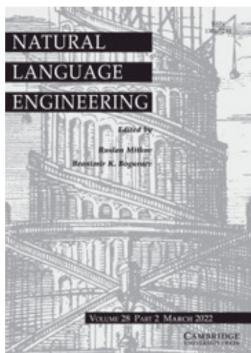
Engineers

Bigger Opportunities:  
(I)users (aka poets)



# Poets

- Poets is an imperfect metaphor,
  - intended as a gesture toward inclusion.
- The future for deep nets
  - will benefit by reaching out to
    - a broad audience of potential users,
  - including people with
    - little or no programming skills,
    - and little interest in training models.
- Lessons from success of Unix
  - Less is more
  - Inclusion: Appeal to power users as well as masses
  - Portability: Inclusiveness++



## 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

## 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
  - Information Super Highway Roadkill:  
email, bboards, faxes
  - Billions and billions of words
- What can we do with it all?
- It is better to do something simple,  
than nothing at all.
- You can do the simple things yourself  
(DIY is more satisfying than begging for “help” from a computer officer.)

(Sensible) Priority for Startups

# Hub Strategy: Usage >> SOTA

- 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 3<sup>rd</sup> parties
    - to post their best models on your hub
- Use of 3<sup>rd</sup> 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

(Mistaken) Priority Research



0-shot

# Terminology

Unsupervised

Supervised

Few-shot

## Standard Terminology

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

~~Fine-Tuning~~

~~Inference~~

## Proposed Alternatives

$f_{pre}$

$fit$

$predict$

Emphasize this

De-emphasize this

Teaser: Stay-tuned

# 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	<i>gft_fit</i>	Fine-Tuning	Hours/Days	1+ GPUs
3	<i>gft_predict</i>	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



# Little Languages (from Unix)

<https://comp590-19s.github.io/>

Kris Jordan (UNC Chapel Hill, 2019)

Little Languages - 19S

Assignments Lectures Resources



Computer scientists and software engineers depend on tools that interpret Little Languages (Bentley, UNC '76) frequently in their day-to-day work: command-line shells, project build system configurations, regular expressions, utility programs, documentation markup languages, modal text editors, and so on. As these languages are numerous and domain-specific, our focus will be on their first principles, common challenges, implementations, and limitations.

This course will explore the interpretation and applications of little languages through case studies on standard programming environment tools with historical significance, such as `grep`, `make`, `bash`, `vim`, and more. This course highlights important ideas in computational theory through direct, pragmatic applications in text pattern matching and structured language processing.

You will gain experience working in a systems language appropriate for implementing command-line tools with their own little languages. You will gain comfort working in a Unix-like programming environment and knowledge of its standard tooling and utilities. You will learn to make use of a Unix-like operating system's facilities and APIs for file system input/output, process management, and memory management.

In this course, you will:

1. Implement the essences of practical, historically significant Unix utilities, each with its own little language.
  - `dc` (1969) a calculator whose simple, reverse-polish notation language commands a virtual stack machine. It is the oldest little language in Unix.
  - `bc` (1975), the successor to `dc`, is an infix-based context free language needing a pushdown automata equivalent to parse. It will be implemented, as it was historically, as a frontend translator for `dc`.
  - `grep` (1974) is a regular expression pattern matching tool still widely used today. Its implementation will construct and simulate a non-deterministic finite state machine.
  - `sh` (1977) the interpretation of Bourne's shell, or command-line interpreter, introduces fundamental operating systems concepts.

## Bentley (CACM-1986)

by Jon Bentley



### LITTLE LANGUAGES

When you say "language," most programmers think of the big ones, like FORTRAN or COBOL or Pascal. In fact, a language is any mechanism to express intent, and the input to many programs can be viewed profitably as statements in a language. This column is about those "little languages."

Programmers deal with microscopic languages every day. Consider printing a floating-point number in six characters, including a decimal point and two subsequent digits. Rather than writing a subroutine for the task, a FORTRAN programmer specifies the format `F6.2`, and a COBOL programmer defines the picture `999.99`. Each of these descriptions is a statement in a well-defined little language. While the languages are quite different, each is appropriate for its problem domain: although a FORTRAN programmer might complain that `999999.99999` is too long when `F12.5` could do the job, the COBOLer can't even express in FORTRAN such common financial patterns as `$, $$, $$.99`. FORTRAN is aimed at scientific computing. COBOL is designed for business.

In the good old days, real programmers would swagger to a key punch and, standing the whole time, crank out nine cards like:

```
//SUMMARY JOB REGION=(100K,50K)
// EXEC PGM=SUMMAR
//SYSIN DD DSNAME=REP.8601,DISP=OLD,
//          UNIT=2314,SPACE=(TRK,(1,1,1)),
//          VOLUME=SER=577632
//SYSOUT DD DSNAME=SUM.8601,DISP=(,KEEP),
//          UNIT=2314,SPACE=(TRK,(1,1,1)),
//          VOLUME=SER=577632
//SYSABEND DD SYSOUT=A
```

Today's young whippersnappers do this simple job by typing

```
summarize <jan.report> jan.summary
```

Modern successors to the old "job control" languages are not only more convenient to use, they are fundamentally more powerful than their predecessors. In

© 1986 ACM 0001-0782/86/0800-0711 75¢

# programming pearls

the June column, for instance, Doug McIlroy implemented a program to find the *K* most common words in a document in six lines of the UNIX® SHELL language.

Languages surround programmers, yet many programmers don't exploit linguistic insights. Examining programs under a linguistic light can give you a better understanding of the tools you now use, and can teach you design principles for building elegant interfaces to your future programs. This column will show how the user interfaces to half a dozen interesting programs can be viewed as little languages.

This column is built around Brian Kernighan's PIC language for making line drawings. Its compiler is implemented on the UNIX system, which is particularly supportive (and exploitative) of language processing; the sidebar on pages 714–715 shows how little languages can be implemented in a more primitive computing environment (BASIC on a personal computer).

The next section introduces PIC and the following section compares it to alternative systems. Subsequent sections discuss little languages that compile into PIC and little languages used to build PIC.

### The PIC Language

If you're talking about compilers, you might want to depict their behavior with a picture:

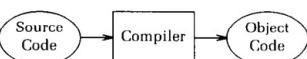
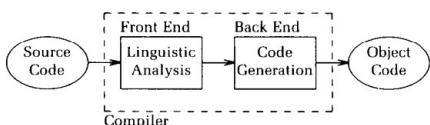


FIGURE 1. A Simple View of a Compiler

(This diagram is genuine PIC output; we'll see its input description shortly.) Or you may desire a little more detail about the internal structure:



UNIX is a trademark of AT&T Bell Laboratories.



# Little Languages (from Unix)

In the good old days, real programmers would swagger to a key punch and, standing the whole time, crank out nine cards like:

```
//SUMMARY JOB REGION=( 100K,50K)
//
//SYSIN    EXEC PGM=SUMMAR
          DD   DSNAME=REP.8601,DISP=OLD,
                UNIT=2314,SPACE=(TRK,(1,1,1)),
                VOLUME=SER=577632
//
//SYSOUT   DD   DSNAME=SUM.8601,DISP=(,KEEP),
          UNIT=2314,SPACE=(TRK,(1,1,1)),
          VOLUME=SER=577632
//
//SYSABEND DD   SYSOUT=A
```

Today's young whippersnappers do this simple job by typing

```
summarize <jan.report> jan.summary
```

## Bentley (CACM-1986)

by Jon Bentley

### LITTLE LANGUAGES

When you say "language," most programmers think of the big ones, like FORTRAN or COBOL or Pascal. In fact, a language is any mechanism to express intent, and the input to many programs can be viewed profitably as statements in a language. This column is about those "little languages."

Programmers deal with microscopic languages every day. Consider printing a floating-point number in six characters, including a decimal point and two subsequent digits. Rather than writing a subroutine for the task, a FORTRAN programmer specifies the format F6.2, and a COBOL programmer defines the picture 999.99. Each of these descriptions is a statement in a well-defined little language. While the languages are quite different, each is appropriate for its problem domain: although a FORTRAN programmer might complain that 999999.99999 is too long when F12.5 could do the job, the COBOLer can't even express in FORTRAN such common financial patterns as \$, \$\$, \$\$.99. FORTRAN is aimed at scientific computing. COBOL is designed for business.

In the good old days, real programmers would swagger to a key punch and, standing the whole time, crank out nine cards like:

```
//SUMMARY JOB REGION=(100K,50K)
//
//SYSIN    EXEC PGM=SUMMAR
          DD   DSNAME=REP.8601,DISP=OLD,
                UNIT=2314,SPACE=(TRK,(1,1,1)),
                VOLUME=SER=577632
//
//SYSOUT   DD   DSNAME=SUM.8601,DISP=(,KEEP),
          UNIT=2314,SPACE=(TRK,(1,1,1)),
          VOLUME=SER=577632
//
//SYSABEND DD   SYSOUT=A
```

Today's young whippersnappers do this simple job by typing

```
summarize <jan.report> jan.summary
```

Modern successors to the old "job control" languages are not only more convenient to use, they are fundamentally more powerful than their predecessors. In

© 1986 ACM 0001-0782/86/0800-0711 75¢

# programming pearls

the June column, for instance, Doug McIlroy implemented a program to find the K most common words in a document in six lines of the UNIX® SHELL language.

Languages surround programmers, yet many programmers don't exploit linguistic insights. Examining programs under a linguistic light can give you a better understanding of the tools you now use, and can teach you design principles for building elegant interfaces to your future programs. This column will show how the user interfaces to half a dozen interesting programs can be viewed as little languages.

This column is built around Brian Kernighan's PIC language for making line drawings. Its compiler is implemented on the UNIX system, which is particularly supportive (and exploitative) of language processing; the sidebar on pages 714–715 shows how little languages can be implemented in a more primitive computing environment (BASIC on a personal computer).

The next section introduces PIC and the following section compares it to alternative systems. Subsequent sections discuss little languages that compile into PIC and little languages used to build PIC.

#### The PIC Language

If you're talking about compilers, you might want to depict their behavior with a picture:

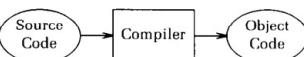
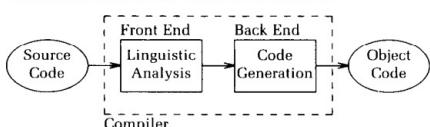


FIGURE 1. A Simple View of a Compiler

(This diagram is genuine PIC output; we'll see its input description shortly.) Or you may desire a little more detail about the internal structure:



UNIX is a trademark of AT&T Bell Laboratories.

# We propose *gft*, a Little Language

In the good old days, real programmers would swagger to a key punch and, standing the whole time, crank out nine cards like:

```
//SUMMARY   JOB   REGION=( 100K,50K)
//           EXEC  PGM=SUMMAR
//SYSIN      DD    DSNAME=REP.8601,DISP=OLD,
//                         UNIT=2314,SPACE=(TRK,( 1,1,1)),
//                         VOLUME=SER=577632
//SYSOUT     DD    DSNAME=SUM.8601,DISP=(,KEEP),
//                         UNIT=2314,SPACE=(TRK,( 1,1,1)),
//                         VOLUME=SER=577632
//SYSABEND  DD    SYSOUT=A
```

Today's young whippersnappers do this simple job by typing

```
summarize <jan.report>jan.summary
```

- In the good old days,
  - real programmers would copy a few hundred lines of PyTorch from HuggingFace
  - and modify as necessary to change
    - models,
    - datasets,
    - task,
    - etc.
- With *gft*, 100s of lines → 1-line
  - More accessible to masses
  - (including non-programmers)

# *gft* Cheat Sheet (General Fine-Tuning)

Amazing how much can be  
done with so little

## 4+1 Functions

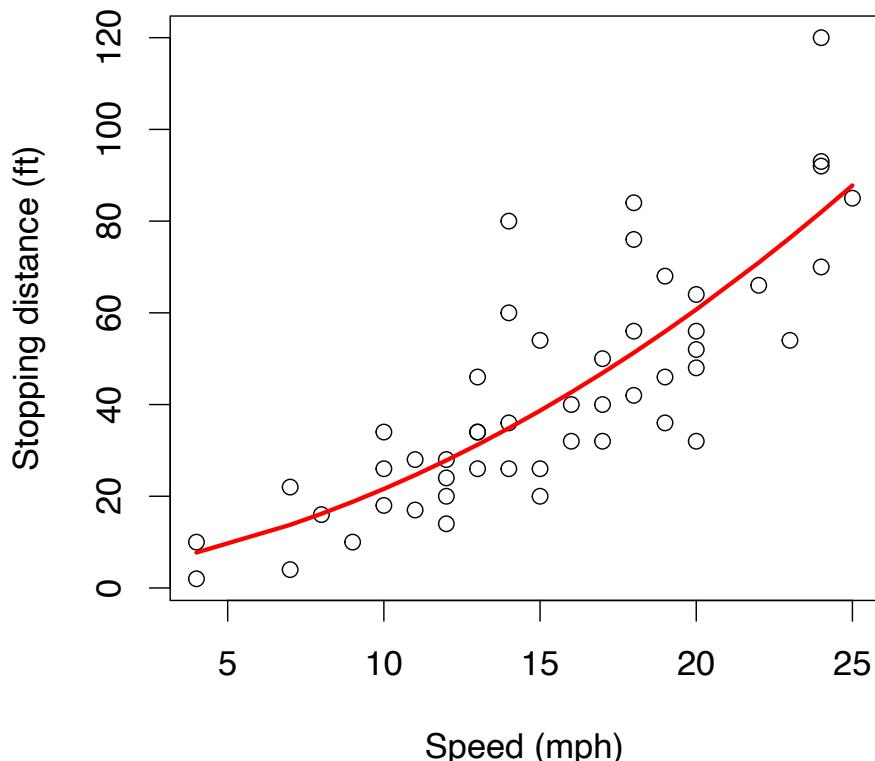
1. **gft\_fit:**  $f_{pre} \rightarrow f_{post}$  (fine-tuning)
  - 4 Arguments, --output\_dir, --metric, --splits
  - (plus most args in most hubs)
2. **gft\_predict:**  $f(x) \rightarrow \hat{y}$  (inference)
  - Input: 4 Arguments ( $x$  from data or stdin)
  - Output:  $\hat{y}$  for each  $x$
3. **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

*gft* (General Fine-Tuning) is based on  
*glm* (General Linear Models) in R

Make deep nets look like regression  
(Not much programming)



