Multitask Learning, and some Prompting Techniques

CMSC 473/673

Frank Ferraro

Outline

Multi-task Learning

Prompting

Remember from Earlier

Classification Types (Terminology)

Name	Number of Tasks (Domains) Labels are Associated with	# Label Types	Example
(Binary) Classification	1	2	Sentiment: Choose one of {positive or negative}
Multi-class Classification	1	> 2	Part-of-speech: Choose one of {Noun, Verb, Det, Prep, …}
Multi-label Classification	1	> 2	Sentiment: Choose multiple of {positive, angry, sad, excited, …}
Multi-task Classification	> 1	Per task: 2 or > 2 (can apply to binary or multi-class)	Task 1: part-of-speech Task 2: named entity tagging … ---------------------- Task 1: document labeling Task 2: sentiment

Multi-Label vs. Multi-Task

- These can be considered the same thing but often they're different
- "Task": a thing of interest to predict

Multi-Label vs. Multi-Task

- These can be considered the same thing but often they’re different
- “Task”: a thing of interest to predict
- Multi-label classification often involves multiple labels for the same task
  - E.g., sentiment (a tweet could be both “HAPPY” and “EXCITED”)

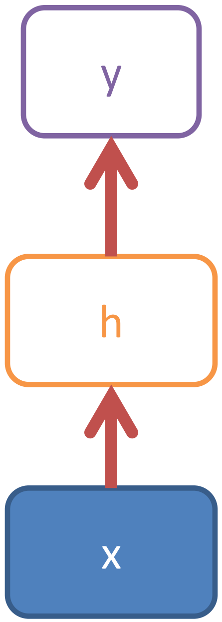
Multi-Label vs. Multi-Task

- These can be considered the same thing but often they’re different
- “Task”: a thing of interest to predict
- Multi-label classification often involves multiple labels for the same task
  - E.g., sentiment (a tweet could be both “HAPPY” and “EXCITED”)
- Multi-task learning is for different “tasks,” e.g.,
  - Task 1: Category of document (SPORTS, FINANCE, etc.)
  - Task 2: Sentiment of document
  - Task 3: Part-of-speech per token
  - Task 4: Syntactic parsing
  - …

Multi-Task Learning

Single-Task Learning

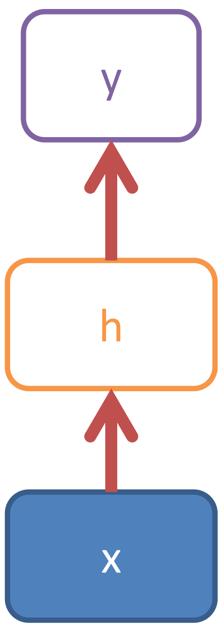
Train a system to “do one thing”
(make predictions for one task)



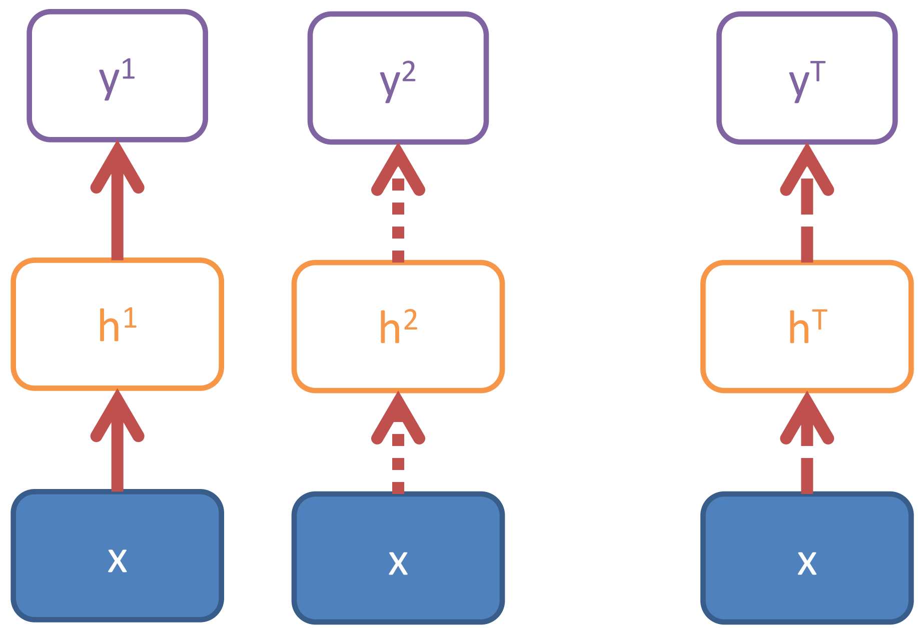
Multi-Task Learning

Single-Task Learning

Train a system to "do one thing" (make predictions for one task)



If you have multiple (T) tasks, then train multiple systems

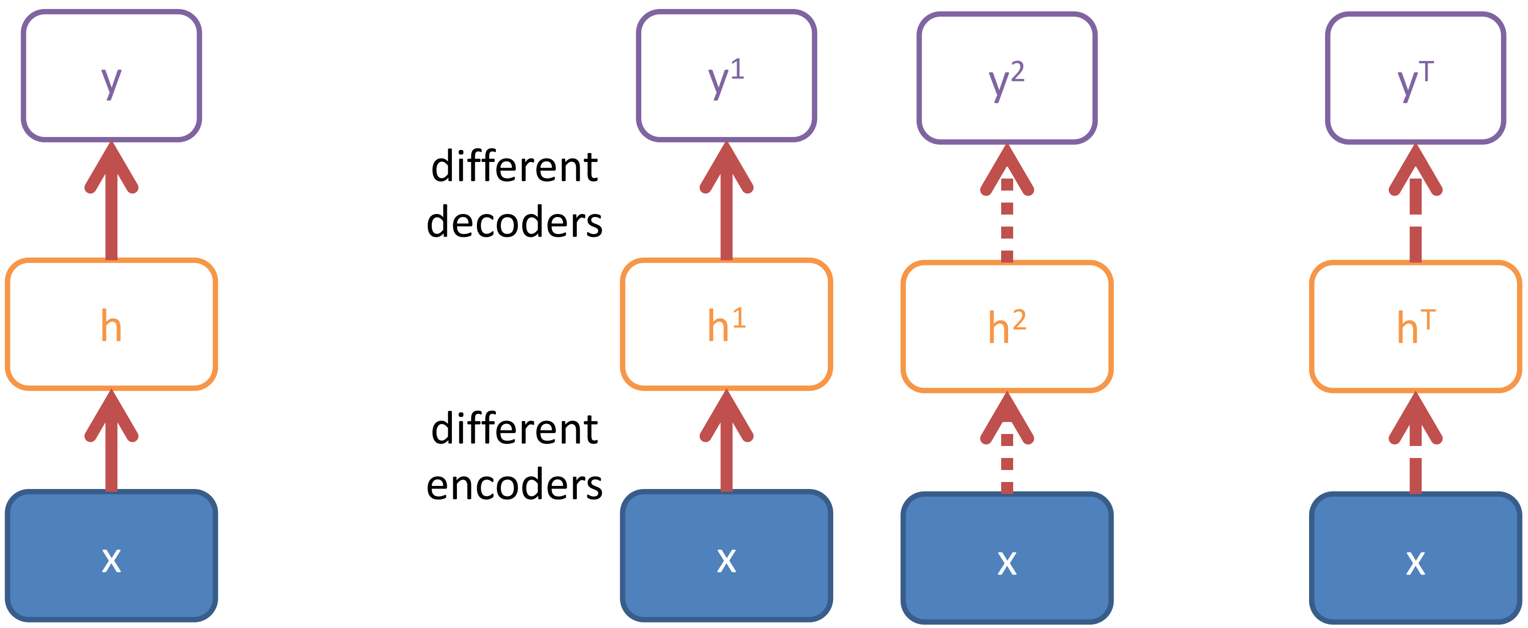


Multi-Task Learning

Single-Task Learning

Train a system to "do one thing" (make predictions for one task)

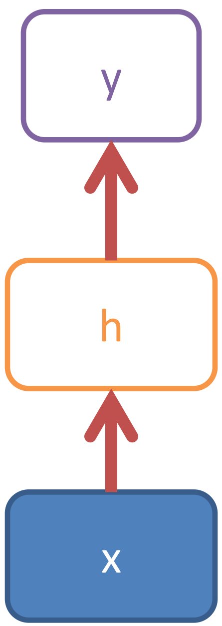
If you have multiple (T) tasks, then train multiple systems



Multi-Task Learning

Single-Task Learning

Train a system to “do one thing” (make predictions for one task)



Multi-Task Learning

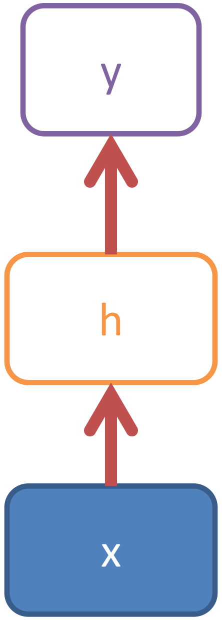
Train a system to “do multiple things” (make predictions for T different tasks)

Key idea/assumption: if the tasks are somehow related, can we leverage an ability to do task i well into an ability to do task j well?