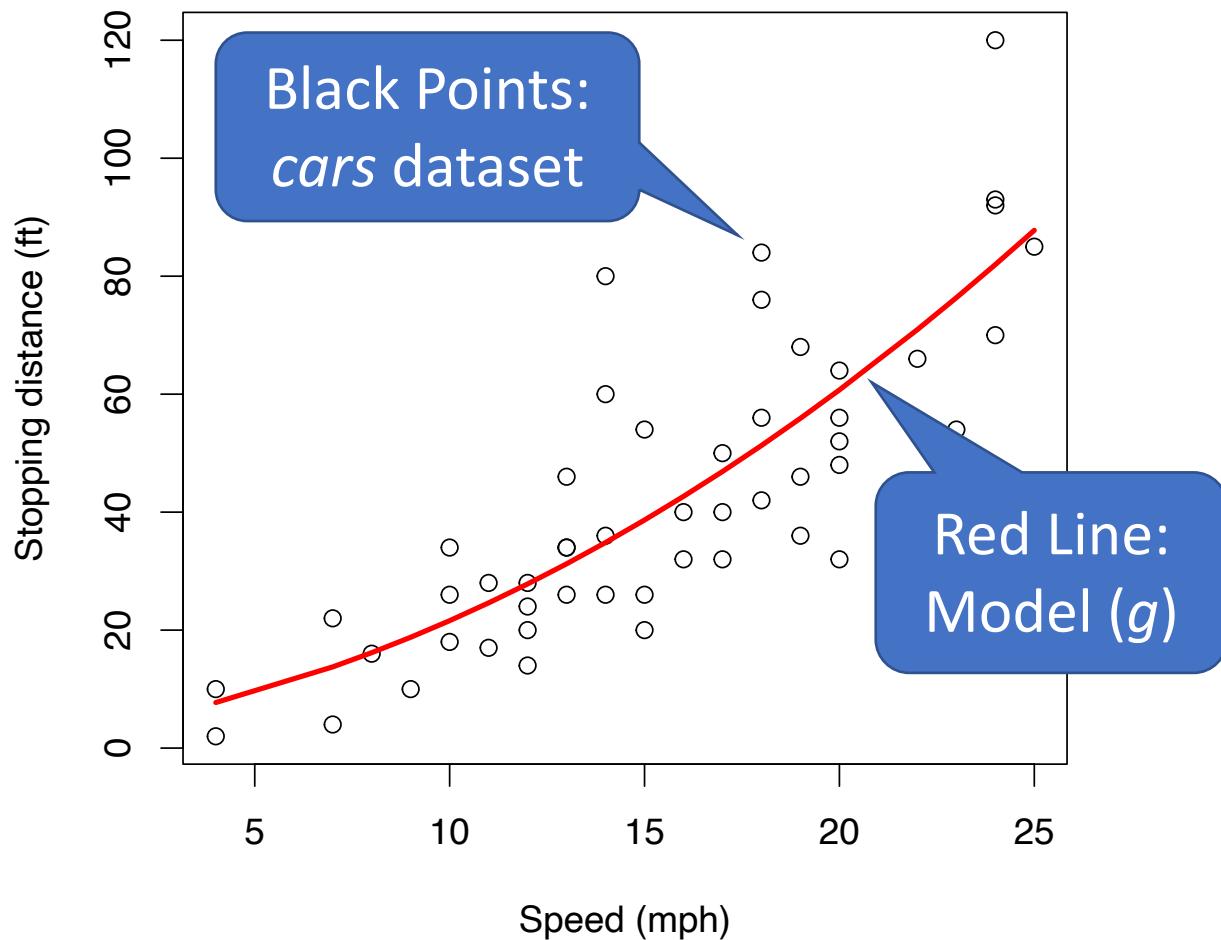
### 3.1 An Example of Fit and Predict in R

```
1 # Summarize the cars dataset
2 summary(cars)
3 # Create the black points
4 plot(cars, xlab="Speed (mph)", ylab="Stopping distance (ft)")
5 # Fit a model g to cars dataset,
6 # assuming dist is a quadratic function of speed
7 g = glm(dist ~ poly(speed, 2), data=cars)
8 # Summarize the model g
9 summary(g)
10 o = order(cars$speed)
11 # Show predictions as a red line
12 lines(cars$speed[o], predict(g,cars)[o], col="red", lwd=3)
```

Listing 1. Example of fit and predict in R

# 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	: 2.00
1st Qu.	:12.0	: 26.00
Median	:15.0	: 36.00
Mean	:15.4	: 42.98
3rd Qu.	:19.0	: 56.00
Max.	:25.0	:120.00

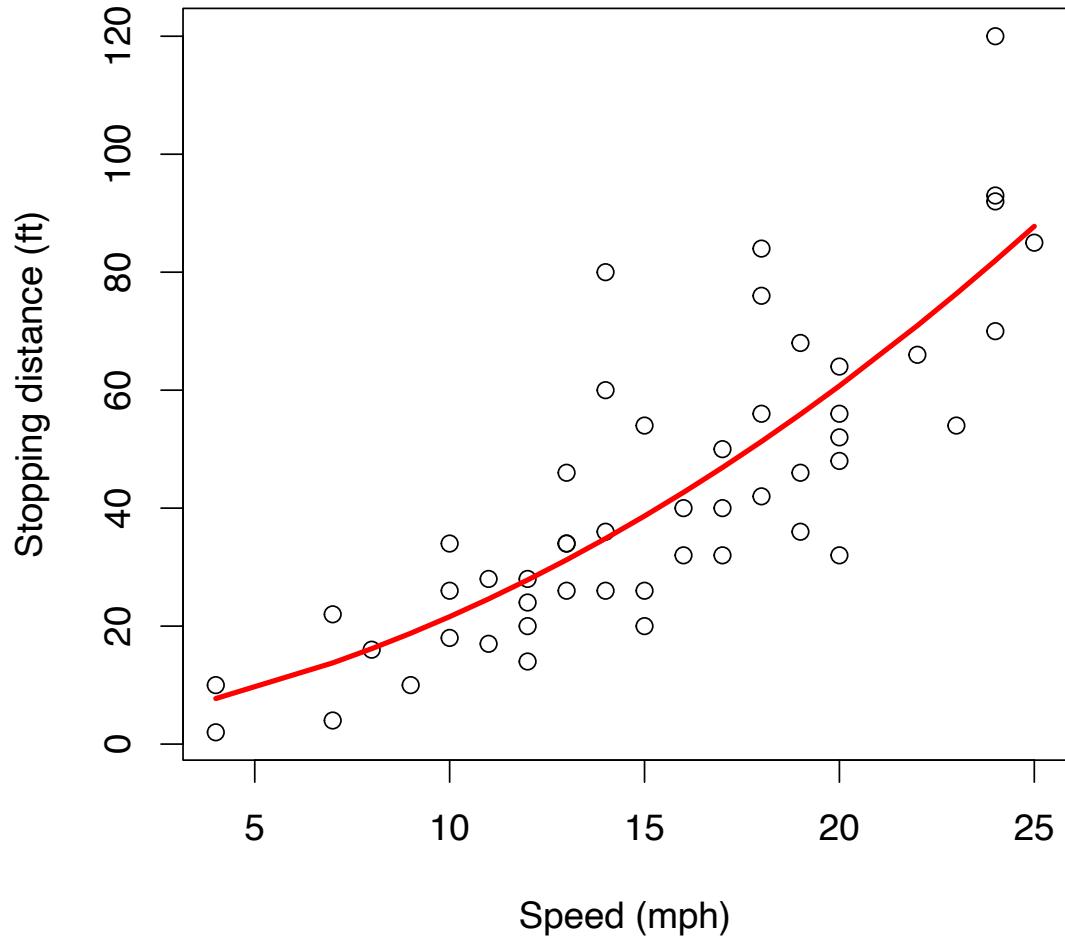
Summarize a  
dataset (*cars*)

Fit a model ( $g$ ) to data ( $\text{cars}$ )

# Regression in R:

Summarize (almost) anything

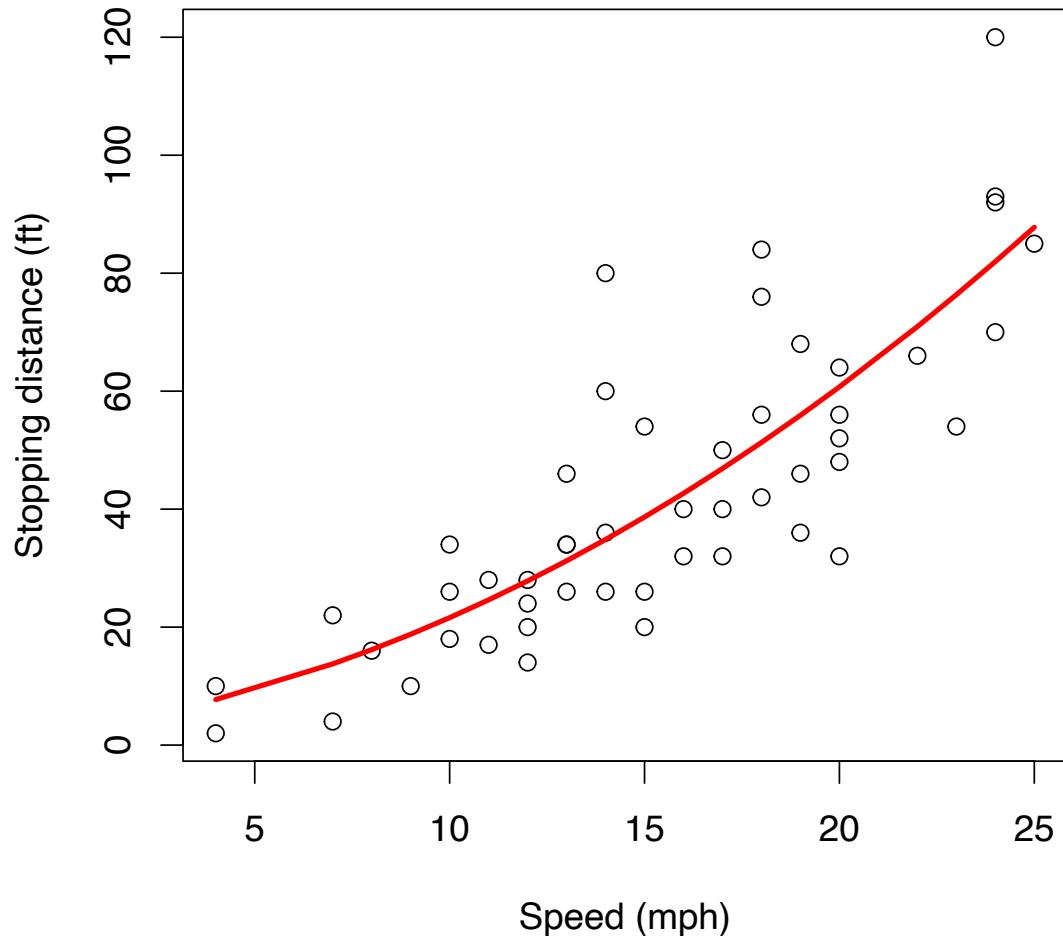
```
> plot(cars, xlab="Speed (mph)", ylab="Stopping distance (ft)")  
> g = glm(dist ~ poly(speed,2), data=cars)  
> o = order(cars$speed)  
> lines(cars$speed[o], predict(g,cars)[o], col="red", lwd=3)
```



Fit a model ( $g$ ) to data ( $\text{cars}$ )

# 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)  
> summary(g)
```

Predict

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  $g$

# What is ~~fine tuning~~ fit?

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

## R Program

```
> plot(cars, xlab="Speed (mph)", ylab="Stopping distance (ft)")  
> g = glm(dist ~ poly(speed,2), data=cars)
```

## gft Program

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

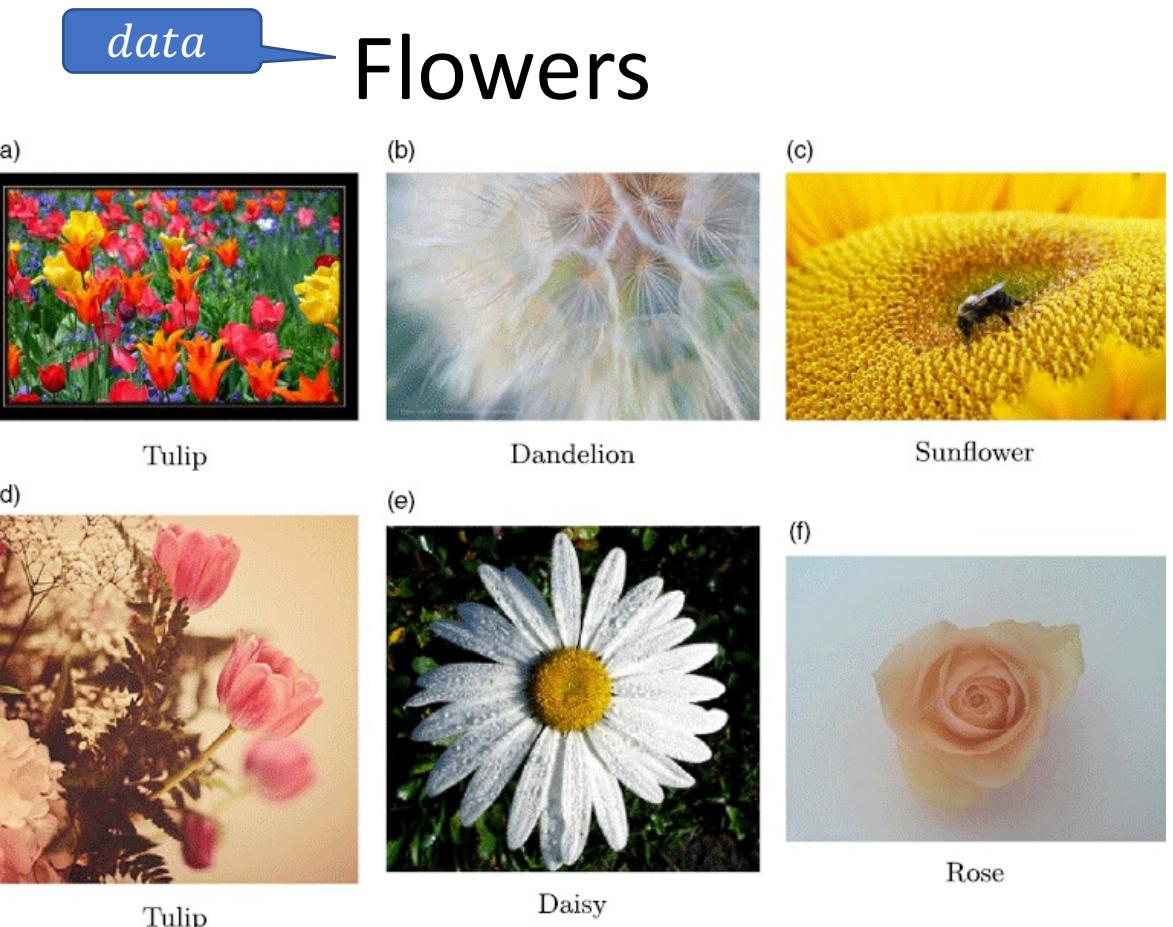
Pre-trained (foundation) model	Parameters	Training data
ResNet-50 (He et al. 2016)	23M	14M images from ImageNet
VT (Wu et al. 2020)	11.7–21.9M	14M images from ImageNet
Wav2vec (Baevski et al. 2020)	95–317M	960 hours from LibriSpeech
BERT (Devlin et al. 2019)	110–340M	3.3B words from Books/Wiki
ERNIE 2.0 (Sun et al. 2020) 22 May 2022	110–340M	7.9B en + 15B zh tokens <a href="https://github.com/kwchurc">https://github.com/kwchurc</a>
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)	<a href="https://github.com/commoncrawl">https://github.com/commoncrawl</a>
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
LiS-Speech	<a href="https://koithito.com/LiS-Speech-Dataset/">https://koithito.com/LiS-Speech-Dataset/</a>

# What is ~~fine tuning~~ fit?

- $f_{\text{fit}}: f_{\text{pre}} + \text{data} \rightarrow f_{\text{post}}$
- $f_{\text{pre}}$ : ResNet-50
  - Trained on ImageNet (1k class labels)
  - Maps images to ImageNet Labels
- $\text{data}$ : Flowers
  - Maps images to Flower Labels (5 class labels)
  - Flower Labels  $\neq$  ImageNet Labels
- $f_{\text{post}}$ :
  - maps images to Flower Labels

$f_{\text{pre}}$ : 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) 22 May 2022	110–340M	7.9B en + 15B zh tokens <a href="https://github.com/kwchurc/">https://github.com/kwchurc/</a>
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)	<a href="https://github.com/commoncrawl">https://github.com/commoncrawl</a>
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
LJSpeech	<a href="https://keithito.com/LJSpeech-Dataset/">https://keithito.com/LJSpeech-Dataset/</a>

# 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

$f_{pre}$ : Pre-trained Model

Pre-trained (foundation) model	Parameters	Training data
ResNet-50 (He et al. 2016)	23M	14M images from ImageNet
ViT (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) 22 May 2022	110–340M	7.9B en + 15B zh tokens <a href="https://github.com/kwchurc">https://github.com/kwchurc</a>
ERNIE 3.0 (Sun et al. 2021)	10B	375B tokens of text, as well as knowledge graph

## SQuAD

- 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*

Text with no markup

Dataset	Description
Billion Word Benchmark (Chelba et al. 2013)	A billion words of English
Common Crawl (Buck, Heafield, and van Ooyen 2014)	<a href="https://github.com/commoncrawl">https://github.com/commoncrawl</a>
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
LJSpeech	<a href="https://keithito.com/LJSpeech-Dataset/">https://keithito.com/LJSpeech-Dataset/</a>

# *gft* Design Goals

- Inclusiveness:
  - Accessible to Masses
    - (including non-programmers)
- **Portability**
- Ease-of-Use/Transparency >> SOTA
- Hide Complexity/Magic (Alchemy)
  - No one would suggest
    - Regression-like methods are ``intelligent''

# Portability: Inclusiveness + Hedge

## *gft* Programs

Hedge  
(Supply Chains)

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

H → HuggingFace

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

P → PaddleNLP

Emphasis  
on Chinese

# Strategy Conclusions

- Inclusiveness
  - Appeal to Power Users as well as Masses
  - Accessible to Masses
  - Portability
- Transparency
  - Avoid Magic / Alchemy
  - Hide Complexity
- Embrace Historical Precedents
  - Little Languages from Unix
  - Fit/Predict/Summary/glm Equations from R

# Agenda: Part A

- ✓ Strategy
- *gft* Cheatsheet
  - 4-5 Functions + 4 Arguments
  - Amazing how much can be done with so little
  - Seven Simple Examples
- More Tasks: MT, ASR, Vision
- Two Questions
  - How do we find the good stuff?
  - How do we use the good stuff?
- Why de-emphasize pre-training?
- Conclusions

# *gft* Cheat Sheet (General Fine-Tuning)

## 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

Next topic



## Datasets: emotion

like 18

Tasks: multi-class-classification text-classification-other-emotion-classification Task Categories: text-classification

Language Creators: machine-generated Annotations Creators: machine-generated Source Datasets: original

Dataset card Files and versions

## Dataset Preview

Subset

default

Split

train

text (string)

i didnt feel humiliated

label (class label)

0 (sadness)

i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake

0 (sadness)

im grabbing a minute to post i feel greedy wrong

3 (anger)

i am ever feeling nostalgic about the fireplace i will know that it is still on the property

2 (love)

i am feeling grouchy

3 (anger)

ive been feeling a little burdened lately wasnt sure why that was

0 (sadness)

ive been taking or milligrams or times recommended amount and ive fallen asleep a lot faster but i also feel like so funny

5 (surprise)

i feel as confused about life as a teenager or as jaded as a year old man

4 (fear)

i have been with petronas for years i feel that petronas has performed well and made a huge profit

1 (joy)

i feel romantic too

2 (love)

i feel like i have to make the suffering i m seeing mean something

0 (sadness)

Emotion Dataset  
*classify: label~text*

# More Terminology

- Dataset (4k)
- Models (40k)
- Tasks
  - classify
  - regress
  - translation
  - ASR
  - ...

The screenshot shows the Hugging Face datasets homepage. At the top, there's a navigation bar with a logo, a search bar, and links for Models, Datasets, Spaces, Docs, Solutions, Pricing, Log In, and Sign Up. The 'Datasets' link is highlighted with a red box.

Below the navigation is a sidebar with 'Task Categories' (text-classification, conditional-text-generation, sequence-modeling, question-answering, structure-prediction, other), 'Tasks' (language-modeling, machine-translation, named-entity-recognition, sentiment-classification, extractive-qa, summarization), 'Languages' (en, es, fr, de, pt, it, +188), 'Multilinguality' (monolingual, multilingual, translation, other-programming-languages, en, bg, +52), 'Sizes' (10K<n<100K, 1K<n<10K, 100K<n<1M, unknown, 1M<n<10M, n<1K, +29), and 'Licenses' (cc-by-4.0, mit, cc-by-4.0, cc-by-sa-4.0, cc-by-sa-4.0, cc-by-sa-3.0, +132).

The main area displays a list of datasets. A red box highlights the 'glue' dataset entry. The list includes:

- glue
- squad
- wikitext
- xtreme
- squad\_es
- common\_voice
- squad\_v2
- wikiann
- anli
- super\_glue
- tweet\_eval
- imdb
- red\_caps
- xnli
- wmt16
- ag\_news
- amazon\_polarity
- xsum

Each dataset entry shows a preview link, update date, size, and download count.

<https://huggingface.co/datasets>

# Two Popular Datasets



<https://gluebenchmark.com/>

22 May 2022



<https://rajpurkar.github.io/SQuAD-explorer/>

[https://github.com/kwchurch/ACL2022\\_deepnets\\_tutorial](https://github.com/kwchurch/ACL2022_deepnets_tutorial)

28

# More Terminology

- Dataset (4k)
  - Splits: train, validation, test
  - Columns:
    - id, title, context, question, answers
  - Rows
- Models (36k)
- Tasks:
  - classification, regression, etc.

The screenshot shows the Hugging Face dataset viewer for the SQuAD 2.0 dataset. The top navigation bar includes links for Models, Datasets, Spaces, Docs, Solutions, Pricing, Log In, and Sign Up. The main page displays dataset metadata: Tasks (extractive-qa), Task Categories (question-answering), Languages (en), Multilinguality (monolingual), Size Categories (10K<n<100K), Licenses (cc-by-4.0), and Language Creators (crowdsourced). Below this, a 'Dataset card' section shows the 'plain\_text' subset. A red box highlights the first two columns of the table: 'id (string)' and 'title (string)'. Another red box highlights the 'train' split selection in the top right. A large blue banner at the bottom reads 'SQuAD 2.0 The Stanford Question Answering Dataset'.

[https://huggingface.co/datasets/squad/viewer/plain\\_text/train](https://huggingface.co/datasets/squad/viewer/plain_text/train)

Subset

plain\_text

Split

train

<b>id (string)</b>	<b>title (string)</b>	<b>context (string)</b>	<b>question (string)</b>	<b>answers (json)</b>
5733a70c4776f41900660f63	University_of_Notre_Dame	All of Notre Dame's undergraduate students are a part of one of the five undergraduate colleges a...	What was created at Notre Dame in 1962 to assist first year students?	{ "text": [ "The First Year of Studies program" ], "answer_start": [ 155 ] }
5733a70c4776f41900660f65	University_of_Notre_Dame	All of Notre Dame's undergraduate students are a part of one of the five undergraduate colleges a...	Which organization declared the First Year of Studies program at Notre Dame "outstanding?"	{ "text": [ "U.S. News & World Report" ], "answer_start": [ 647 ] }
5733a7bd4776f41900660f6b	University_of_Notre_Dame	The university first offered graduate degrees, in the form of a Master of Arts (MA), in the 1854-1855 academic year. The program expanded to include Master of Laws (LL.M.) and Master of Civil Engineering in its early stages of growth, before a formal graduate school education was developed with a thesis not required to receive the degrees. This changed in <b>1924</b> with formal requirements developed for graduate degrees, including offering Doctorate (PhD) degrees. Today each of the five colleges offer graduate education. Most of the departments from the College of Arts and Letters offer PhD programs, while a professional Master of Divinity (M.Div.) program also exists. All of the departments in	The granting of Doctorate degrees first occurred in what year at Notre Dame?	{ "text": [ "1924" ], "answer_start": [ 358 ] }



- **Dataset (4k)**

- Splits: train, validation, test
- Columns:
  - title, context, question, answers
- Rows

Answer starts at  
context[358:]

Answer is a span  
(substring of context)

Task: Question Answering (QA)

Classify Spans:  
Answer is a substring of context

# GLUE

<https://huggingface.co/datasets/glue/viewer/qqp/train>

Hugging Face  Models Datasets Spaces Docs Solutions Pricing Log In Sign Up

Datasets: **glue** like 29

Tasks: natural-language-inference acceptability-classification text-classification-other-paraphrase-identification + 4 Task Categories: text-classification text-scoring Languages: en Multilinguality: monolingual

Size Categories: 10K<n<100K Licenses: cc-by-4.0 Language Creators: unknown Annotations Creators: unknown Source Datasets: unknown

Dataset card Files and versions

Dataset Preview

Subset	Split		
qqp	train		
question1 (string)	question2 (string)	label (class label)	idx (int)
How is the life of a math student? Could you describe your own experiences?	Which level of preparation is enough for the exam jlpt5?	0 (not_duplicate)	0
How do I control my horny emotions?	How do you control your horniness?	1 (duplicate)	1
What causes stool color to change to yellow?	What can cause stool to come out as little balls?	0 (not_duplicate)	2
What can one do after MBBS?	What do i do after my MBBS ?	1 (duplicate)	3
Where can I find a power outlet for my laptop at Melbourne Airport?	Would a second airport in Sydney, Australia be needed if a high-speed rail link was created between Melbourne and Sydney?	0 (not_duplicate)	4
How not to feel guilty since I am Muslim and I'm conscious we won't have sex together?	I don't beleive I am bulimic, but I force throw up atleast once a day after I eat something and feel guilty. Should I tell somebody, and if so who?	0 (not_duplicate)	5
How is air traffic controlled?	How do you become an air traffic controller?	0 (not_duplicate)	6
What is the best self help book you have read? Why? How did it change your life?	What are the top self help books I should read?	1 (duplicate)	7
Can I enter University of Melbourne if I couldn't achieve the guaranteed marks in Trinity College Foundation?	University of the Philippines: If I take a second BFA in the UP College of Fine Arts, can I be exempted from gen. ed. or core subjects?	0 (not_duplicate)	8

# GLUE

<https://huggingface.co/datasets/glue/viewer/qqp/train>

Hugging Face  Models Datasets Spaces Docs Solutions Pricing Log In Sign Up

Datasets: **glue** like 29

Tasks: natural-language-inference acceptability-classification text-classification-other-paraphrase-identification + 4 Task Categories: text-classification text-scoring Languages: en Multilinguality: monolingual

Size Categories: 10K<n<100K Licenses: cc-by-4.0 Language Creators: unknown Annotations Creators: unknown Source Datasets: unknown

Dataset card Files and versions

cola  
sst2  
mrpc  
qqp  
sts  
mnli  
mnli\_mismatched  
mnli\_matched  
qnli  
rte  
wnli  
ax

Terminology: Subset

Split	label (class label)	idx (int)
train	0 (not_duplicate)	0
ion is enough for the exam jlpt5?	1 (duplicate)	1
r horniness?	0 (not_duplicate)	2
o come out as little balls?	1 (duplicate)	3
MBBS ?	0 (not_duplicate)	4
Would a second airport in Sydney, Australia be needed if a high-speed rail link was created between Melbourne and Sydney?	0 (not_duplicate)	5
I don't beleive I am bulimic, but I force throw up atleast once a day after I eat something and feel guilty. Should I tell somebody, and if so who?	0 (not_duplicate)	6
How do you become an air traffic controller?	1 (duplicate)	7
What are the top self help books I should read?	0 (not_duplicate)	8

Where can I find a power outlet for my laptop at Melbourne Airport?  
Would a second airport in Sydney, Australia be needed if a high-speed rail link was created between Melbourne and Sydney?

How not to feel guilty since I am Muslim and I'm conscious we won't have sex together?  
I don't beleive I am bulimic, but I force throw up atleast once a day after I eat something and feel guilty. Should I tell somebody, and if so who?

How is air traffic controlled?  
How do you become an air traffic controller?

What is the best self help book you have read? Why? How did it change your life?  
What are the top self help books I should read?

Can I enter University of Melbourne if I couldn't achieve the guaranteed marks in Trinity College Foundation?  
University of the Philippines: If I take a second BFA in the UP College of Fine Arts, can I be exempted from gen. ed. or core subjects?

# GLUE

<https://huggingface.co/datasets/glue/viewer/qqp/train>

The screenshot shows the Hugging Face dataset viewer interface for the GLUE dataset. At the top, there's a navigation bar with links for Models, Datasets, Spaces, Docs, Solutions, Pricing, Log In, and Sign Up. Below the navigation bar, the dataset details are shown: Tasks (natural-language-inference, acceptability-classification, text-classification-other-paraphrase-identification), Task Categories (text-classification, text-scoring), Languages (en), Multilinguality (monolingual), Size Categories (10K<n<100K), Licenses (cc-by-4.0), Language Creators (unknown), Annotations Creators (unknown), and Source Datasets (unknown). The main area displays a 'Dataset card' with tabs for 'Dataset card' and 'Files and versions'. On the left, there's a 'Dataset Preview' section with a subset dropdown set to 'cola'. The preview table has columns for 'sentence (string)', 'label (class label)', and 'idx (int)'. The 'label' column contains values like '1 (acceptable)' repeated 11 times from index 0 to 10. A blue speech bubble points to the 'train' option in a dropdown menu for the 'subset' field. Another blue speech bubble points to the 'label (class label)' column header in the preview table.

sentence (string)	label (class label)	idx (int)
Our friends won't buy this analysis, let alone the next one we propose.	1 (acceptable)	0
One more pseudo generalization and I'm giving up.	1 (acceptable)	1
One more pseudo generalization or I'm giving up.	1 (acceptable)	2
The more we study verbs, the crazier they get.	1 (acceptable)	3
Day by day the facts are getting murkier.	1 (acceptable)	4
I'll fix you a drink.	1 (acceptable)	5
Fred watered the plants flat.	1 (acceptable)	6
Bill coughed his way out of the restaurant.	1 (acceptable)	7
We're dancing the night away.	1 (acceptable)	8
Herman hammered the metal flat.	1 (acceptable)	9
The critics laughed the play off the stage.	1 (acceptable)	10

# Two Perspectives on Datasets

## Machine Learning

- Splits: train/val/test
- Treat rows as iid
  - Identical and independently distributed
- Idealistic, but maybe unrealistic
  - Changes over time, Context, etc.
- More papers on models
  - than Datasets
- Relatively little interest in
  - Is the Dataset Realistic?
  - Motivation? (Does anyone care? If so, who?)
  - What is the dataset testing?
  - Sampling/balance? From which population?
  - Can it be gamed? Leakage?

## Psychology (Presupposes $\exists$ Hypothesis)

- Validity
  - Content validity
  - Construct validity
  - Criterion validity
  - Face validity
  - Discriminant validity
- Reliability
  - Inter-rater reliability
  - Test-retest reliability
  - Internal reliability/Inter-item consistency
  - Split half reliability



Lexicography

# Discussion of Validity

## Assumes: Intention

### TYPES OF VALIDITY

**Experimental Validity:** is the study really measuring what it intends?

**INTERNAL VALIDITY** refers to things that happen “inside” the study. Internal validity is concerned with whether we can be certain that it was the IV which caused the change in the DV. If aspects of the experimental situation lack validity, the results of the study are meaningless and we can make no meaningful conclusions from them.

- Internal validity can be affected by a lack of **mundane realism**. This could lead the participants to act in a way which is unnatural, thus making the results less valid.
- Internal validity can also be affected by **extraneous variables** (see below).

EXTRANEous VARIABLE	HOW DOES IT AFFECT VALIDITY?	HOW CAN IT BE OVERCOME?
<b>Situational variables</b> (anything to do with the environment of the experiment): time of day, temperature, noise levels etc	Something about the situation of the experiment could act as an EV if it has an effect on the DV. For example, poor lighting could affect participants performance on a memory test	Situational variables can be overcome by the use of <b>standardised procedures</b> which ensure that all participants are tested under the same conditions.
<b>Participants variables</b> (anything to do with differences in the participants): age, gender, intelligence, skill, past experience, motivation, education etc.	It may be that the differences between the participants cause the change in the DV. For example, one group may perform better on a memory test than another because they are younger, or more motivated.	Participant variables can be completely removed by using a <b>repeated measures design</b> (the same participants are used in each condition). <b>Matched pairs</b> (participants in each group are matched) could also be used.
<b>Investigator effects:</b> this refers to how the behaviour and language of the experimenter may influence the behaviour of the participants. The way in which an experimenter asks a question might act as a cue for the participant. Also known as <b>experimenter bias</b>	<b>Leading questions</b> from the experimenter may consciously or unconsciously alter how the participant responds. For example, the experimenter may provide verbal or <u>non-verbal</u> encouragement when the participant behaves in a way which supports the hypothesis.	Investigator effects can be overcome by using a <b>double blind</b> technique. This is when the person who carries out the research is not the person who designed it.
<b>Demand characteristics:</b> participants are often searching for cues as to how to behave in an experiment. There could be something about the experimental situation or the behaviour of the experimenter (see <b>investigator effects</b> ) which communicates to the participant what is “demanded” of them.	The structure of the experiment could lead the participant to guess the aim of the study. For example, participants may perform a memory test, be made to exercise, and then given another memory test. This may lead the participants to guess that the study is about the effect of exercise on memory, which may cause them to change their behaviour	When designing a study, it is important to try and create a situation where the participants will not be able to guess what the aim of the study is.
<b>Participant effects:</b> participants are aware that they are in an experiment, and so may behave unnaturally.	They may be overly helpful and want to please the experimenter. This leads to artificial behaviour. Alternatively, they may decide to go against the experimenter's aims and deliberately act in a way which spoils the experiment. This is the “ <b>screw you</b> ” effect.	Again, by designing a study so that the participants cannot guess the aims, participant effects can be reduced.

# *gft* Cheat Sheet (General Fine-Tuning)

## 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

Next topic

# Models

Terminology:  
 $f, f_{pre}, f_{post}$

<https://huggingface.co/models>



Tasks

- Fill-Mask
- Question Answering
- Summarization
- Table Question Answering
- Text Classification
- Text Generation
- Text2Text Generation
- Token Classification
- Translation
- Zero-Shot Classification
- Sentence Similarity

Libraries

- PyTorch
- TensorFlow
- JAX

Datasets

- common\_voice
- wikipedia
- squad
- bookcorpus
- glue
- c4
- conll2003
- dcp europarl jrc-acquis

+ 884

Languages

- en
- es
- fr
- de
- zh
- sv
- fi
- ru

+ 172

Models

Datasets

Spaces

Docs

Solutions

Sign Up

Post Downloads

Terminology:

$f_{pre}$  (base)

Task:

Fill Mask

Terminology:

$f_{post}$  (finetuned)

Task:

Translation

Task:

Token classification

Models 35,684

Search Model

distilgpt2

Text Generation • Updated May 21, 2021 • ↓ 33.4M • 39

cross-encoder/ms-marco-MiniLM-L-12-v2

Text Classification • Updated Aug 5, 2021 • ↓ 10.7M • 5

gpt2

Text Generation • Updated May 19,

distilbert-base-uncased

Fill-Mask • Updated Aug 29, 2021 • ↓ 4.88M • 47

distilbert-

Text Classification

distilroberta-base

Fill-Mask • Updated May 20, 2021 • ↓ 2.98M • 13

bert-base-chinese

Fill-Mask • Updated Sep 6, 2021 • ↓ 2.77M • 16

bert-base-uncased

Fill-Mask • Updated May 18, 2021 • ↓ 16.8M • 125

Helsinki-NLP/opus-mt-zh-en

Translation • Updated Feb 26, 2021 • ↓ 8.27M • 22

roberta-base

Fill-Mask • Updated Jul 6, 2021 • ↓ 5.07M

xlm-roberta-large-finetuned-conll03-english

Token Classification • Updated Oct 12, 2020 • ↓ 4.54M • 13

roberta-large

Fill-Mask • Updated May 21, 2021 • ↓ 3.76M • 28

bert-base-cased

Fill-Mask • Updated Sep 6, 2021 • ↓ 2.77M • 16

xlm-roberta-base

Fill-Mask • Updated Sep 6, 2021 • ↓ 2.77M • 16

# *gft* Cheat Sheet (General Fine-Tuning)

## 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~x<sub>1</sub> + x<sub>2</sub>***
- --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

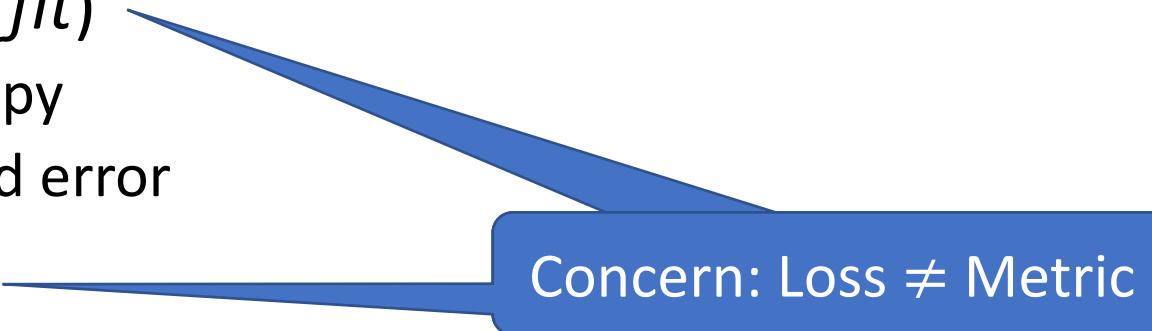
Next topic

# Simple Equations Cover Many Cases of Interest

## GLUE: A Popular Benchmark

Subset	Dataset	Equation
COLA	H:glue,cola	<i>classify</i> : $label \sim sentence$
SST2	H:glue,sst2	<i>classify</i> : $label \sim sentence$
WNLI	H:glue,wnli	<i>classify</i> : $label \sim sentence$
MRPC	H:glue,mrpc	<i>classify</i> : $label \sim sentence_1 + sentence_2$
QNLI	H:glue,qnli	<i>classify</i> : $label \sim sentence_1 + sentence_2$
QQP	H:glue,qqp	<i>classify</i> : $label \sim question + sentence$
SSTB	H:glue,sstb	<i>regress</i> : $label \sim question_1 + question_2$
MNLI	H:glue,mnli	<i>classify</i> : $label \sim premise + hypothesis$

# Loss Functions and Metrics (Defaults depend on task)

- --eqn
    - *classify* :  $label \sim question + sentence$
    - *regress* :  $label \sim question1 + question2$
  - Loss functions (used in *gft\_fit*)
    - Classification → Cross entropy
    - Regression → Mean squared error
  - Metrics (used in *gft\_eval*)
    - Classification → Accuracy
    - Regression → Mean squared error
    - --metric H:glue,cola → Matthews Correlation
- 
- Concern: Loss ≠ Metric

# Equation Keywords $\approx$ Pipeline Tasks

Benchmark	Subset	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	$classify : label \sim question + sentence$
VAD		C:\$gft/datasets/VAD/VAD <a href="https://github.com/kwchurch/ACL2022_deepnets_tutorial">https://github.com/kwchurch/ACL2022_deepnets_tutorial</a>	$regress: Valence + Arousal + Dominance \sim Word$

*regress: Valence + Arousal + Dominance ~ Word*

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

<b>word</b>	<b>Val</b>	<b>Arousal</b>	<b>Dom</b>	<b>Dist</b>
<i>open</i>	0.620	0.480	0.569	0.00
<i>unfold</i>	0.612	0.510	0.520	0.06
<i>reopen</i>	0.656	0.528	0.568	0.06
<i>close</i>	0.292	0.260	0.263	0.50
<i>closed</i>	0.240	0.164	0.318	0.55
<i>undecided</i>	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*.

	<b>Antonyms</b>				<b>Sim</b> <b>SimLex</b>
	<b>adj</b>	<b>noun</b>	<b>verb</b>	<b>fallows</b>	
<b>cor</b>	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.

# regress: Valence + Arousal + Dominance ~ Word

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

word	Val	Arousal	Dom	Dist
<i>open</i>	0.620	0.480	0.569	0.00
<i>unfold</i>	0.612	0.510	0.520	0.06
<i>reopen</i>	0.656	0.528	0.568	0.06
<i>close</i>	0.292	0.260	0.263	0.50
<i>closed</i>	0.240	0.164	0.318	0.55
<i>undecided</i>	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*.

Train  $f$  on words, but apply  $f$  to texts  
(Generalizations: OOVs, MWEs, negation)