Multi-Task Learning

Single-Task Learning

Train a system to "do one thing" (make predictions for one task)



Multi-Task Learning

Train a system to "do multiple things" (make predictions for T different tasks)

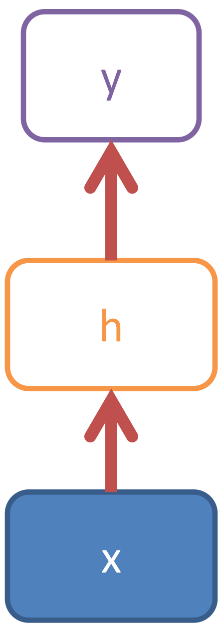
Key idea/assumption: if the tasks are somehow related, can we leverage an ability to do task i well into an ability to do task j well?

Example: could features/embeddings useful for *language modeling* (task i) also be useful for *part-of-speech tagging* (task j)?

Multi-Task Learning

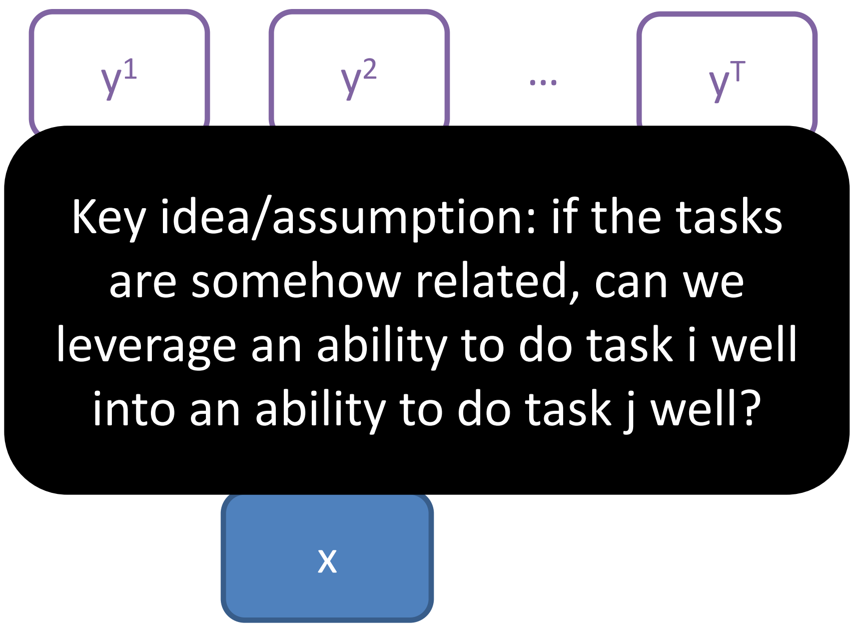
Single-Task Learning

Train a system to “do one thing” (make predictions for one task)



Multi-Task Learning

Train a system to “do multiple things” (make predictions for T different tasks)

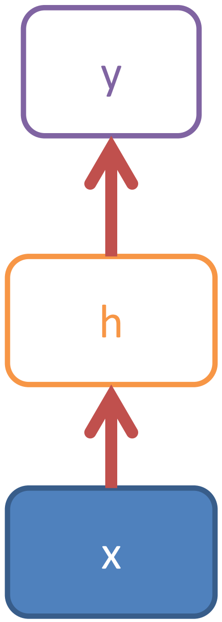


Key idea/assumption: if the tasks are somehow related, can we leverage an ability to do task i well into an ability to do task j well?

Multi-Task Learning

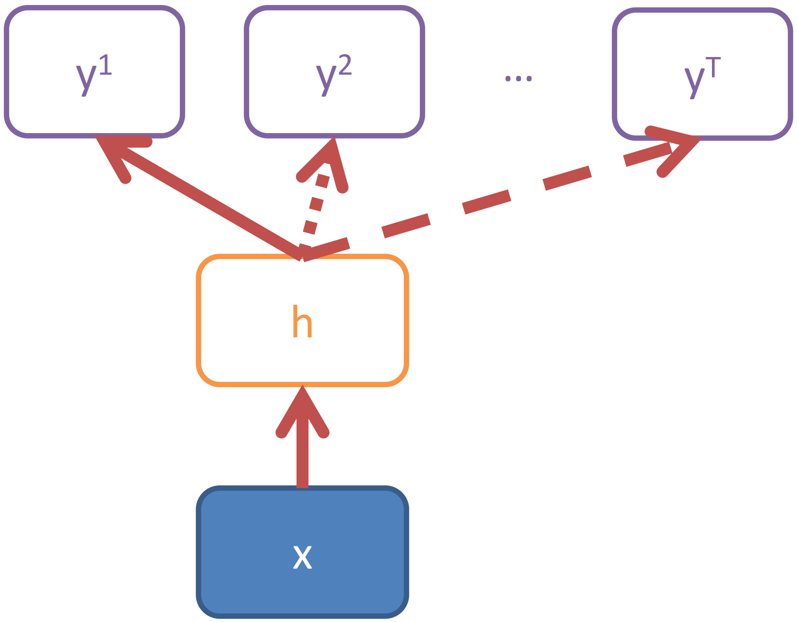
Single-Task Learning

Train a system to "do one thing" (make predictions for one task)



Multi-Task Learning

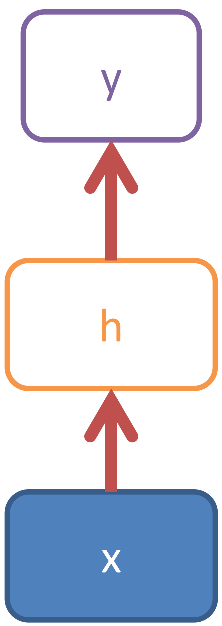
Train a system to "do multiple things" (make predictions for T different tasks)



Multi-Task Learning

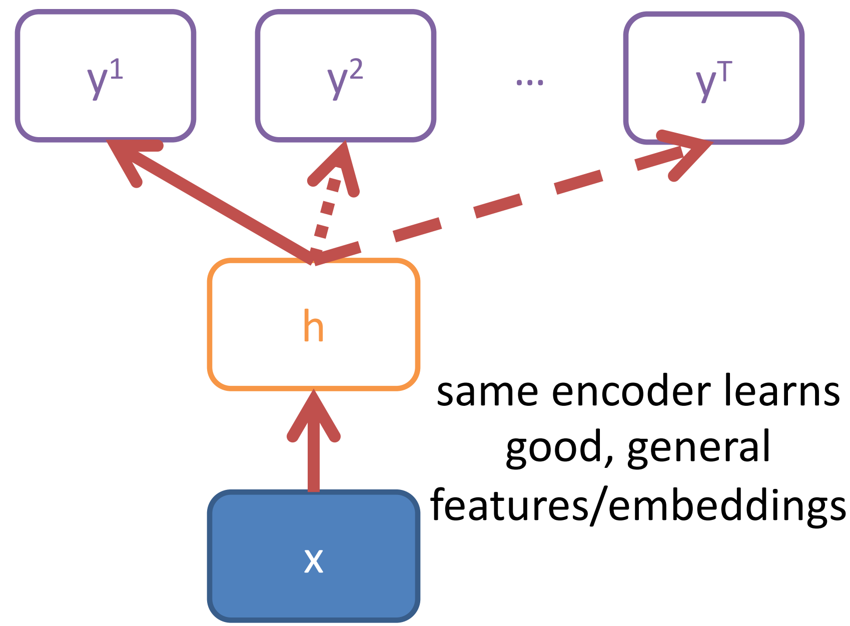
Single-Task Learning

Train a system to “do one thing” (make predictions for one task)