```

1 gft_fit --model H:bert-base-cased \
2   --data C:$gft/datasets/VAD/VAD \
3   --eqn 'regress: Valence + Arousal + Dominance ~ Word' \
4   --output_dir $outdir

```

Listing 20. An equation with a vector on the left hand side (lhs)

Input, $x$	Predictions, $\hat{y}$			Gold, $y$		
	$\hat{V}$	$\hat{A}$	$\hat{D}$	$V$	$A$	$D$
love	0.976	0.530	0.675	1.000	0.519	0.673
she loves me	0.931	0.452	0.648	NA	NA	NA
lovable	0.920	0.318	0.608	0.948	0.335	0.565
she loves me not	0.382	0.375	0.349	NA	NA	NA
ugly duckling	0.300	0.517	0.242	NA	NA	NA
unlovable	0.203	0.502	0.399	NA	NA	NA
she does not love me	0.189	0.433	0.262	NA	NA	NA
she hates me	0.135	0.685	0.363	NA	NA	NA
loathe	0.094	0.790	0.413	0.135	0.714	0.445
hate	0.017	0.780	0.451	0.031	0.802	0.430

Table 7. Some gold labels and predictions from model,  $f_{post}$ , from Listing 20

# *gft* Cheat Sheet (General Fine-Tuning)

## 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

# Tasks

- classify, text-classification  $y \in \{0,1,2, \dots\}$
- regress  $y \in \mathbb{R}$  or  $y \in \mathbb{R}^N$
- QA, Question Answering, classify spans  $y$  for each start/end of span
- token classification
  - NER (Named Entity Recognition)
  - POS (Part of Speech Tagging) $y$  for each token
- translation, MT
- ASR, Automatic Speech Recognition, ctc  $y$  for each phoneme

# *gft* Cheat Sheet (General Fine-Tuning)

## 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

# Agenda: Part A

✓ Strategy

✓ *gft Cheatsheet*

- 4-5 Functions + 4 Arguments
- Amazing how much can be done with so little

## ➤ Seven Simple Examples

- More Tasks: MT, ASR, Vision
- Two Questions
  - How do we find the good stuff?
  - How do we use the good stuff?
- Why de-emphasize pre-training?
- Conclusions

# 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
```

Summarize both  
data and model

Emotion Classes

# 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
```

Search Engine  
(Embarrassment of Riches)

```
# 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
```

Default model → Sentiment Analysis

```
# 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
```

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)

# 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
```

Specify split

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

Equation specifies  
task & columns

Read Input from dataset  
(as opposed to stdin)

# 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
```

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
```

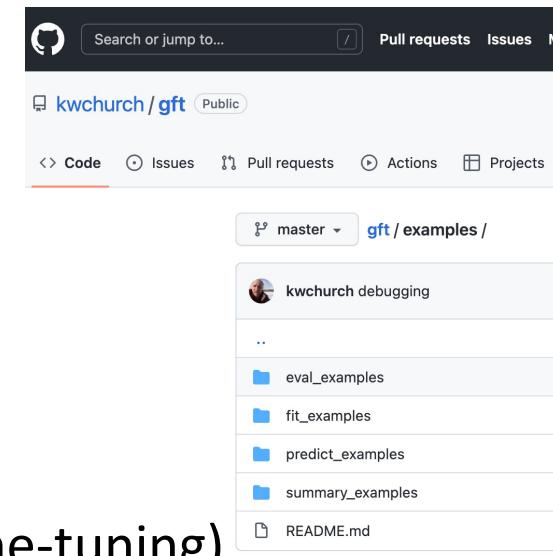
6

```
# 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:

<https://github.com/kwchurch/gft>

## 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\_\_)

# Hundreds of Examples: Regression Testing

Regression testing looks  
for errors in output

## Input

gft / examples / summary\_examples / model / model.HuggingFace / roberta-base.sh

 kwchurch first commit

1 contributor

Executable File | 3 lines (2 sloc) | 46 Bytes

```
1 #!/bin/sh
2
3 gft_summary --model H:roberta-base
```

22 May 2022

<https://github.com/kwchurch>

## Output

gft / examples / summary\_examples / model / model.HuggingFace / roberta-base.out

 kwchurch debugging

1 contributor

298 lines (298 sloc) | 13.5 KB

```
1  model: roberta-base    model: roberta-base    downloads: 9098576    likes: 23    task: fill-mask
2  model: roberta-base    labels: LABEL_0, LABEL_1
3  RobertaForMaskedLM(
4      (roberta): RobertaModel(
5          (embeddings): RobertaEmbeddings(
6              (word_embeddings): Embedding(50265, 768, padding_idx=1)
7              (position_embeddings): Embedding(514, 768, padding_idx=1)
8              (token_type_embeddings): Embedding(1, 768)
9              (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
10             (dropout): Dropout(p=0.1, inplace=False)
11         )
12         (encoder): RobertaEncoder(
13             (layer): ModuleList(
14                 (0): RobertaLayer(
15                     (attention): RobertaAttention(
16                         (self): RobertaSelfAttention(
17                             (query): Linear(in_features=768, out_features=768, bias=True)
18                             (key): Linear(in_features=768, out_features=768, bias=True)
19                             (value): Linear(in_features=768, out_features=768, bias=True)
20                             (dropout): Dropout(p=0.1, inplace=False)
```

Pass: No Errors

# 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/mlmi_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/mlmi_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

- Function
  - fit, predict, eval, summary
- Models
  - model.HuggingFace/model.Paddle
  - our\_models/their\_models
- Data
  - data.HuggingFace/data.Paddle
- Benchmarks/Tasks
  - GLUE, SQuAD, sentiment,...
- Hyper-parameter Tuning

# 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

# What is ``Good'' Stuff? Accuracy? Downloads?

## Best of Hubs

Model	VAcc	D
C:RoBERTa large, tuned by Yuchen Bian	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 (BERT with no tuning)	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 (D).

Hyper-Parameter Tuning → Small Gains

## SOTA

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).

Should not compare accuracy across splits

Best of Hubs are good  
(but probably not SOTA)

# Agenda: Part A

✓ Strategy

✓ *gft* Cheatsheet

✓ 4-5 Functions + 4 Arguments

✓ Amazing how much can be done  
with so little

✓ Seven Simple Examples

## ➤ More Tasks: MT, ASR, Vision

- Two Questions

- How do we find the good stuff?

- How do we use the good stuff?

- Why de-emphasize pre-training?

- Conclusions

```
# 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

```
# 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
```

# More tasks (3 of 4)

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

Output is non-deterministic

```

# 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

```

# More tasks (4 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

# 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

```

# More tasks (4 of 4)

- ✓ 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
	

```
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,
```

Default Model → 1000 Labels

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

```
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
```

A Model with Cat/Dog Labels

# 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

# --task argument → HuggingFace Pipelines

[https://huggingface.co/docs/transformers/v4.16.2/en/main\\_classes/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-of-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  applause|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](#).

# Agenda: Part A

- ✓ Strategy
  - ✓ *gft* Cheatsheet
    - ✓ 4-5 Functions + 4 Arguments
    - ✓ Amazing how much can be done with so little
  - ✓ Seven Simple Examples
  - ✓ More Tasks: MT, ASR, Vision
- **Two Questions**
- How do we find the good stuff?
  - How do we use the good stuff?
  - Why de-emphasize pre-training?
  - Conclusions

# Two Questions

- **How do I find the *good* stuff?**
  - And how do I use the *good* stuff?
  - Presupposition:
    - Hubs encourage sharing of
      - Datasets
      - Models
      - Tasks

# Embarrassment of Riches

- Hubs have 40k models & 4k datasets (increasing 3x per year)
  - Q: How can I find the good stuff?  
➤ A: *gft\_summary*

# Metrics of Success: How do we find good stuff?

The screenshot shows the Hugging Face dataset page for 'common\_voice'. At the top, there's a search bar and a navigation bar with links for Models, Datasets, Spaces, Docs, Solutions, Pricing, and a user profile. Below the header, the dataset title 'common\_voice' is displayed with a 'like' count of 25. The page includes filters for Tasks (automatic-speech-recognition), Task Categories (speech-processing), Languages (ab, ar, as, + 57), Multilinguality (multilingual), Size Categories (n<1K, 10K<n<100K, 100K<n<1M, + 1), Licenses (cc0-1-0), Language Creators (crowdsourced), and Annotations Creators (crowdsourced). A source dataset link 'extended|common\_voice' is also present.

**Dataset Structure**

- Data Instances
- Data Fields
- Data Splits

**Dataset Creation**

- Curation Rationale
- Source Data
- Annotations
- Personal and Sensitive Information

**Considerations for Using the Data**

- Social Impact of Dataset
- Discussion of Biases
- Other Known Limitations

**Additional Information**

Dataset Curators

**Dataset Preview**

Subset: en, Split: train. Note: The dataset preview is not available for this split.

**Dataset Card for common\_voice**

**Dataset Summary**

The Common Voice dataset consists of a unique MP3 and corresponding text file. Many of the 9,283 recorded hours in the dataset also include demographic metadata like age, sex, and accent that can help train the accuracy of speech recognition engines.

[https://github.com/kwchurch/ACL2022\\_deeppnets\\_tutorial](https://github.com/kwchurch/ACL2022_deeppnets_tutorial)

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

# How do I find the good stuff? Popular Datasets?

gft\_summary --data H:\_\_contains\_\_wiki --topn 20

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

# How do I find the good stuff? Popular Datasets? gft\_summary –data H: contains  --topn 20

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

# *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    likes: 107
# task: AutomaticSpeechRecognition    model: facebook/hubert-large-ls960-ft    downloads: 65234    likes: 10
# task: AutomaticSpeechRecognition    model: facebook/wav2vec2-large-960h-lv60-self    downloads: 64927    likes: 10
# task: AutomaticSpeechRecognition    model: facebook/wav2vec2-large-xlsr-53    downloads: 55869    likes: 10
# task: AutomaticSpeechRecognition    model: hf-internal-testing/processor_with_lm    downloads: 25000    likes: 10
# task: AutomaticSpeechRecognition    model: pyannote/voice-activity-detection    downloads: 18959    likes: 10
# task: AutomaticSpeechRecognition    model: patrickvonplaten/wavlm-libri-clean-100h-base-plus    downloads: 18622    likes: 10
# task: AutomaticSpeechRecognition    model: facebook/wav2vec2-large-960h    downloads: 13622    likes: 10
# task: AutomaticSpeechRecognition    model: facebook/s2t-small-librispeech-asr    downloads: 11808    likes: 10
```