Multi-Task Learning

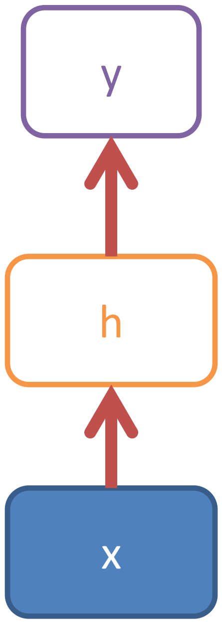
Train a system to “do multiple things” (make predictions for T different tasks)



Multi-Task Learning

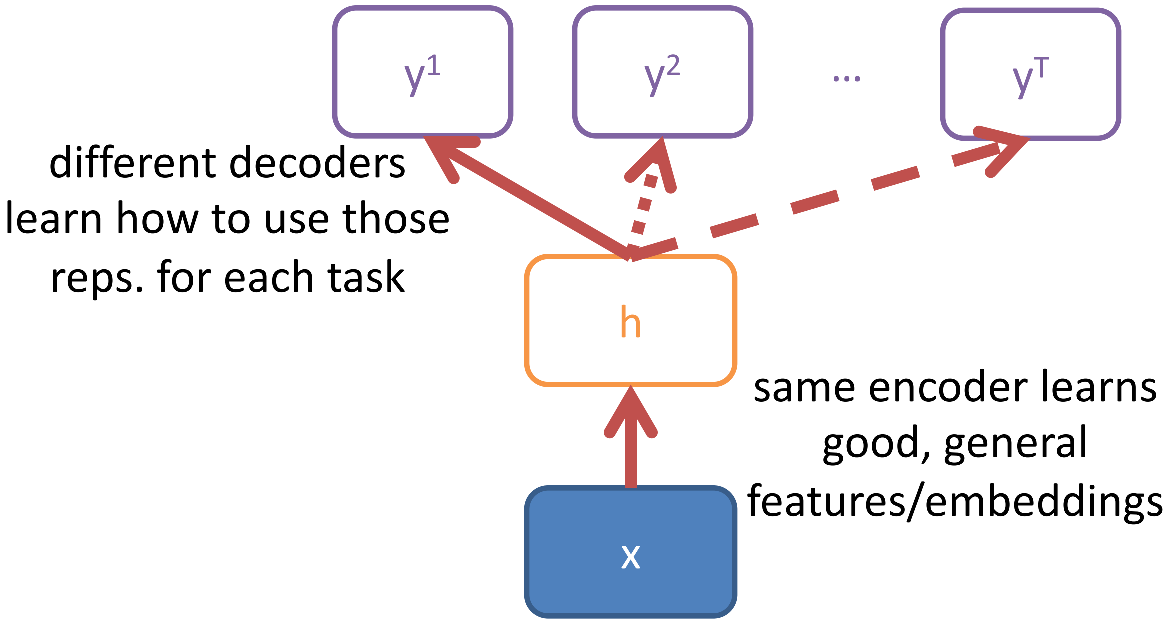
Single-Task Learning

Train a system to “do one thing” (make predictions for one task)



Multi-Task Learning

Train a system to “do multiple things” (make predictions for T different tasks)



General Multi-Task Training Procedure

Given:

T different corpora $C_1, \dots C_T$ for tasks

$$C_t = \{(x_1^t, y_1^t), \dots, (x_{N_t}^t, y_{N_t}^t)\}$$

Encoder E and T different decoders $D_1, \dots D_T$

These have weights (parameters) you need to learn

General Multi-Task Training Procedure

Given:

T different corpora $C_1, \dots C_T$ for tasks

$$C_t = \{(x_1^t, y_1^t), \dots, (x_{N_t}^t, y_{N_t}^t)\}$$

Encoder E and T different decoders $D_1, \dots D_T$

Until converged or done:

1. Select the next task t
2. Randomly sample an instance $\left(x_i^t, y_i^t\right)$ from C_t
3. Train the encoder E and decoder C_t on $\left(x_i^t, y_i^t\right)$

PSA:

Multi-task learning did not begin in 2008

Two Well-Known Instances of Multi-Task Learning in NLP

Collobert and Weston (2008, ICML)

BERT [Devlin et al., 2019 NAACL)

A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning

Ronan Collobert COLLOBER@NEC-LABS.COM
Jason Weston JASONW@NEC-LABS.COM
NEC Labs America, 4 Independence Way, Princeton, NJ 08540 USA

Abstract

We describe a single convolutional neural network architecture that, given a sentence, outputs a host of language processing predictions: part-of-speech tags, chunks, named entity tags, semantic roles, semantically similar words and the likelihood that the sentence makes sense (grammatically and semantically) using a language model. The entire network is trained *jointly* on all these tasks using weight-sharing, an instance of *multitask learning*. All the tasks use labeled data except the language model which is learnt from unlabeled text and represents a novel form of *semi-supervised learning* for the shared tasks. We show how both *multitask learning* and *semi-supervised learning* improve the generalization of the shared tasks, resulting in state-of-the-art performance.

Currently, most research analyzes those tasks *separately*. Many systems possess few characteristics that would help develop a unified architecture which would presumably be necessary for deeper semantic tasks. In particular, many systems possess three failings in this regard: (i) they are *shallow* in the sense that the classifier is often linear, (ii) for good performance with a linear classifier they must incorporate many hand-engineered features specific for the task; and (iii) they cascade features learnt separately from other tasks, thus propagating errors.

In this work we attempt to define a unified architecture for Natural Language Processing that *learns features* that are relevant to the tasks at hand given very limited prior knowledge. This is achieved by training a *deep neural network*, building upon work by (Bengio & Ducharme, 2001) and (Collobert & Weston, 2007). We define a rather general convolutional network architecture and describe its application to many well known NLP tasks including part-of-speech tagging, chunking,

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are

Two Well-Known Instances of Multi-Task Learning in NLP

Collobert and Weston (2008, ICML)

A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning

Ronan Collobert COLLOBER@NEC-LABS.COM
Jason Weston JASONW@NEC-LABS.COM
NEC Labs America, 4 Independence Way, Princeton, NJ 08540 USA

Abstract

We describe a single convolutional neural network architecture that, given a sentence, outputs a host of language processing predictions: part-of-speech tags, chunks, named entity tags, semantic roles, semantically similar words and the likelihood that the sentence makes sense (grammatically and semantically) using a language model. The entire network is trained *jointly* on all these tasks using weight-sharing, an instance of *multitask learning*. All the tasks use labeled data except the language model which is learnt from unlabeled text and represents a novel form of *semi-supervised learning* for the shared tasks. We show how both *multitask learning* and *semi-supervised learning* improve the generalization of the shared tasks, resulting in state-of-the-art performance.

Currently, most research analyzes those tasks *separately*. Many systems possess few characteristics that would help develop a unified architecture which would presumably be necessary for deeper semantic tasks. In particular, many systems possess three failings in this regard: (i) they are *shallow* in the sense that the classifier is often linear, (ii) for good performance with a linear classifier they must incorporate many hand-engineered features specific for the task; and (iii) they cascade features learnt separately from other tasks, thus propagating errors.

In this work we attempt to define a unified architecture for Natural Language Processing that *learns features* that are relevant to the tasks at hand given very limited prior knowledge. This is achieved by training a *deep neural network*, building upon work by (Bengio & Ducharme, 2001) and (Collobert & Weston, 2007). We define a rather general convolutional network architecture and describe its application to many well known NLP tasks including part-of-speech tagging, chunking,

BERT [Devlin et al., 2019 NAACL)

(already saw this)

Collobert and Weston (2008, ICML)

Core task: Semantic Role Labeling

Present a unified architecture for doing five other, related NLP tasks

- Part-of-Speech Tagging
- Chunking
- Named Entity Recognition
- Language Modeling
- Prediction of Semantic Relatedness

Collobert and Weston (2008, ICML)

Core task: Semantic Role Labeling

Present a unified architecture for doing five other, related NLP tasks

- Part-of-Speech Tagging
- Chunking
- Named Entity Recognition
- Language Modeling
- Prediction of Semantic Relatedness

Remember from Deck 5

Semantic Role Labeling (SRL)

- For each predicate (e.g., verb)
  1. find its arguments (e.g., NPs)
  2. determine their **semantic roles**

John drove Mary from Austin to Dallas in his Toyota Prius.

The hammer broke the window.

- agent: Actor of an action
- patient: Entity affected by the action
- source: Origin of the affected entity
- destination: Destination of the affected entity
- instrument: Tool used in performing action.
- beneficiary: Entity for whom action is performed

Remember from Deck 5

Uses of Semantic Roles

- Find the answer to a user's question
  - "Who" questions usually want Agents
  - "What" question usually want Patients
  - "How" and "with what" questions usually want Instruments
  - "Where" questions frequently want Sources/Destinations.
  - "For whom" questions usually want Beneficiaries
  - "To whom" questions usually want Destinations
- Generate text
  - Many languages have specific syntactic constructions that must or should be used for specific semantic roles.
- Word sense disambiguation, using selectional restrictions
  - The **bat** ate the **bug**. (what kind of bat? what kind of bug?)
    - Agents (particularly of "eat") should be animate – animal bat, not baseball bat
    - Patients of "eat" should be edible – animal bug, not software bug
  - John **fired** the secretary.

    John **fired** the rifle.

    Patients of $fire_1$ are different than patients of $fire_2$

Collobert and Weston (2008, ICML)

Core task: Semantic Role Labeling

Present a unified architecture for doing five other, related NLP tasks

- Part-of-Speech Tagging
- Chunking
- Named Entity Recognition
- Language Modeling
- Prediction of Semantic Relatedness

Collobert and Weston (2008, ICML)

Core task: Semantic Role Labeling

Present a unified architecture for doing five other, related NLP tasks

- Part-of-Speech Tagging
- Chunking
- Named Entity Recognition
- Language Modeling
- Prediction of Semantic Relatedness

Collobert and Weston (2008, ICML)

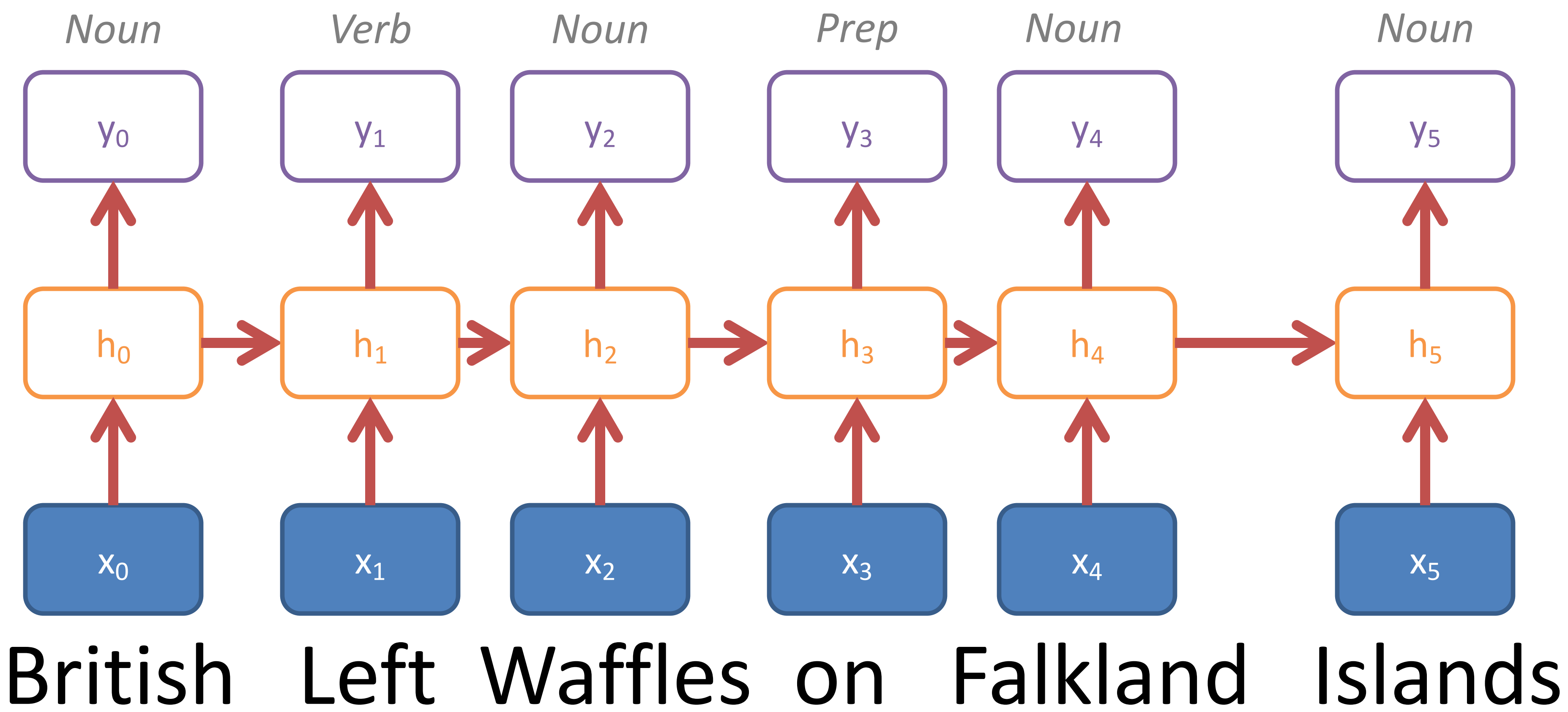
Core task: Semantic Role Labeling

Present a unified architecture for doing five other, related NLP tasks

- Part-of-Speech Tagging
- Chunking
- Named Entity Recognition
- Language Modeling
- Prediction of Semantic Relatedness

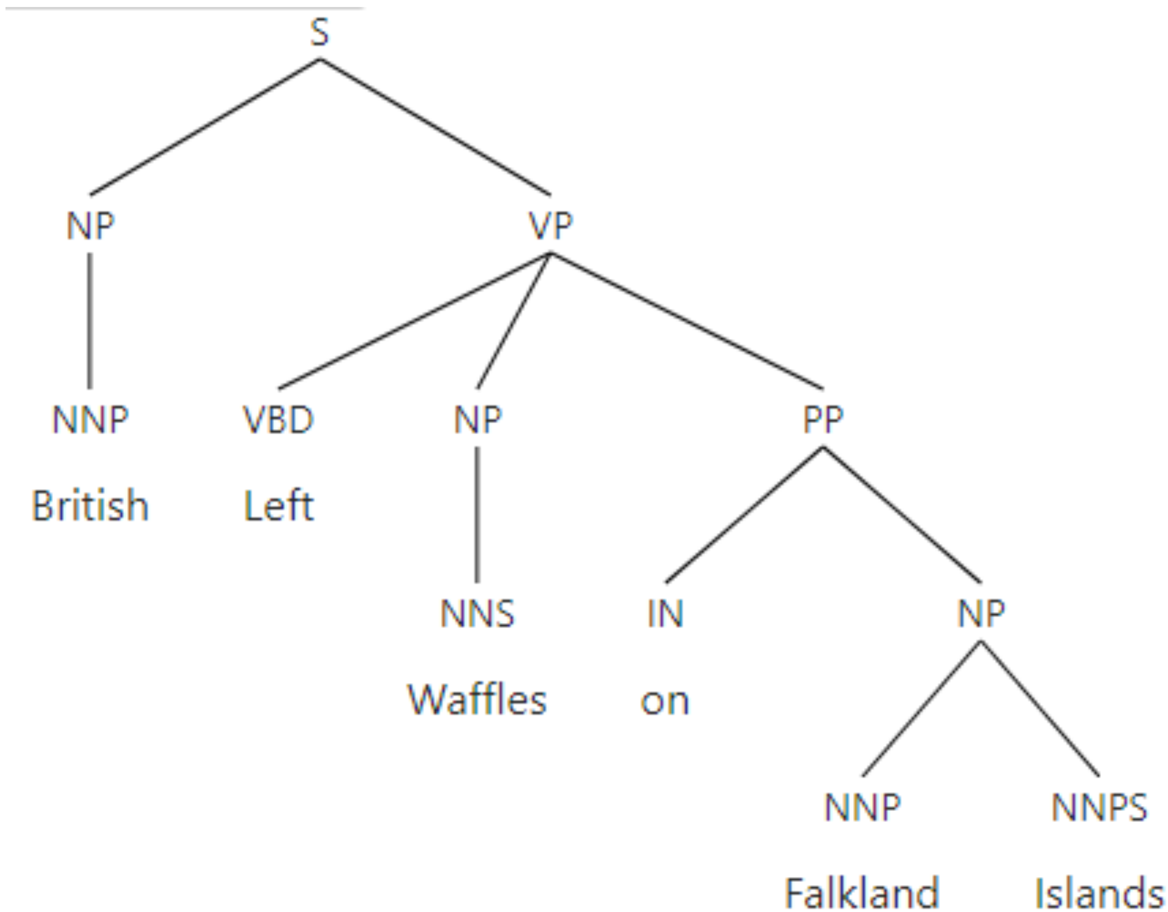
Part of Speech Tagging

(sequence is probably not right!)