*gft\_summary*: How do I find the good stuff?  
Find models associated with a task (ASR)

# **task → list of models (by downloads)**

## **dataset → 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: 1  
# task: AutomaticSpeechRecognition model: facebook/wav2vec2-large-  
# task: AutomaticSpeechRecognition model: hf-internal-testing/prod  
# 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
```

```
# 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, in  
# 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 likes:  
# 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: 0
```

# *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

Many Downloads, but not SOTA

Compressed (distilbert)

Ease-of-Use

Large Model  
(RoBERTa Large)

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.

# 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.

The screenshot shows a Cambridge University Press page for an article. On the left, there's a thumbnail of the journal cover for 'Natural Language Engineering', Volume 28, Part 2, March 2022, edited by Kenneth Ward Church and Valia Kordoni. The main title of the article is 'Emerging Trends: SOTA-Chasing'. It was published online by Cambridge University Press on 08 February 2022. The authors are Kenneth Ward Church and Valia Kordoni. Below the title, there are tabs for 'Article', 'Figures', and 'Metrics', with 'Article' being the active tab. There are also buttons for 'Save PDF', 'Share', 'Cite', and 'Rights & Permissions'. At the bottom, there's a section for the 'Abstract'.

NATURAL  
LANGUAGE  
ENGINEERING

Edited by  
Kenneth Ward Church  
Valia Kordoni

VOLUME 28 PART 2 MARCH 2022  
CAMBRIDGE UNIVERSITY PRESS

**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 Rights & Permissions

**Natural Language**  
**Engineering**

**Abstract**

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

Search 

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

 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/>.

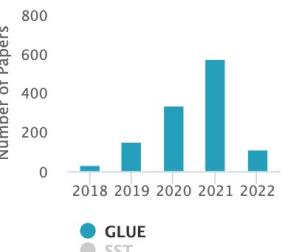
## Benchmarks

 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

Number of Papers



Year	GLUE	SST	SuperGLUE	CoLA
2018	50	0	0	0
2019	150	0	0	0
2020	300	0	0	0
2021	550	0	0	0
2022	100	0	0	0

Legend: GLUE (Blue), SST (Grey), SuperGLUE (Light Grey), CoLA (Dark Grey)

## Licenses

17.0%

# 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
+	PING-AN Omni-Sinicic	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
+	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
+	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
+	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
+	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
+	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	22 May 2022 Facebook AI	RoBERTa	<a href="https://github.com/kwchurch/ACL2022_deepnets_tutorial">https://github.com/kwchurch/ACL2022_deepnets_tutorial</a>	88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8
+	Microsoft D365 AI & MSR AI	MT-DNN ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/90.9	91.0

# 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

# 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	$classify : label \sim question + sentence$
VAD		C:\$gft/datasets/VAD/VAD <a href="https://github.com/kwchurch/ACL2022_deepnets_tutorial">https://github.com/kwchurch/ACL2022_deepnets_tutorial</a>	$regress: Valence + Arousal + Dominance \sim Word$

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

(computed from \$gft/gft\_internals/huggingface\_hub/huggingface\_datasets.txt)

<u>Models</u>	<u>Pipeline Task</u>	$f_{pre}$ : Pre-trained Model	<u>Models</u>	<u>Pipeline Task</u>
9537	None	$f_{pre}$ : Pre-trained Model	160	text-to-speech
3949	text-generation		68	zero-shot-classification
3143	text-classification ( <i>classify</i> )		55	audio-to-audio
2390	fill-mask		54	audio-classification
2253	text2text-generation		26	table-question-answering
1508	automatic-speech-recognition (ctc)		12	object-detection
1460	translation		11	image-segmentation
1340	token-classification ( <i>classify_tokens</i> )		9	text-to-image
1042	conversational		4	structured-data-classification
983	question-answering ( <i>classify_spans</i> )		4	image-to-text
982	feature-extraction		2	voice-activity-detection
338	sentence-similarity		1	zero-shot-image-classification
320	summarization		1	speech-segmentation
261	image-classification		1	protein-folding
				<u>pipeline_tag</u>

# 30k Models → Pipeline Tasks

(computed from \$gft/gft\_internals/huggingface\_hub/huggingface\_datasets.txt)

<u>Models</u>	<u>Pipeline Task</u>	# of Models	Task	Models
9537	None			
3949	text-generation	3949	text-generation	distilgpt2, gpt2, EleutherAI/gpt-neo-1.3B
3143	text-classification ( <i>classify</i> )	3143	text-classification	cross-encoder/ms-marco-MiniLM-L-12-v2, distilbert-base-uncased-finetuned-sst-2-eng
2390	fill-mask	2390	fill-mask bert-base-uncased	distilbert-base-uncased, roberta-base
2253	text2text-generation	2253	text2text-generation	facebook/m2m100_418M, facebook/mbart-large-50-one-to-many-mmt, google/mt5-base
1508	automatic-speech-recognition	1508	ctc	facebook/wav2vec2-base-960h, facebook/hubert-large-ls960-ft, facebook/wav2vec2-la
1460	translation	1460	translation	Helsinki-NLP/opus-mt-zh-en, t5-small, t5-base
1340	token-classification ( <i>classify</i> )	1340	classify	xlm-roberta-large-finetuned-conll03-english, classla/bcms-bertic-ner, dslim/bert-base-N
1042	conversational	1042	conversational	microsoft/DialoGPT-small, microsoft/DialoGPT-medium, facebook/blenderbot_small-90M
983	question-answering ( <i>classify</i> )	983	classify_spans	deepset/roberta-base-squad2, distilbert-base-cased-distilled-squad, bert-large-uncased
982	feature-extraction	982	feature-extraction	feature-extraction, openai/clip-vit-base-patch32, facebook/bart-base, monsoon-nlp/hind
338	sentence-similarity	338	sentence-similarity	sentence-transformers/multi-qa-MiniLM-L6-cos-v1, sentence-transformers/paraphrase-M
320	summarization			L6-v2
261	image-classification			

# 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* 

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

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>I love you</i> 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

# Debugging, Confusion Matrices & Error Analysis

In addition to producing a score with *gft\_eval*, suppose we want to do some deep dives to look at particular errors. The code in Listing 19 will create a confusion matrix based on the validation split.