Part-of-speech tagging: assign a part-of-speech tag to every word in a sentence

Syntactic Parsing (One Option)



(parse is probably not right!)

(parse from the Berkeley parser: https://parser.kitaev.io/)

Part-of-speech tagging: assign a part-of-speech tag to every word in a sentence

Syntactic parsing: produce an analysis of a sentence according to some grammatical rules

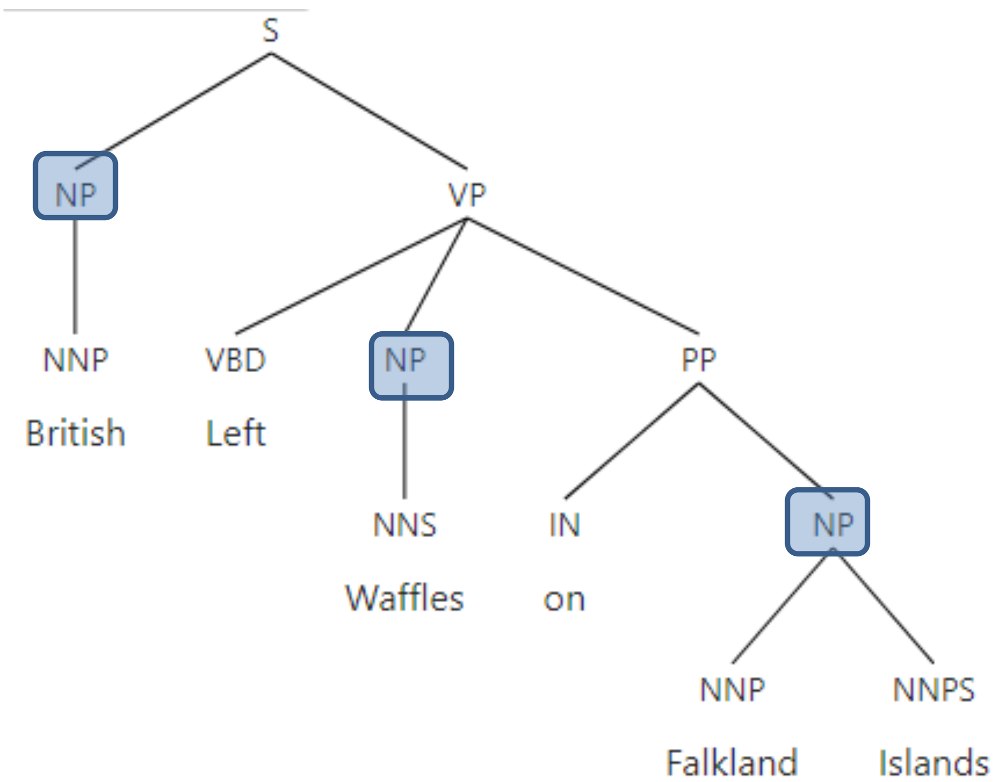
Part-of-speech tagging: assign a part-of-speech tag to every word in a sentence

Chunking: A Shallow Syntactic Parsing

Syntactic parsing: produce an analysis of a sentence according to some grammatical rules

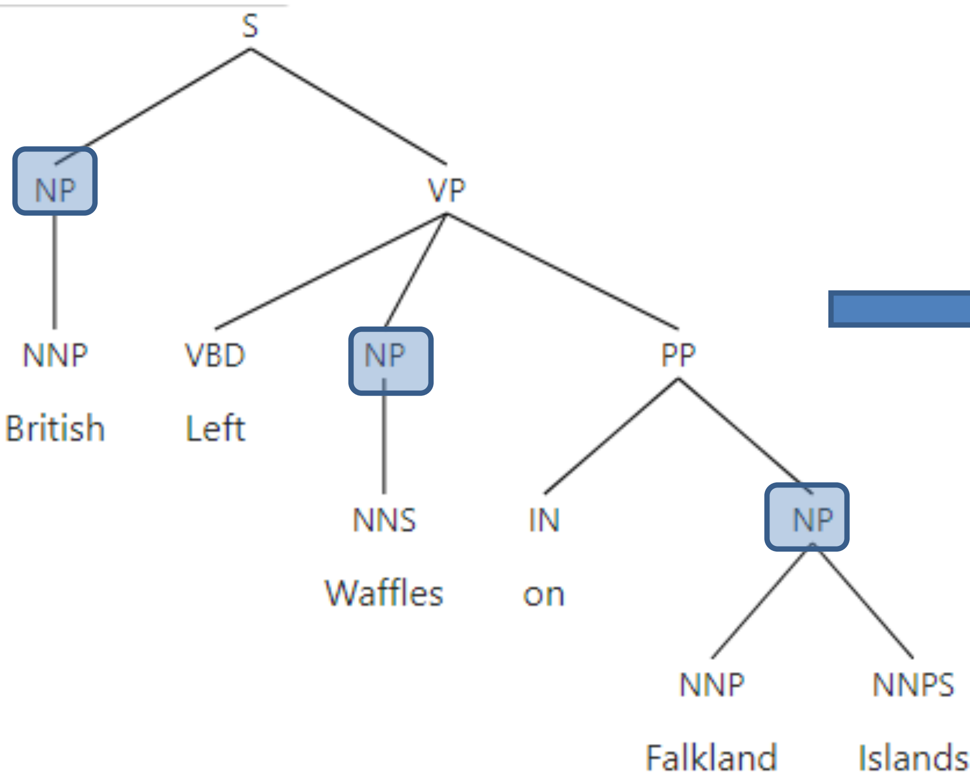
Chunking: A Shallow Syntactic Parse

- (Variant 1) For every token, predict whether it's in a noun-phrase (NP) or not



Chunking: A Shallow Syntactic Parse

- (Variant 1) For every token, predict whether it's in a noun-phrase (NP) or not



✓ X ✓ X ✓ ✓

British Left Waffles on Falkland Islands

Treat this as a sequence prediction problem

Chunking: A Shallow Syntactic Parse

- (Variant 1) For every token, predict whether it's in a noun-phrase (NP) or not
- (Variant 2) For every token, predict the type of *grammatical phrase* it should be part of

Collobert and Weston (2008, ICML)

Core task: Semantic Role Labeling

Present a unified architecture for doing five other, related NLP tasks

- Part-of-Speech Tagging
- Chunking
- Named Entity Recognition
- Language Modeling
- Prediction of Semantic Relatedness

Collobert & Weston Language Modeling

Our approach so far: predict a word given some previous words

$$p(w_1, \ldots, w_N) = \prod_i p(w_i | w_{<i})$$

V-class classification

Their approach: predict* whether w_i is the correct word, based on context

$$p(y = 1 \,|c = (w_{i-M}, w_{i-1}, \ldots, w_{i+1}, w_{i+M}), w_i)$$

Binary classification

*They actually use a *ranking* loss, but it's close enough to what's described here

Collobert & Weston Language Modeling (Example)

Sentence: British Left Waffles on Falkland Islands

Word: "Waffles"

Predict*:

$$p(y = 1\,|c = (\text{Left}, \text{on}), w_i = \text{Waffles})$$
$$p(y = 0\,|c = (\text{Left}, \text{on}), w_i = \text{Hats})$$

Any word but "Waffles"

*They actually use a *ranking* loss, but it's close enough to what's described here

Collobert and Weston (2008, ICML)

Core task: Semantic Role Labeling

Present a unified architecture for doing five other, related NLP tasks

- Part-of-Speech Tagging
- Chunking
- Named Entity Recognition
- Language Modeling
- Prediction of Semantic Relatedness

Prediction of Semantic Relatedness

Are two words "semantically related?"

- Synonym: different word, same meaning
- Is-a relationships
- Part/whole relationships
- (and others)

Prediction of Semantic Relatedness

Are two words “semantically related?”

- Synonym: different word, same meaning
- Is-a relationships
  - X hypernym Y: X is a (sub)type of Y
    - car hypernym “motor vehicle”
  - X hyponym Y: X is a (super)type of Y
    - car hyponym sedan
- Part/whole relationships
- (and others)

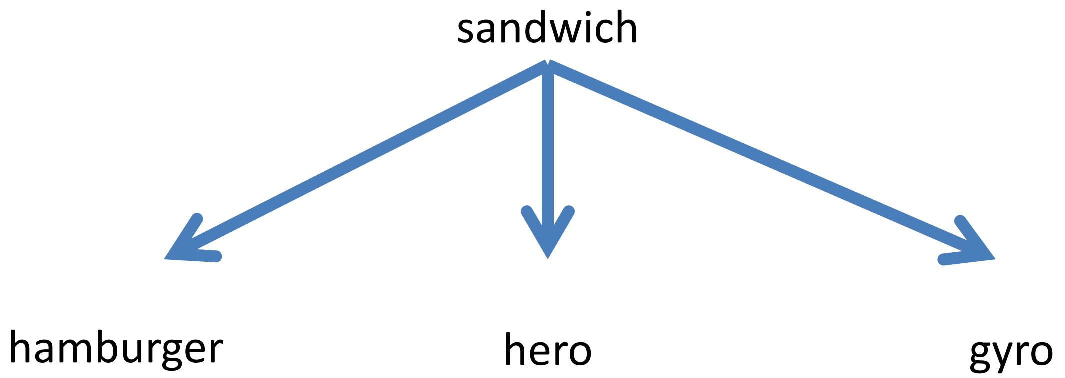
Prediction of Semantic Relatedness

Are two words “semantically related?”

- Synonym: different word, same meaning
- Is-a relationships
  - X hypernym Y: X is a (sub)type of Y
    - car hypernym “motor vehicle”
  - X hyponym Y: X is a (super)type of Y
    - car hyponym sedan
- Part/whole relationships
  - X meronym Y: X is a part of Y
    - window meronym car
  - X holonym Y: X is the whole, with Y as a part
    - car holonym window
- (and others)

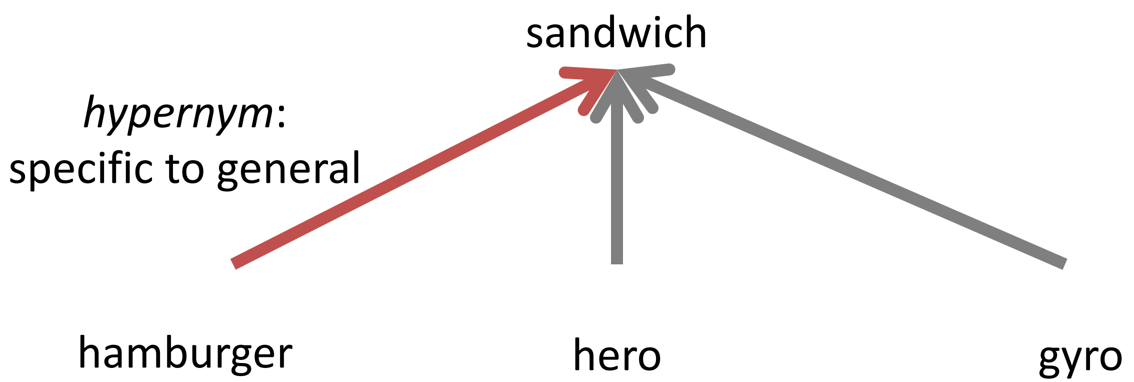
WordNet

Knowledge graph containing *concept* relations



WordNet

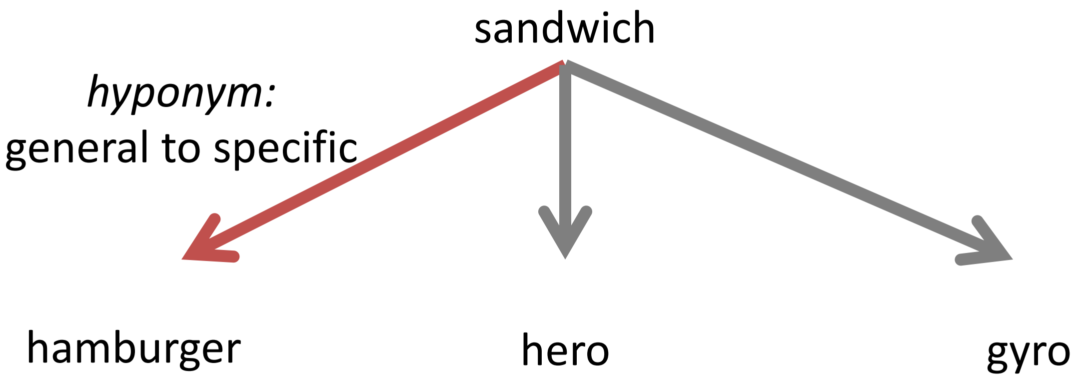
Knowledge graph containing *concept* relations



a hamburger is-a sandwich

WordNet

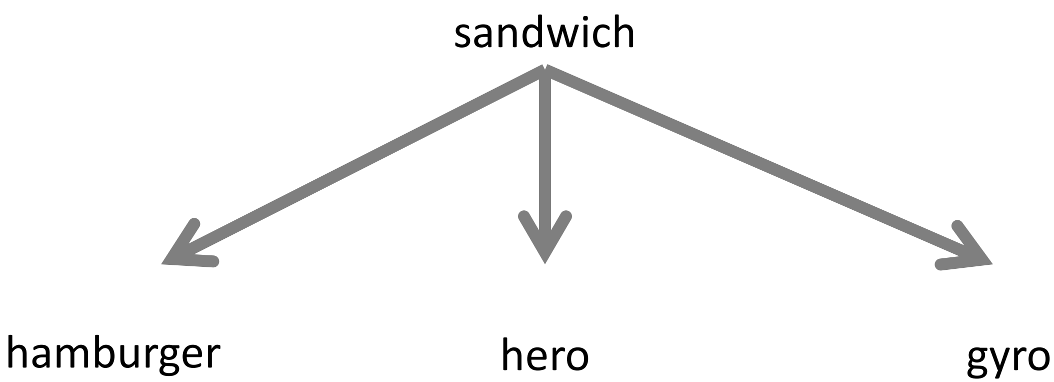
Knowledge graph containing *concept* relations



a hamburger is-a sandwich

WordNet

Knowledge graph containing *concept* relations



Other relationships too:

- meronymy, holonymy
  (part of whole, whole of part)
- troponymy
  (describing *manner* of an event)
- entailment
  (what else *must* happen in an event)

WordNet Knows About Hamburgers

specific

hamburger

sandwich

snack food

dish

nutriment

food

substance

matter

physical entity

entity

general

Browsing WordNet

http://wordnetweb.princeton.edu/perl/webwn

Word to search for: car [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **car**, auto, automobile, machine, motorcar (a motor vehicle with four wheels; usually propelled by an internal combustion engine) *"he needs a car to get to work"*
- S: (n) **car**, railcar, railway car, railroad car (a wheeled vehicle adapted to the rails of railroad) *"three cars had jumped the rails"*
- S: (n) **car**, gondola (the compartment that is suspended from an airship and that carries personnel and the cargo and the power plant)
- S: (n) **car**, elevator car (where passengers ride up and down) *"the car was on the top floor"*
- S: (n) cable car, **car** (a conveyance for passengers or freight on a cable railway) *"they took a cable car to the top of the mountain"*

Each of these is a **synset** (**syn**onym **set**)

Browsing WordNet

http://wordnetweb.princeton.edu/perl/webwn

Word to search for: car [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- **S: (n) car, auto, automobile, machine, motorcar (a motor vehicle with four wheels; usually propelled by an internal combustion engine) *"he needs a car to get to work"***
  - *direct hyponym* / *full hyponym*
  - *part meronym*
  - *domain term category*
  - *direct hypernym* / *inherited hypernym* / *sister term*
  - *derivationally related form*
- S: (n) **car**, railcar, railway car, railroad car (a wheeled vehicle adapted to the rails of railroad) *"three cars had jumped the rails"*
- S: (n) **car**, gondola (the compartment that is suspended from an airship and that carries personnel and the cargo and the power plant)
- S: (n) **car**, elevator car (where passengers ride up and down) *"the car was on the top floor"*
- S: (n) cable car, **car** (a conveyance for passengers or freight on a cable railway) *"they took a cable car to the top of the mountain"*

Get the relationships for each synset

Results (Error Rate: Lower is Better)