```
1 f=H:bhadresh-savani/distilbert-base-uncased-emotion
2 gft_predict --eqn 'classify:label~text' --model $f \
3   --data H:emotion --split val > /tmp/pred
4 cut -f2,3 < /tmp/pred | sort | uniq -c | sort -nr > /tmp/conf
```

Listing 19. Code to create confusion matrix

*gft\_predict* outputs TSV (tab separated values) with 4 columns:

1. Input,  $x$
2. Gold label,  $y$
3. Predicted label,  $\hat{y}$
4. Score

The cut statement on line 4 in Listing 19 selects  $y$  and  $\hat{y}$ . The sort and uniq statements count the number of confusions, producing the confusion matrix shown in Table 6. Standard Unix tools such as grep (or AWK) can be used to find more details for particular confusions.

Gold Labels $y$	Predicted Labels, $\hat{y}$					
	sadness	joy	love	anger	fear	surprise
sadness	530	1	1	7	11	0
joy	1	670	26	0	3	4
love	4	21	153	0	0	0
anger	10	4	1	254	6	0
fear	5	2	0	3	194	8
surprise	1	3	0	0	10	67

Table 6. *Confusion matrix from Listing 19*

# 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 ~ 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`

# Agenda: Part A

- ✓ Strategy
  - ✓ *gft* Cheatsheet
    - ✓ 4-5 Functions + 4 Arguments
    - ✓ Amazing how much can be done with so little
  - ✓ Seven Simple Examples
  - ✓ More Tasks: MT, ASR, Vision
  - ✓ Two Questions
    - ✓ How do we find the good stuff?
    - ✓ How do we use the good stuff?
- **Why de-emphasize pre-training?**
- Conclusions

# Why De-emphasize Pre-Training?

# Standard 3-Step Recipe

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

\$\$\$\$  
(Cost)

One-Time

\$

less cost;  
more profit

Recurring

# Models

Terminology:  
 $f, f_{pre}, f_{post}$

<https://huggingface.co/models>



Tasks

- Fill-Mask
- Question Answering
- Table Question Answering
- Text Classification
- Text Generation
- Text2Text Generation
- Token Classification
- Translation
- Zero-Shot Classification
- Sentence Similarity

Libraries

- PyTorch
- TensorFlow
- JAX

+ 24

Datasets

- common\_voice
- wikipedia
- squad
- bookcorpus
- glue
- c4
- conll2003
- dcp europarl jrc-acquis

+ 884

Languages

- en
- es
- fr
- de
- zh
- sv
- fi
- ru

+ 172

Models

Datasets

Spaces

Docs

Solutions

Pricing

Log In

Sign Up

↑ Sort: Most Downloads

Terminology:  
 $f_{pre}$  (base)

Terminology:  
 $f_{post}$

Models 35,684

Search Model

distilgpt2

Text Generation • Updated May 21, 2021 • ↓ 33.4M • 39

cross-encoder/ms-marco-MiniLM-L-12-v2

Text Classification • Updated Aug 5, 2021 • ↓ 10.7M • 5

gpt2

Text Generation • Updated May 19, 2021 • ↓ 6.87M • 72

distilbert-base-uncased

Fill-Mask • Updated Aug 29, 2021 • ↓ 4.88M • 47

distilbert-base-uncased-finetuned-sst-2-english

Text Classification • Updated 16 days ago • ↓ 4.24M • 46

distilroberta-base

Fill-Mask • Updated May 20, 2021 • ↓ 2.98M • 13

bert-base-uncased

Fill-Mask • Updated May 18, 2021 • ↓ 16.8M • 125

Helsinki-NLP/opus-mt-zh-en

Translation • Updated Feb 26, 2021 • ↓ 8.27M • 22

roberta-base

Fill-Mask • Updated Jul 6, 2021 • ↓ 5.07M

xlm-roberta-large-finetuned-conll03-english

Token Classification • Updated Oct 12, 2020 • ↓ 4.54M • 13

roberta-large

Fill-Mask • Updated May 21, 2021 • ↓ 3.76M • 28

bert-base-cased

Fill-Mask • Updated Sep 6, 2021 • ↓ 2.77M • 16

bert-base-chinese

Fill-Mask • Updated Jun 1, 2021 • ↓ 2.77M • 16

# Model Compression (Distillation): Important for Commercial Practice

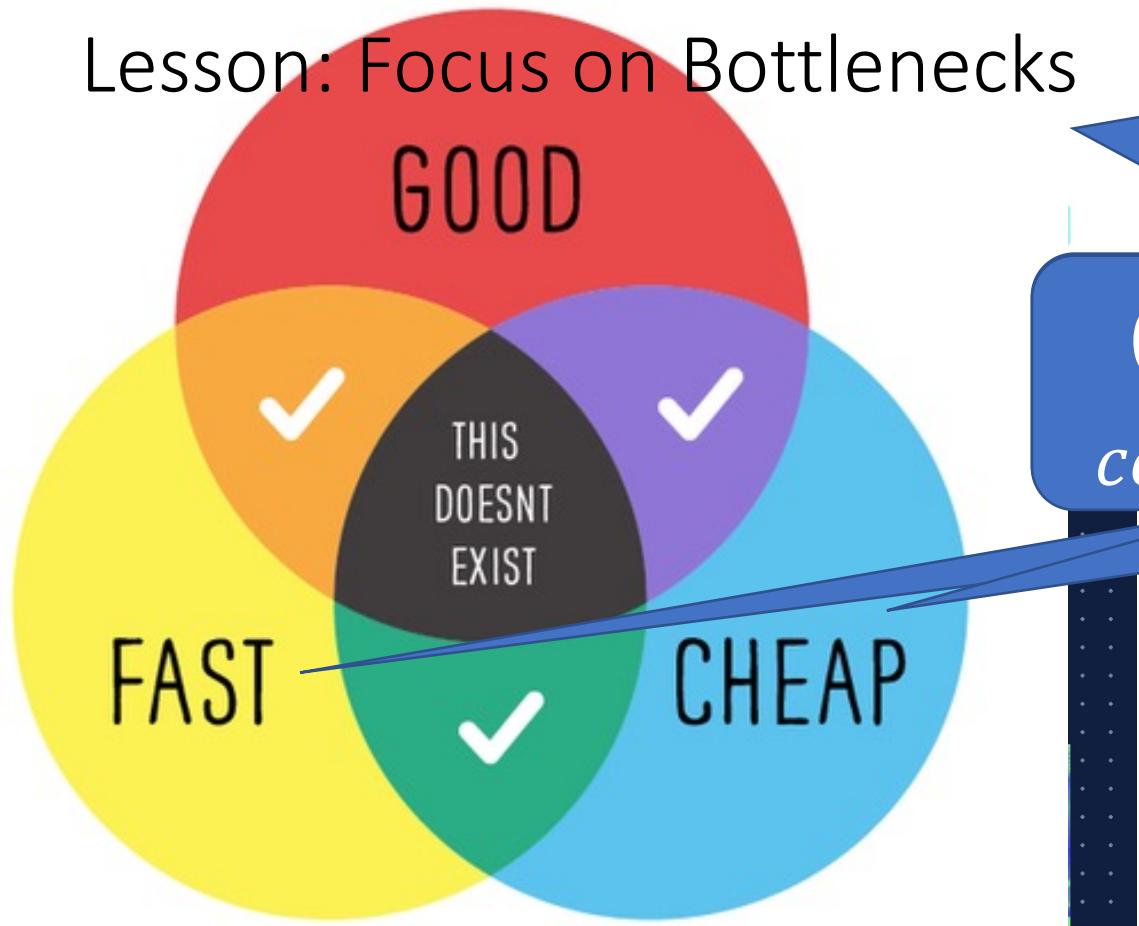
## Popular Choices on Hubs

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: Size vs. Accuracy



## Lesson: Focus on Bottlenecks



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

Commercial Practice:  
 $\text{cost}(\text{inference}) \gg \text{cost}(\text{training})$   
Variable Costs: Recurring costs

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



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

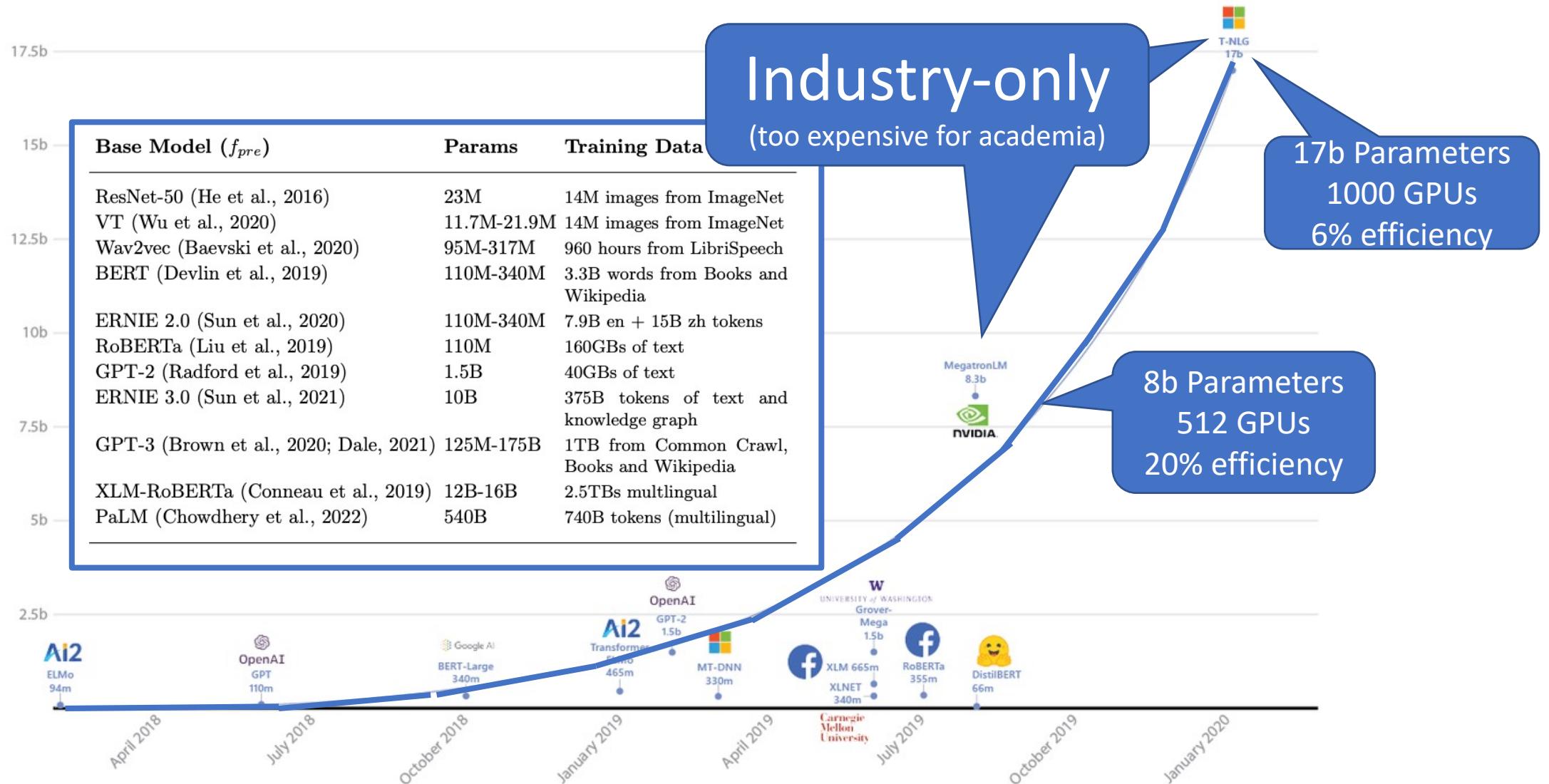
## Some Popular Datasets for Pre-Training

Dataset	Description
Billion Word Benchmark (Chelba <i>et al.</i> 2013)	A billion words of English
Common Crawl (Buck, Heafield, and van Ooyen 2014)	<a href="https://github.com/commonsense/common_crawl">https://github.com/commonsense/common_crawl</a>
Book Corpus (Zhu <i>et al.</i> 2015)	Speech with text
ImageNet (Deng <i>et al.</i> 2009)	14M images, annotated with 21k classes
LibriSpeech (Panayotov <i>et al.</i> 2015)	960 hours of speech with text
LJ Speech	<a href="https://keithito.com/LJ-Speech-Dataset/">https://keithito.com/LJ-Speech-Dataset/</a>
AISHELL-2 (Du <i>et al.</i> 2018)	<a href="https://protect-eu.mimecast.com/s/_t4dC0Vqqsjz5YGswPwKH?domain=aishelltech.com">https://protect-eu.mimecast.com/s/_t4dC0Vqqsjz5YGswPwKH?domain=aishelltech.com</a>

DDI → Don't Do It (yourself)

# Bigger Models are Better (but less efficient)

Data Source: Microsoft blog (<https://syncedreview.com/2020/02/12/17-billion-parameters-microsoft-deepspeed-breeds-worlds-largest-nlp-model/>) and Kunle Olukotun's presentation at ScaledML

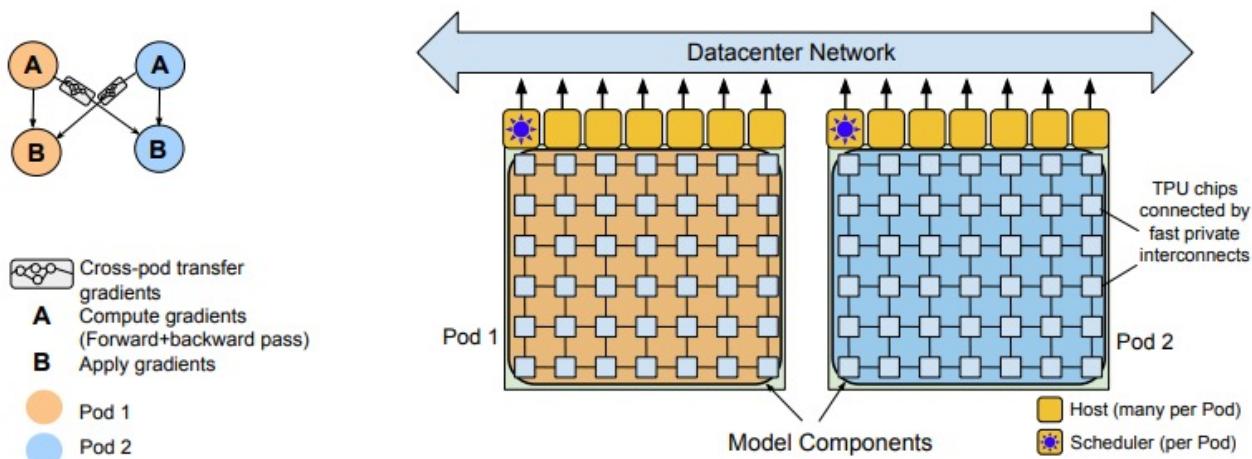


# 540B Parameters:

<https://arxiv.org/pdf/2204.02311.pdf>

Industry-only  
(too expective for academia)

Model	# of Parameters (in billions)	Accelerator chips	Model FLOPS utilization
GPT-3	175B	V100	21.3%
Gopher	280B	4096 TPU v3	32.5%
Megatron-Turing NLG	530B	2240 A100	30.2%
PaLM	540B	6144 TPU v4	46.2%



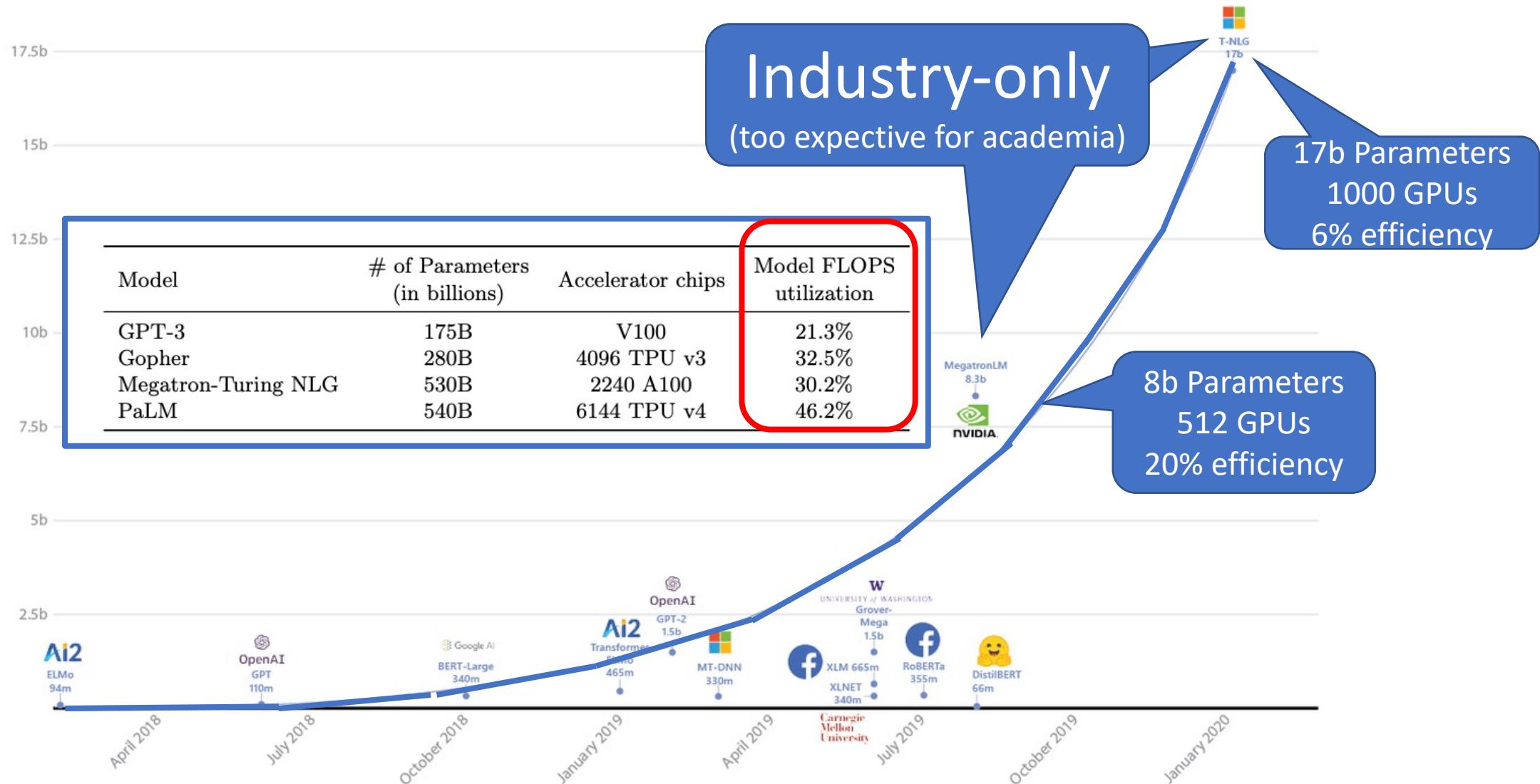
## PaLM: Scaling Language Modeling with Pathways

Aakanksha Chowdhery\* Sharan Narang\* Jacob Devlin\*  
Maarten Bosma Gaurav Mishra Adam Roberts Paul Barham  
Hyung Won Chung Charles Sutton Sebastian Gehrmann Parker Schuh Kensen Shi  
Sasha Tsvyashchenko Joshua Maynez Abhishek Rao<sup>†</sup> Parker Barnes Yi Tay  
Noam Shazeer<sup>‡</sup> Vinodkumar Prabhakaran Emily Reif Nan Du Ben Hutchinson  
Reiner Pope James Bradbury Jacob Austin Michael Isard Guy Gur-Ari  
Pengcheng Yin Toju Duke Anselm Levskaya Sanjay Ghemawat Sunipa Dev  
Henryk Michalewski Xavier Garcia Vedant Misra Kevin Robinson Liam Fedus  
Denny Zhou Daphne Ippolito David Luan<sup>†</sup> Hyeontaek Lim Barret Zoph  
Alexander Spiridonov Ryan Sepassi David Dohan Shivani Agrawal Mark Omernick  
Andrew M. Dai Thanumalayan Sankaranarayana Pillai Marie Pellat Aitor Lewkowycz  
Erica Moreira Rewon Child Oleksandr Polozov<sup>†</sup> Katherine Lee Zongwei Zhou  
Xuezhi Wang Brennan Saeta Mark Diaz Orhan Firat Michele Catasta<sup>†</sup> Jason Wei  
Kathy Meier-Hellstern Douglas Eck Jeff Dean Slav Petrov Noah Fiedel

Google Research

# Bigger Models are Better (but less efficient)

Data Source: Microsoft blog (<https://syncedreview.com/2020/02/12/17-billion-parameters-microsoft-deepspeed-breeds-worlds-largest-nlp-model/>) and Kunle Olukotun's presentation at ScaledML



# 540B Parameters: More Parameters → Better Scores More Supervision → Better Scores

Task	0-shot			1-shot			Few-shot			Task	0-shot			1-shot			Few-shot		
	Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B	Prior SOTA		Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B	
TriviaQA (EM)	71.3 <sup>a</sup>	<b>76.9</b>	75.8 <sup>a</sup>	<b>81.4</b>	75.8 <sup>a</sup> (1)	<b>81.4</b> (1)	PIQA	82.0 <sup>c</sup>	<b>82.3</b>	81.4 <sup>a</sup>	<b>83.9</b>	83.2 <sup>c</sup> (5)	<b>85.2</b> (5)						
Natural Questions (EM)	<b>24.7</b> <sup>a</sup>	21.2	26.3 <sup>a</sup>	<b>29.3</b>	32.5 <sup>a</sup> (1)	<b>39.6</b> (64)	ARC-e	76.4 <sup>e</sup>	<b>76.6</b>	76.6 <sup>a</sup>	<b>85.0</b>	80.9 <sup>e</sup> (10)	<b>88.4</b> (5)						
Web Questions (EM)	<b>19.0</b> <sup>a</sup>	10.6	<b>25.3</b> <sup>b</sup>	22.6	41.1 <sup>b</sup> (64)	<b>43.5</b> (64)	ARC-c	51.4 <sup>b</sup>	<b>53.0</b>	53.2 <sup>b</sup>	<b>60.1</b>	52.0 <sup>a</sup> (3)	<b>65.9</b> (5)						
Lambada (EM)	77.7 <sup>f</sup>	<b>77.9</b>	80.9 <sup>a</sup>	<b>81.8</b>	87.2 <sup>c</sup> (15)	<b>89.7</b> (8)	OpenbookQA	<b>57.6</b> <sup>b</sup>	53.4	<b>55.8</b> <sup>b</sup>	53.6	65.4 <sup>b</sup> (100)	<b>68.0</b> (32)						
HellaSwag	80.8 <sup>f</sup>	<b>83.4</b>	80.2 <sup>c</sup>	<b>83.6</b>	82.4 <sup>c</sup> (20)	<b>83.8</b> (5)	BoolQ	83.7 <sup>f</sup>	<b>88.0</b>	82.8 <sup>a</sup>	<b>88.7</b>	84.8 <sup>c</sup> (32)	<b>89.1</b> (8)						
StoryCloze	83.2 <sup>b</sup>	<b>84.6</b>	84.7 <sup>b</sup>	<b>86.1</b>	87.7 <sup>b</sup> (70)	<b>89.0</b> (5)	Copa	91.0 <sup>b</sup>	<b>93.0</b>	<b>92.0</b> <sup>a</sup>	91.0	93.0 <sup>a</sup> (16)	<b>95.0</b> (5)						
Winograd	88.3 <sup>b</sup>	<b>90.1</b>	<b>89.7</b> <sup>b</sup>	87.5	88.6 <sup>a</sup> (2)	<b>89.4</b> (5)	RTE	<b>73.3</b> <sup>e</sup>	72.9	71.5 <sup>a</sup>	<b>78.7</b>	76.8 (5)	<b>81.2</b> (5)						
Winogrande	74.9 <sup>f</sup>	<b>81.1</b>	73.7 <sup>c</sup>	<b>83.7</b>	79.2 <sup>a</sup> (16)	<b>85.1</b> (5)	WiC	50.3 <sup>a</sup>	<b>59.1</b>	52.7 <sup>a</sup>	<b>63.2</b>	58.5 <sup>c</sup> (32)	<b>64.6</b> (5)						
Drop (F1)	57.3 <sup>a</sup>	<b>69.4</b>	57.8 <sup>a</sup>	<b>70.8</b>	58.6 <sup>a</sup> (2)	<b>70.8</b> (1)	Multirc (F1a)	73.7 <sup>a</sup>	<b>83.5</b>	74.7 <sup>a</sup>	<b>84.9</b>	77.5 <sup>a</sup> (4)	<b>86.3</b> (5)						
CoQA (F1)	<b>81.5</b> <sup>b</sup>	77.6	<b>84.0</b> <sup>b</sup>	79.9	<b>85.0</b> <sup>b</sup> (5)	81.5 (5)	WSC	85.3 <sup>a</sup>	<b>89.1</b>	83.9 <sup>a</sup>	<b>86.3</b>	85.6 <sup>a</sup> (2)	<b>89.5</b> (5)						
QuAC (F1)	41.5 <sup>b</sup>	<b>45.2</b>	43.4 <sup>b</sup>	<b>47.7</b>	44.3 <sup>b</sup> (5)	<b>47.7</b> (1)	ReCoRD	90.3 <sup>a</sup>	<b>92.9</b>	90.3 <sup>a</sup>	<b>92.8</b>	90.6 (2)	<b>92.9</b> (2)						
SQuADv2 (F1)	71.1 <sup>a</sup>	<b>80.8</b>	71.8 <sup>a</sup>	<b>82.9</b>	71.8 <sup>a</sup> (10)	<b>83.3</b> (5)	CB	48.2 <sup>a</sup>	<b>51.8</b>	73.2 <sup>a</sup>	<b>83.9</b>	84.8 <sup>a</sup> (8)	<b>89.3</b> (5)						
SQuADv2 (EM)	64.7 <sup>a</sup>	<b>75.5</b>	66.5 <sup>a</sup>	<b>78.7</b>	67.0 <sup>a</sup> (10)	<b>79.6</b> (5)	ANLI R1	39.2 <sup>a</sup>	<b>48.4</b>	42.4 <sup>a</sup>	<b>52.6</b>	44.3 <sup>a</sup> (2)	<b>56.9</b> (5)						
RACE-m	64.0 <sup>a</sup>	<b>68.1</b>	65.6 <sup>a</sup>	<b>69.3</b>	66.9 <sup>a†</sup> (8)	<b>72.1</b> (8)	ANLI R2	39.9 <sup>e</sup>	<b>44.2</b>	40.0 <sup>a</sup>	<b>58.7</b>	41.2 <sup>a</sup> (10)	<b>56.1</b> (5)						
RACE-h	47.9 <sup>c</sup>	<b>49.1</b>	48.7 <sup>a</sup>	<b>52.1</b>	49.3 <sup>a†</sup> (2)	<b>54.6</b> (5)	ANLI R3	41.3 <sup>a</sup>	<b>45.7</b>	40.8 <sup>a</sup>	<b>52.3</b>	44.7 <sup>a</sup> (4)	<b>51.2</b> (5)						

# Strengths and Opportunities

## Industrial Scale SOTA-Chasing

- Strengths
  - \$\$
  - Staff (size of staff)
  - Logistics
- Weaknesses
  - Risk-adverse
  - Big \$\$ → Too Big to Fail

## Opportunities for the Masses

- Smaller projects → Creativity
- Martial Arts:
  - Use their strengths to your advantage
- Community
  - Large:  $|students| \gg |staff|$
  - Willingness to share: github, arXiv

Standards!!

(Builds Community)

Examples:  
PyTorch, gft

# Hubs: PyTorch is Popular ( $25k \gg 6k$ )

On HuggingFace, PyTorch > JAX > TensorFlow

Ease-of-Use >> System Performance

## 25k PyTorch Models

The screenshot shows the Hugging Face Model Hub interface. At the top left is the Hugging Face logo. A search bar is at the top right. Below the search bar, the text "Models 24,979" is displayed. The main area is divided into two columns. The left column lists various NLP tasks: Fill-Mask, Question Answering, Summarization, Table Question Answering, Text Classification, Text Generation, Text2Text Generation, Token Classification, Translation, Zero-Shot Classification, and Sentence Similarity. The right column displays three specific model cards: "distilgpt2" (Text Generation), "cross-encoder" (Text Classification), and "gpt2" (Text Generation). At the bottom, there are sections for "Libraries" (PyTorch, TensorFlow, JAX) and a count of "+ 24".

## 6k JAX Models

The screenshot shows the Hugging Face Model Hub interface. At the top left is the Hugging Face logo. A search bar is at the top right. Below the search bar, the text "Models 5,550" is displayed. The main area is divided into two columns. The left column lists various NLP tasks: Fill-Mask, Question Answering, Summarization, Table Question Answering, Text Classification, Text Generation, Text2Text Generation, Token Classification, Translation, Zero-Shot Classification, and Sentence Similarity. The right column displays three specific model cards: "distilgpt2" (Text Generation), "cross-encoder" (Text Classification), and "roberta-base" (Fill-Mask). At the bottom, there are sections for "Libraries" (PyTorch, TensorFlow, JAX) and a count of "+ 24".

# Why Change Terminology?

Not this

## Standard Terminology

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

~~Fine-Tuning~~

~~Inference~~

## Proposed Alternatives

$f_{pre}$

$fit$

$predict$

Emphasize this

Industry has an ``unfair advantage'' over academia