	Word embedding size		
	wsz=15	wsz=50	wsz=100
SRL	16.54	17.33	18.40

Results (Error Rate: Lower is Better)

	Word embedding size		
	wsz=15	wsz=50	wsz=100
SRL	16.54	17.33	18.40
SRL + POS	15.99	16.57	16.53
SRL + Chunking	16.42	16.39	16.48
SRL + NER	16.67	17.29	17.21
SRL + Synonyms	15.46	15.17	15.17
SRL + Language model	14.42	14.30	14.46

Results (Error Rate: Lower is Better)

	Word embedding size		
	wsz=15	wsz=50	wsz=100
SRL	16.54	17.33	18.40
SRL + POS	15.99	16.57	16.53
SRL + Chunking	16.42	16.39	16.48
SRL + NER	16.67	17.29	17.21
SRL + Synonyms	15.46	15.17	15.17
SRL + Language model	14.42	14.30	14.46
SRL + POS + Chunking	16.46	15.95	16.41
SRL + POS + NER	16.45	16.89	16.29
SRL + POS + Chunking + NER	16.33	16.36	16.27
SRL + POS + Chunking + NER + Synonyms	15.71	14.76	15.48
SRL + POS + Chunking + NER + Language model	14.63	14.44	14.50

Outline

Multi-task Learning

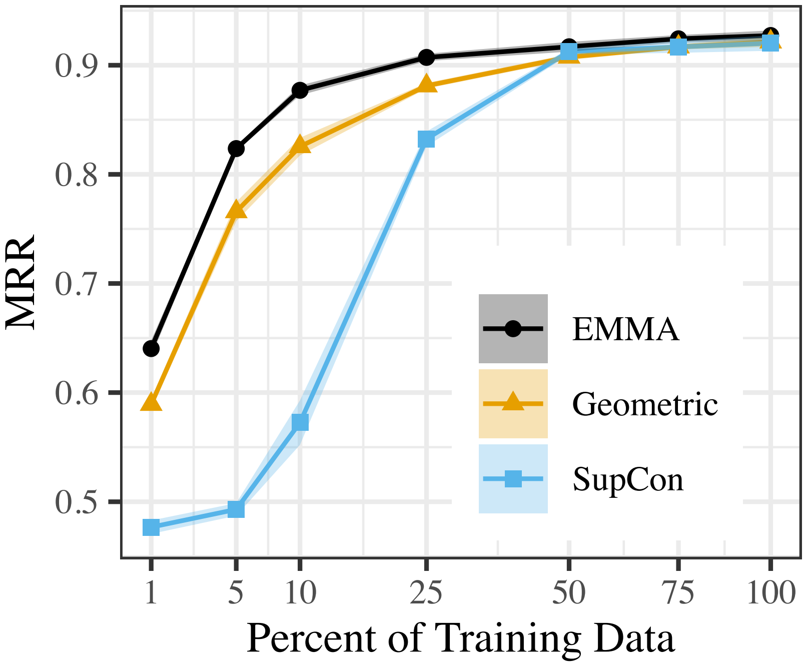
Prompting

Terminology: Few-shot (k-shot)

- Given only *k* “training” examples, can the model generalize to test time. A few ways this can be realized:
  - *k*-shot fine-tuning
  - *k*-shot in-context learning (prompting)
  - *k*-shot prompt tuning
  - *…*

k-shot fine-tuning

- Effectively, your *entire* training set *only* has *k* labeled examples
- Use this training set to update model parameters
- Evaluate as normal



Darvish et al., 2023

GPT-3

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam

Girish Sastry Amanda Askell Sandhini Agarwal Ariel Herbert-Voss

Gretchen Krueger Tom Henighan Rewon Child Aditya Ramesh

Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

Abstract

We demonstrate that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even becoming competitive with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks. We also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora.

1 Introduction

NLP has shifted from learning task-specific representations and designing task-specific architectures to using task-agnostic pre-training and task-agnostic architectures. This shift has led to substantial progress on many challenging NLP tasks such as reading comprehension, question answering, textual entailment, among others. Even though the architecture and initial representations are now task-agnostic, a final task-specific step remains: fine-tuning on a large dataset of examples to adapt a task agnostic model to perform a desired task.

Recent work [RWC+19] suggested this final step may not be necessary. [RWC+19] demonstrated that a single pretrained language model can be zero-shot transferred to perform standard NLP tasks

*Equal contribution
†Johns Hopkins University, OpenAI

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

GPT-3

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam
Girish Sastry Amanda Askell Sandhini Agarwal Ariel Herbert-Voss
Gretchen Krueger Tom Henighan Rewon Child Aditya Ramesh
Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

Abstract

We demonstrate that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even becoming competitive with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks. We also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora.

1 Introduction

NLP has shifted from learning task-specific representations and designing task-specific architectures to using task-agnostic pre-training and task-agnostic architectures. This shift has led to substantial progress on many challenging NLP tasks such as reading comprehension, question answering, textual entailment, among others. Even though the architecture and initial representations are now task-agnostic, a final task-specific step remains: fine-tuning on a large dataset of examples to adapt a task agnostic model to perform a desired task.

Recent work [RWC+19] suggested this final step may not be necessary. [RWC+19] demonstrated that a single pretrained language model can be zero-shot transferred to perform standard NLP tasks

*Equal contribution
†Johns Hopkins University, OpenAI

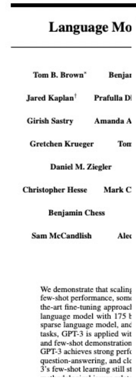
34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2.1 Model and Architectures

We use the same model and architecture as GPT-2 [RWC+19], including the modified initialization, pre-normalization, and reversible tokenization described therein, with the exception that we use alternating dense and locally banded sparse attention patterns in the layers of the transformer, similar to the Sparse Transformer [CGRS19]. To study the dependence of ML performance on model size, we train 8 different sizes of model, from 125 million parameters to 175 billion parameters, with the last being the model we call GPT-3. This range of model sizes allows us to test the scaling laws introduced in [KMH+20].

More details on the sizes and architectures of our models can be found in the appendix. We partition each model across GPUs along both the depth and width dimension in order to minimize data-transfer between nodes.

GPT-3


Language Mo

Tom B. Brown* Benja
Jared Kaplan† Prafulla Di
Girish Sastry Amanda A
Gretchen Krueger Tom
Daniel M. Ziegler
Christopher Hesse Mark C
Benjamin Chess
Sam McCandlish Alec

2.1 Model and Architectures

We use the same model and architecture as GPT-2 [RWC+19], including the modified initialization, pre-normalization, and reversible tokenization described therein, with the exception that we use alternating dense and locally banded sparse attention patterns in the layers of the transformer, similar to the Sparse Transformer [CGRS19]. To study the dependence of ML performance on model size, we train 8 different sizes of model, from 125 million parameters to 175 billion parameters, with the last being the model we call GPT-3. This range of model sizes allows us to test the scaling laws introduced in [KMH+20].

More details on the sizes and architectures of our models can be found in the appendix. We partition each model across GPUs along both the depth and width dimension in order to minimize data-transfer between nodes.

B Details of Model Training

To train all versions of GPT-3, we use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$, we clip the global norm of the gradient at 1.0, and we use cosine decay for learning rate down to 10% of its value, over 260 billion tokens (after 260 billion tokens, training continues at 10% of the original learning rate). There is a linear LR warmup over the first 375 million tokens. We also gradually increase the batch size linearly from a small value (32k tokens) to the full value over the first 4-12 billion tokens of training, depending on the model size. Data are sampled without replacement during training (until an epoch boundary is reached) to minimize overfitting. All models use weight decay of 0.1 to provide a small amount of regularization [LH17].

During training we always train on sequences of the full $n_{\text{ctx}} = 2048$ token context window, packing multiple documents into a single sequence when documents are shorter than 2048, in order to increase computational efficiency. Sequences with multiple documents are not masked in any special way but instead documents within a sequence are delimited with a special end of text token, giving the language model the information necessary to infer that context separated by the end of text token is unrelated. This allows for efficient training without need for any special sequence-specific masking.

n_{params} is the total number of trainable parameters, n_{layers} is the total number of layers, d_{model} is the number of units in each bottleneck layer (we always have the feedforward layer four times the size of the bottleneck layer, $d_{\text{ff}} = 4 * d_{\text{model}}$), and d_{head} is the dimension of each attention head. All models use a context window of $n_{\text{ctx}} = 2048$ tokens.