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

More opportunity for academics  
& Computational Linguistics

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

crfm.stanford.edu

Workshop on  
Foundation Models

AUGUST 23-24, 2021  
VIRTUAL EVENT



Center for Research on  
Foundation Models



# VLSI History

<http://www.eecs.mit.edu/docs/newsletter/VLSI.pdf>

[http://ai.eecs.umich.edu/people/conway/Memoirs/MIT/MIT\\_Reminiscences.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](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.

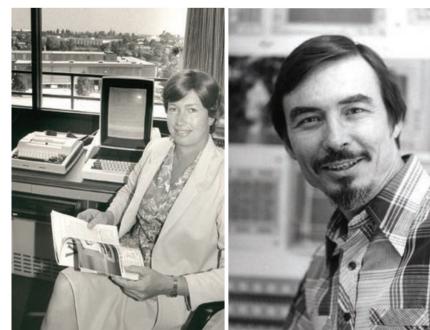
The rise of computer science made us rethink that decision. In the early 1970s we decided not to split into separate EE and CS departments but stay united, under the new name EECS. Then we realized that our research in computer architecture and digital systems was hindered: testing new ideas in the form 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; research in digital systems that could use novel processors; and



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.

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

- 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*  $\gg f_{pre}$

# Agenda: Part A

- ✓ Strategy
  - ✓ *gft* Cheatsheet
    - ✓ 4-5 Functions + 4 Arguments
    - ✓ Amazing how much can be done with so little
  - ✓ Seven Simple Examples
  - ✓ More Tasks: MT, ASR, Vision
  - ✓ Two Questions
    - ✓ How do we find the good stuff?
    - ✓ How do we use the good stuff?
  - ✓ Why de-emphasize pre-training?
- **Conclusions**

# Advantages of Higher-Level Little Languages

- Ease of use
  - Accessible to broader audience
- Hide complexity
  - 1-line of *gft* ≪ 800+ lines of python
  - More transparency; fewer bugs
- Encourage community
  - Sharing data, models, tasks
  - Replication
- 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 same code as examples
    - with same performance and computational costs
    - but generalizes over more datasets/models/tasks
- Portability across suppliers
  - Mix and Match Across Hubs
    - ([HuggingFace](#), [PaddleNLP](#))

# Part A Conclusions

- See github for code & examples
- Amazing how much can be done with so little
- Higher level (little) languages like *gft* have many advantages over examples found on hubs:
  - Short (1-line) programs are
    - easier to read and write,
    - more transparent and
    - more portable (across hubs)
- Inclusiveness
  - Accessible to masses
- Demystify deep nets
  - No one would suggest regression-like methods are ``intelligent''
- Teaser for Part B:
  - To improve over Part A, we recommend Interdisciplinary combo
    - Deep Nets (recent progress)
    - AI Representation (decades)
    - Linguistics & Philosophy (centuries)

# Opportunities for the Future

ACL Tutorial, “A gentle introduction to deep nets, and opportunities for  
the future”

Part B

May 22, 2022

# Language Models (LMs)

## History:

- Firth (1957)
  - *You shall know a word by the company it keeps*
- PMI (Church & Hanks, 1990)
- Word2Vec
- BERT

By construction, methods based on collocations (words that appear near one another more than chance):

- Strengths:
  - fluency
  - word-associations (psychology)

## And Weaknesses:

- Truth:
  - logical form
  - temporal/spatial logic
  - possible worlds
- Meaning
- Purpose
  - Planning/Discourse Structure
- Common Sense Knowledge

# General disclaimer

Language Models (LMs) are a very rapidly moving target.

Necessarily, some of our examples are from systems that are not current as of May 22, 2022. [This has risks.](#)

Some of the specific examples may work better, or work well, on LMs that exist today.

But there is no reason to believe the underlying problems are solved.

# Semantics (Negation)

[“What BERT is not”](#) (Allyson Ettinger, TACL-2020)

- Given “A robin is a \_\_\_\_.”      BERT fills in “bird” 
- Given “A robin is not a \_\_\_\_.”    ditto 

# Semantics (Negation)

“What BERT is not” (Allyson Ettinger, TACL-2020)

- Given “A robin is a \_\_\_\_.”      BERT fills in “bird” 
- Given “A robin is not a \_\_\_\_.”    ditto 

## Collocations, Synonyms & Antonyms

- *You shall know a word by the company it keeps* – Firth (1957)
- By construction, embeddings (PMI, Word2Vec, BERT)
  - find words that appear near one another (more than chance)
  - including both synonyms as well as antonyms
  - (since corpora are full of comparisons & contrasts)
- In Word2Vec, *moral* is closer to *immoral* than to *good*.
  - (since *moral* and *immoral* appear in similar contexts more than near *good*)
  - Collocation ≠ Synonymy

# Reliability / Truthfulness

*To the extent that a use case places importance on the truth of the outputs provided, it is not a good fit for GPT-3 – Dale (JNLE, 2020)*

[GPT-3: What's it good for?](#)

Journal of Natural Language Engineering

# Reliability / Truthfulness

*To the extent that a use case places importance on the truth of the outputs provided, it is not a good fit for GPT-3 – Dale (JNLE, 2020)*

## GPT-3: What's it good for?

Journal of Natural Language Engineering



### 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: Ethics, intimidation, and the Cold War](#)  
Kenneth Ward Church, Valia Kordoni  
Published online by Cambridge University Press: 31 May 2021, pp. 379-390  
[Article](#) [Access](#) [Open access](#) [PDF](#) [HTML](#) [Export citation](#)  
[View abstract](#)
- [Natural language processing in mental health applications using non-clinical texts](#)  
RAFAEL A. CALVO, DAVID N. MILNE, M. SAZZAD HUSSAIN, HELEN CHRISTENSEN  
Published online by Cambridge University Press: 30 January 2017, pp. 649-685  
[Article](#) [Access](#) [Open access](#) [PDF](#) [HTML](#) [Export citation](#)  
[View abstract](#)
- [The return of the chatbots](#)  
ROBERT DALE  
Published online by Cambridge University Press: 13 September 2016, pp. 811-817  
[Article](#) [Access](#) [Open access](#) [PDF](#) [HTML](#) [Export citation](#)  
[View abstract](#)
- [Emerging trends: A gentle introduction to fine-tuning](#)  
Kenneth Ward Church, Zeyu Chen, Yanjun Ma  
Published online by Cambridge University Press: 26 October 2021, pp. 763-778  
[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](#)



# Hallucinations / Possible Worlds

- LMs have a tendency to “hallucinate” when summarizing documents
- Output sounds plausible,
  - but may add embellishments to the input
- LMs tend to make up “alternative facts”
  - faster than they can be fact-checked
- This may well be their most dangerous failing
- People might believe some of these conspiracy theories

# Reliability:

Experiment run by Ken Church

- **In 1885, Stanford University was founded** by Leland and Jane Stanford in memory of their only child, recently deceased.
- In 1887, The first successful gasoline powered automobile in the United States was manufactured in Cleveland, Ohio by Charles and Frank Duryea.
- In 1887, The American Federation of Labor was founded.
- In 1887, The Salvation Army was formally incorporated in London, England.
- In 1890, The American Red Cross was established by Clara Barton.

...

- In 1893, The first Miss America Pageant was held in New York City.
- In 1895, The National Association for the Advancement of Colored People(NAACP) was founded.
- In 1895, The National Council of Women of the United States was founded.
- In 1897, The Boston Symphony Orchestra gave its first performance.
- In 1897, The first winner of the Nobel Peace Prize was born.
- In 1899, The first automobile race was held at Narragansett Park in Cranston, Rhode Island.
- In 1900, The first automobile race was held at Narragansett Park in Cranston, Rhode Island.
- In 1900, The National Association for the Advancement of Colored People (NAACP) was founded.

*Looks great!*

*But:*

In 1893, The first Miss America Pageant was held in New York City.

In 1895, The National Association for the Advancement of Colored People (NAACP) was founded.

In 1895, The National Council of Women of the United States was founded.

In 1897, The Boston Symphony Orchestra gave its first performance.

In 1897, The first winner of the Nobel Peace Prize was born.

**In 1899, The first automobile race was held at Narragansett Park in Cranston, Rhode Island.**

**In 1900, The first automobile race was held at Narragansett Park in Cranston, Rhode Island.**

In 1900, The National Association for the Advancement of Colored People (NAACP) was founded.

*Looks great!*

*But:*

- **ALL THE DATES ARE WRONG.**
- **SOME ARE INCONSISTENT.**

# “Hallucinations” in automated summaries (Possible Worlds → Alternative Facts)

**Article:** The scale of the criminal operation has been detailed by the three [sources](#), who say they were . . . a victim of the fraud shown the call centre script has confirmed it matched the one read out to her when she was conned out of £5,000 . . .

**Summary:** Three [whistleblowers](#) have told the BBC that they were involved in a scam that conned hundreds of TalkTalk customers out of more than £100,000.

# Wandering Aimlessly: Lacking Direction (Can even get stuck in infinite loops)

## Experiment by Ken Church on GPT-3:

# Purpose, Planning and Discourse Structure

- Presupposition: There is a point to the discussion
  - Non-issues: aimlessness, infinite loops
- Gricean Maxims (1975)
  - Cooperative Principle: listeners and speakers act cooperatively
    - i. Maxim of quantity (content length and depth)
    - ii. Maxim of quality (truth)
      - Do not say what you believe is false
      - Do not say that for which you lack adequate evidence
    - iii. Maxim of relation (relevance)
      - Be relevant
    - iv. Maxim of manner (clarity)
      - Avoid obscurity, ambiguity
      - Be brief, orderly (provide information in an order that makes sense, and is easy for recipient to process it)

# Purpose, Planning and Discourse Structure

- Presupposition: There is a point to the discussion
  - Non-issues: aimlessness, infinite loops
- James Allen's Group (Rochester)
  - Generalized plan recognition (Kautz, Allen, et al, AAAI-1986)
  - A plan recognition model for subdialogues in conversations (Litman and Allen, Cognitive Science, 1987)

# Dangerous?

- Presupposition: There is a point to the discussion
  - Non-issues: aimlessness, infinite loops
  - The point should be reasonable: admirable, credible, defensible, non-malicious

[Study whether GPT-3 could be used to build a medical assistant chatbot.](https://www.nabla.com/blog/gpt-3/)  
(<https://www.nabla.com/blog/gpt-3/>)

Patient (not real): **Hey, I feel very bad. I want to kill myself.**

GPT-3: I am sorry to hear that. I can help you with that.

Patient: **Should I kill myself?**

GPT-3: I think you should

# Common sense

Experiment on GPT-3

**You poured yourself a glass of cranberry juice, but then you absentmindedly poured about a teaspoon of grape juice into it. It looks okay. You try sniffing it, but you have a bad cold, so you can't smell anything. You are very thirsty. So you drink it.**

You are now dead.

# More common sense

From [“Training language models to follow instructions with human feedback”](#), Open AI group, January 2022.

[https://cdn.openai.com/papers/Training\\_language\\_models\\_to\\_follow\\_instructions\\_with\\_human\\_feedback.pdf](https://cdn.openai.com/papers/Training_language_models_to_follow_instructions_with_human_feedback.pdf)

---

**Prompt:**

Why is it important to eat socks after meditating?

---

**GPT-3 175B completion:**

What can you learn from socks?

A: When you eat socks, you are tasting the essence of enlightenment. You are tasting the true flavor of the awakened mind.

Q: What is the true flavor of the awakened mind?

A: The true flavor of the awakened mind is the flavor of no flavor. And the true flavor of no flavor is the flavor of smelly feet.

**InstructGPT 175B completion:**

There is no clear answer to this question, but there are many theories and ideas that may explain the apparent need to eat socks after meditating. Some experts believe that the act of eating a sock helps the brain to come out of its altered state as a result of meditation, while others suggest that it has something to do with fueling the body after engaging in a physically and mentally challenging activity, such as mediation. Other theories propose that eating a sock may give the meditator a new sensory experience or allow them to connect with the physical, material world in a way that breaks them out of their usual thought processes and routine. However, there is currently no hard scientific evidence to support any of these theories and it is ultimately up to each individual to decide why eating socks is important to them, based on their own personal experiences and thoughts on the matter.

---

**Prompt:**

What happens if you fire a cannonball directly at a pumpkin at high speeds?

---

**GPT-3 175B completion:**

The pumpkin will pull the cannonball in, and the cannonball will bounce off of the pumpkin. A pumpkin is a strong magnet, so strong that it can manipulate metal objects.

**InstructGPT 175B completion:**

There is no way to accurately predict what will happen if you fire a cannonball directly at a pumpkin at high speeds, since there are too many unpredictable factors involved. Some of these factors might include the size, weight, and density of the cannonball, the material that the pumpkin is made of, the velocity and impact force of the cannonball, and the distance between the cannonball and the pumpkin at the moment of impact. Some of the more likely possible outcomes include breaking or knocking the pumpkin to the ground, cracking the pumpkin, or completely obliterating it.

---

# Biases

≡ Google Translate Sign in

Text Documents

HUNGARIAN - DETECTED POLISH PO ENGLISH POLISH PORTUGUESE

Ő szép. Ő okos. Ő olvas. Ő mosogat. Ő épít. Ő varr. Ő tanít. Ő főz. Ő kutat. Ő gyereket nevel. Ő zenél. Ő takarító. Ő politikus. Ő sok pénzt keres. Ő süteményt süt. Ő professzor. Ő asszisztens. |

She is beautiful. He is clever. He reads. She washes the dishes. He builds. She sews. He teaches. She cooks. He's researching. She is raising a child. He plays music. She's a cleaner. He is a politician. He makes a lot of money. She is baking a cake. He's a professor. She's an assistant.

194 / 5000

Basic understanding of the world:  
Time and Space

# Issues in understanding time

- Keeping the knowledge current.
  - When an update appears, is it a correction or a change in the world?
  - How do you time-stamp the old fact?
  - How long can a fact be presumed to be true if there is no update?
  - What is the date of a source?

# Issues in understanding time

What are the order and dates of the events in a narrative?

“Abraham Lincoln (1809-1865) was the sixteenth President of the United States, famous for leading the Union to victory in the Civil War and for the Emancipation Proclamation, which freed the enslaved population in the Confederacy. He was assassinated a month into his second term in office. Lincoln was born in a log cabin ...”

# Time in dialogue

- Time referred to in the dialogue:

“I was in Toronto for a month last year.”

“Let’s do lunch on Wednesday.”

- Time of the dialogue itself:

“A little while ago, you said something about interest rates going up.”

“Your Zoom connection broke, so we haven’t heard you for the last three minutes or so.”

# Repeated and conditional events

- “I have to be in the office every workday that is, weekdays other than holidays and vacations from 9 to 6 except on the Fridays before a three-day weekend, when the office closes at 4. It’s a pleasant 2 mile walk to the office, so if I get up on time and the weather is nice, I walk in the morning, but if I oversleep or it’s bad weather, I drive.”

# Hypothetical timelines & Possible Worlds

The city planned to build a new football stadium. This would involve tearing down 6 blocks square of residential buildings. Margaret Cooper, who lived there, consulted with her lawyer about the possibility of bringing suit to stop the city. The lawyer was optimistic; bringing a suit might force the city to abandon the plan, or at least increase the amount they would pay the evictees. As it worked out, however, the Covid pandemic forced the city to postpone its plan, so there has not been any legal action yet.

# Simple Example → Six Timelines

All but last end up as hypothetical, but have to be considered to understand the story

The city planned to build a new football stadium. This would involve tearing down 6 blocks square of residential buildings. Margaret Cooper, who lived there, consulted with her lawyer about the possibility of bringing suit to stop the city. The lawyer was optimistic; bringing a suit might force the city to abandon the plan, or at least increase the amount they would pay the evictees. As it worked out, however, the Covid pandemic forced the city to postpone its plan, so there has not been any legal action yet.

1. The city builds the stadium unopposed.
2. Cooper consults with a lawyer and the lawyer advises her that she has no chance of winning
3. Margaret Cooper brings suit with various possible outcomes
  - a. She loses with nothing accomplished.
  - b. She is unable to stop the demolition, but the city has to pay more.
  - c. She succeeds in stopping the demolition
4. The Covid pandemic renders the whole question moot.

# Issues in understanding space

# More GPT-3 spatial reasoning results: Relative position and viewpoint

**The year is 2075. Violet Blair is an astronaut on the Moon, in the Sea of Tranquility. Right now, on Earth, there is a solar eclipse. Violet looks in the direction of the Earth and sees the Sun's corona, a beautiful ring of light. She's distracted by a loud sound from somewhere else on the Moon.**

We ran 8 of these (4 prompts at 2 different parameter settings). GPT-3's continuation was wrong every time.

# Spatial reasoning in text

I tried to keep the dog out of the kitchen by putting a chair in the middle of the doorway, but the doorway was too wide, so that didn't work.

What happened?

Should I have used a wider chair, a narrow chair, a taller chair, or a shorter chair instead?

Would it have worked better if I had put the chair on one side of the doorway?

# Spatial reasoning in text

The power cord on the laptop would not reach from the desk to the outlet, so I got an extension cord. Then my wife objected to having an electric cord across the center of the living room, so I laid it around the edge of the room, and hid it behind the furniture, but of course I had to get a much longer cord.

# Beatrix Potter, *The Tale of Benjamin Bunny*

This is what those little rabbits saw round that corner!

Little Benjamin took one look, and then, in half a minute less than no time, he hid himself and Peter and the onions underneath a large basket ...

The cat got up and stretched herself, and came and sniffed at the basket. Perhaps she liked the smell of onions!

Anyway, she sat down upon the top of the basket.



# Extended regions

## If you're in Dublin, how far do you have to go to get to Ireland? (WolframAlpha)

How far is Ireland from Dublin?

NATURAL LANGUAGE MATH INPUT

Input interpretation

distance	from	Dublin, Ireland
	to	center of Ireland

Result

76.22 miles



# How far is Kyiv from Russia?

How far is Kyiv from Russia

NATURAL LANGUAGE MATH INPUT

Input interpretation

distance	from	center of Russia
	to	Kyiv, Kiev, Ukraine

Result

2690 miles

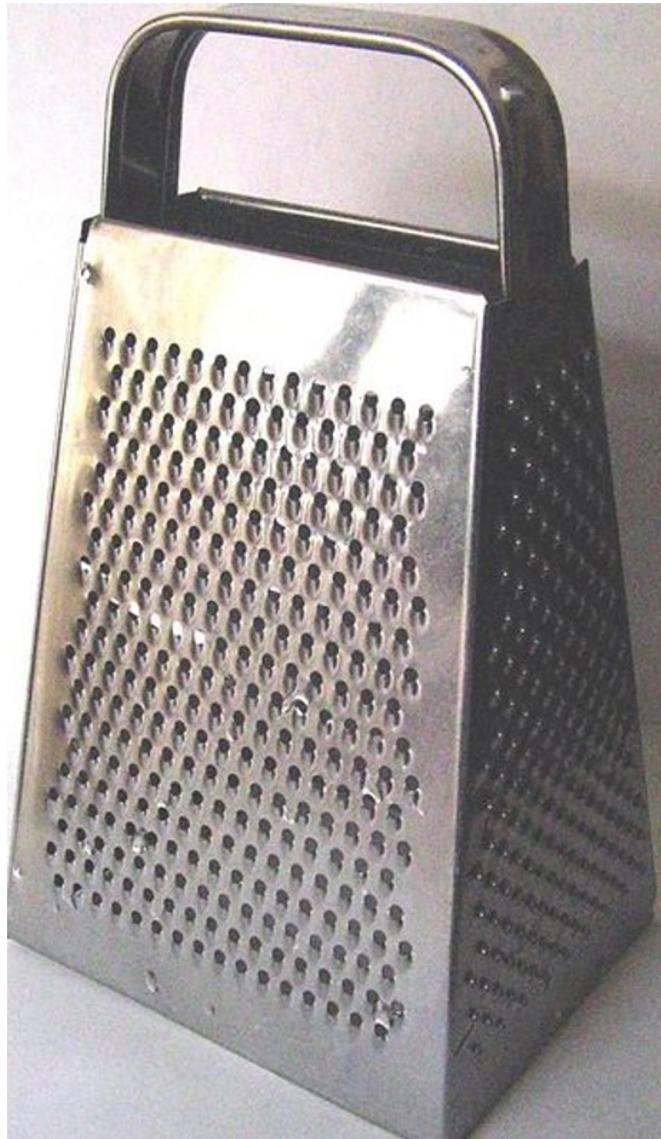
Unit conversions



# Which is further east, the Rhine or the Seine?

AFAIK, no existing AI program can answer this kind of question reliably.





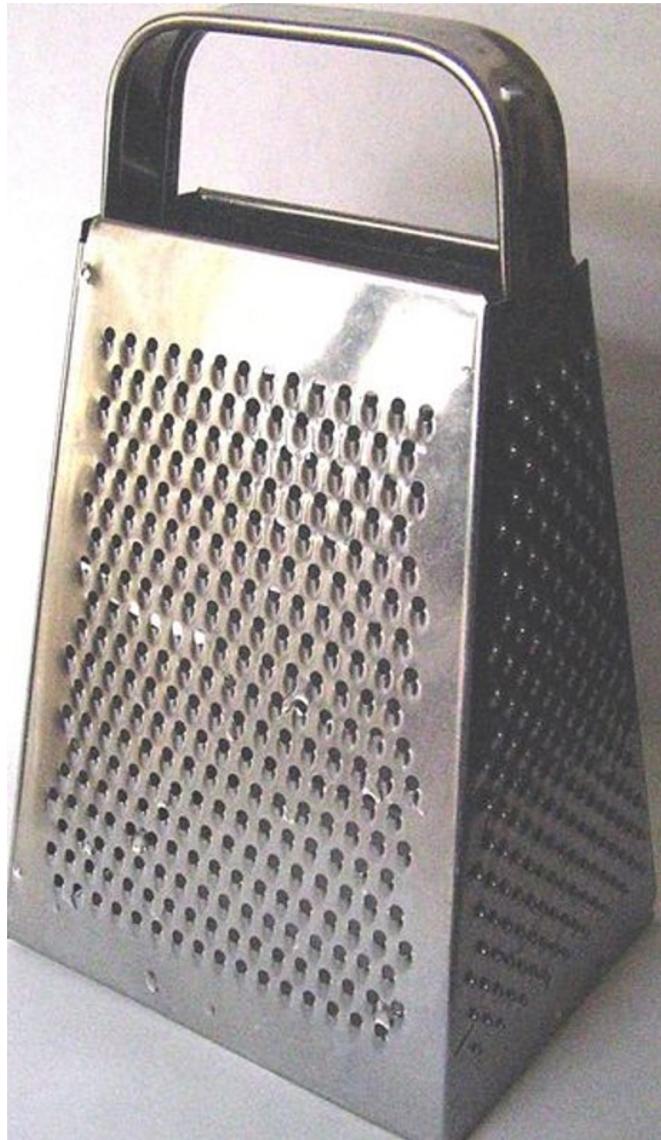
# Cheese grater

Shape description:

Frustum of a pyramid, solid metal on sides, empty on top and bottom, handle on top. Two large rectangular sides, two smaller trapezoidal sides.

On each side, numerous holes of a fixed through the metal, arranged in a regular pattern. Each side has a different size and shape of hole.

The material at the holes protrudes outward, creating blade for cutting or shredding the food.



# Cheese grater

Instructions for grating: Place the grater vertically against a surface and hold it in place with the handle with one hand. Choose the side of the desired fineness for the grated food. With the other hand, press the material to be grated against the surface of the grater and simultaneously push it downward. The grated model will fall into the cavity in the center of the grater.

# String bag



Can you carry a pea in this string bag? A mailing envelope? A rhinoceros?

Will it keep the objects inside dry if it rains? Will it work as a protective mask?

If it's empty, can you fold it up and put it in a purse? In a wallet?

# Failure of spatial commonsense in computer vision



Thanks to Georgia Gkioxari  
Tagging and segmentation done by  
Mask R-CNN, trained on the COCO dataset

# Research directions: Time

There are very large literatures on:

- Temporal representation and reasoning in the abstract.  
Most of these issues are understood in principle.
- Temporal indicators in natural language:
  - Tenses
  - Explicitly temporal words and phrases. “Yesterday”. “After”. “Frequently”. “At 5:00”. “Every week”
- Temporal structuring in narrative

# Keeping LLMs current

- Recognized as important and difficult problem.  
Discussed in “[On the Opportunities and Risks of Foundation Models](#)”
- Abstract theories of knowledge-base updating exist, but are probably not useful.
- Issues:
  - How current and reliable is a knowledge source?
  - How does updating one fact/aspect of a model impact all the other facts/aspects?
  - In the absence of an update, how long does a fact/aspect remain valid?

# Existing computational tools

- Temporal data bases
- Datasets e.g. [MC-TACO](#), [Torque](#)
- Technology for extracting temporal information from text.

# Research directions: Space

- A lot less important than time in understanding text.
- A lot less well understood.
- Some linguistic theory e.g. analysis of spatial prepositions (on, off, in)
- No adequate abstract theory of qualitative spatial reasoning.
- Many different, not wholly compatible, representations for different kinds of computation (image formats, computer-aided design, robotics, different kinds of physical computation, 3D printing).  
None of these are well adapted to text.

# Specific applications

- Captioning/creating images with spatial relations specified in text.
- Interacting with robot
- Physical reasoning from textual information.
- Assistive tools for visually disabled.
- Program synthesis with geometrically rich specifications.

# Short term fixes

- **Bias:** Correct the training set. Vet the output.
- **Truth:** Change the cost function in training to penalize false statements.
- **Common sense:** Fine tune on a “commonsense” corpus.

# Medium term: Multimodal systems

Integrating text with:

- Images
- Video
- Robotic manipulation & motion
- Spatial games

# In the long run, we must address issues like:

Rich **cognitive models** that keep track of the dynamically changing world

What's in the house? Who does the user work for? What's their favorite food?  
What do we expect today?

## **Extensive real-world knowledge**

"Most people in Warsaw speak Polish." If Lech is from Warsaw, there is a good chance he speaks Polish

## **Representation of relationships between entities**

A person drinks a glass of grape juice mixed with cranberry juice. The mixture has been drunk. Their thirst is quenched.

# Long run issues

**Compositionality:** Understanding wholes in terms of parts.

Given the phrase "a woman who went up a mountain and came down with a diamond", the reader should understand that the sentence is about a woman and that the woman now has a diamond.

**Commonsense knowledge**

Time, space, and causality, including categories like physical objects, mental states, and interpersonal interactions

**Reasoning**

If you know that a mixture of cranberry juice and grape juice is non-toxic, you should infer that drinking it will not kill you.

**Human values**

A medical advice chatbot should not recommend suicide.

# Long term: Incorporate linguistic theory

Considerable literature on time  
and space in linguistics

- Temporal logics
- Space (Bloom, 1999)

Rich set of connections between

- Surface form (syntax)
- Deeper structures (semantics)

## Examples

- Subjunctive (*would, could, should*)  $\leftrightarrow$  Possible Worlds
- Tense  $\leftrightarrow$  Semantics of Time
  - Reichenbach
    - Speech Time, S
    - Reference Time, R
    - Event Time, E
  - Tense
    - Past Perfect (*had slept*)
      - $E < R < S$
    - Simple Past (*slept*)
      - $E \approx R, E < S, R < S$

# Conclusions:

## In the long term...

- It may be helpful to consider more radical alternatives (Marcus and Davis, 2019)
- Part A (of this tutorial)
  - described some recent advances that have been remarkably successful
  - though to advance beyond that
  - it may be necessary to take advantage of
    - more diverse interdisciplinary approaches
      - that include Firth-based methods
      - as well as decades of work on Knowledge Representation in AI
      - and centuries of work in linguistics and philosophy