GPT-3

Language Mo

Tom B. Brown* Benjai
Jared Kaplan† Prafulla Di
Girish Sastry Amanda A
Gretchen Krueger Tom
Daniel M. Ziegler
Christopher Hesse Mark C
Benjamin Chess
Sam McCandlish Alec

We demonstrate that scaling
few-shot performance, some
the-art fine-tuning approacl
language model with 175 b
sparse language model, and
tasks, GPT-3 is applied wit
and few-shot demonstration
GPT-3 achieves strong perfo
question-answering, and clo
3's few-shot learning still st
methodological issues relate

1 Introduction

NLP has shifted from learning task-specific representations and designing task-specific architectures to using task-agnostic pre-training and task-agnostic architectures. This shift has led to substantial progress on many challenging NLP tasks such as reading comprehension, question answering, textual entailment, among others. Even though the architecture and initial representations are now task-agnostic, a final task-specific step remains: fine-tuning on a large dataset of examples to adapt a task agnostic model to perform a desired task.

Recent work [RWC+19] suggested this final step may not be necessary. [RWC+19] demonstrated that a single pretrained language model can be zero-shot transferred to perform standard NLP tasks

*Equal contribution
†Johns Hopkins University, OpenAI

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2.1 Model and Archite

We use the same model an
pre-normalization, and re
alternating dense and local
to the Sparse Transformer
we train 8 different sizes o
last being the model we c
introduced in [KMH+20].

More details on the sizes a
each model across GPUs a
between nodes.

B Details of Model Training

To train all versions of GPT-3, we use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$, we clip the global norm of the gradient at 1.0, and we use cosine decay for learning rate down to 10% of its value, over 260 billion tokens (after 260 billion tokens, training continues at 10% of the original learning rate). There is a linear LR warmup over the first 375 million tokens. We also gradually increase the batch size linearly from a small value (32k tokens) to the full value over the first 4-12 billion tokens of training, depending on the model size. Data are sampled without replacement during training (until an epoch boundary is reached) to minimize overfitting. All models use weight decay of 0.1 to provide a small amount of regularization [LH17].

During training we always train on sequences of the full $n_{\text{ctx}} = 2048$ token context window, packing multiple documents into a single sequence when documents are shorter than 2048, in order to increase computational efficiency. Sequences with multiple documents are not masked in any special way but instead documents within a sequence are delimited with a special end of text token, giving the language model the information necessary to infer that context separated by the end of text token is unrelated. This allows for efficient training without need for any special sequence-specific masking.

n_{params} is the total number of trainable parameters, n_{layers} is the total number of layers, d_{model} is the number of units in each bottleneck layer (we always have the feedforward layer four times the size of the bottleneck layer, $d_{\text{ff}} = 4 * d_{\text{model}}$), and d_{head} is the dimension of each attention head. All models use a context window of $n_{\text{ctx}} = 2048$ tokens.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table C.1: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

GPT-3

Language Mo

Tom B. Brown* Benja
Jared Kaplan† Prafulla D
Girish Sastry Amanda A
Gretchen Krueger Tom
Daniel M. Ziegler
Christopher Hesse Mark C
Benjamin Chess
Sam McCandlish Alec

2.1 Model and Architec

We use the same model an
pre-norma
alternating
to the Spa
we train 8
last being
introduced

More detai
each mode
between n

1 Introduction

B Details of Model Training

To train all versions of GPT-3, we use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$, we clip the global norm of the gradient at 1.0, and we use cosine decay for learning rate down to 10% of its value,

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

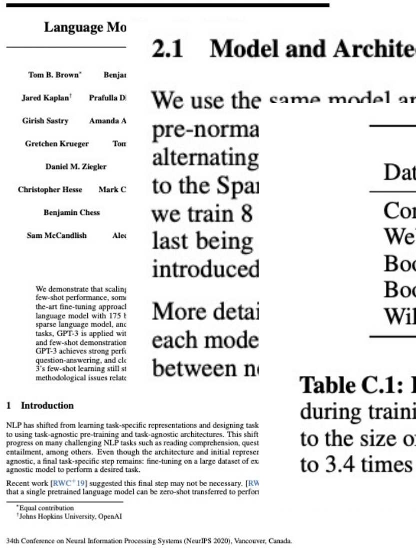
Table C.1: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

"Since our training dataset is sourced from the internet, it is possible that our model was trained on some of our benchmark test sets…On the one hand, the dataset and model size are about two orders of magnitude larger than those used for GPT-2, and include a large amount of Common Crawl, increasing the potential for contamination and memorization. On the other hand, precisely due to the large amount of data, even GPT-3 175B does not overfit its training set by a significant amount, measured relative to a held-out validation set with which it was deduplicated (Figure C.1). Thus, we expect that contamination is likely to be frequent, but that its effects may not be as large as feared…" (Appendix C)

GPT-3

"Since our training dataset is sourced from the internet, it is possible that our model was trained on some of our benchmark test sets…On the one hand, the dataset and model size are about two orders of magnitude larger than those used for GPT-2, and include a large amount of Common Crawl, increasing the potential for contamination and memorization. On the other hand, precisely due to the large amount of data, even GPT-3 175B does not overfit its training set by a significant amount, measured relative to a held-out validation set with which it was deduplicated (Figure C.1). Thus, we expect that contamination is likely to be frequent, but that its effects may not be as large as feared…" (Appendix C)


Language Mo

1 Introduction

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2.1 Model and Archite

We use the same model an
pre-norma
alternating
to the Spa
we train 8
last being
introduced

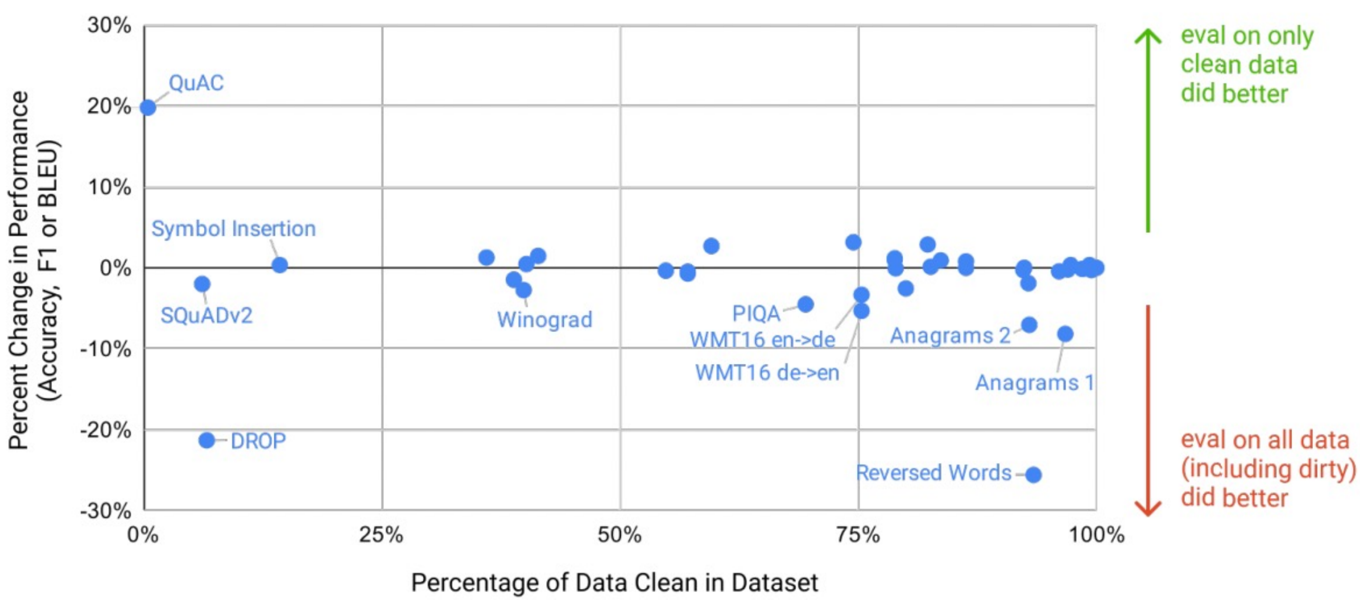
More detai
each mode
between n

B Details of Model Training

To train all versions of GPT-3, we use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$, we clip the global norm of the gradient at 1.0, and we use cosine decay for learning rate down to 10% of its value,

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table C.1: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

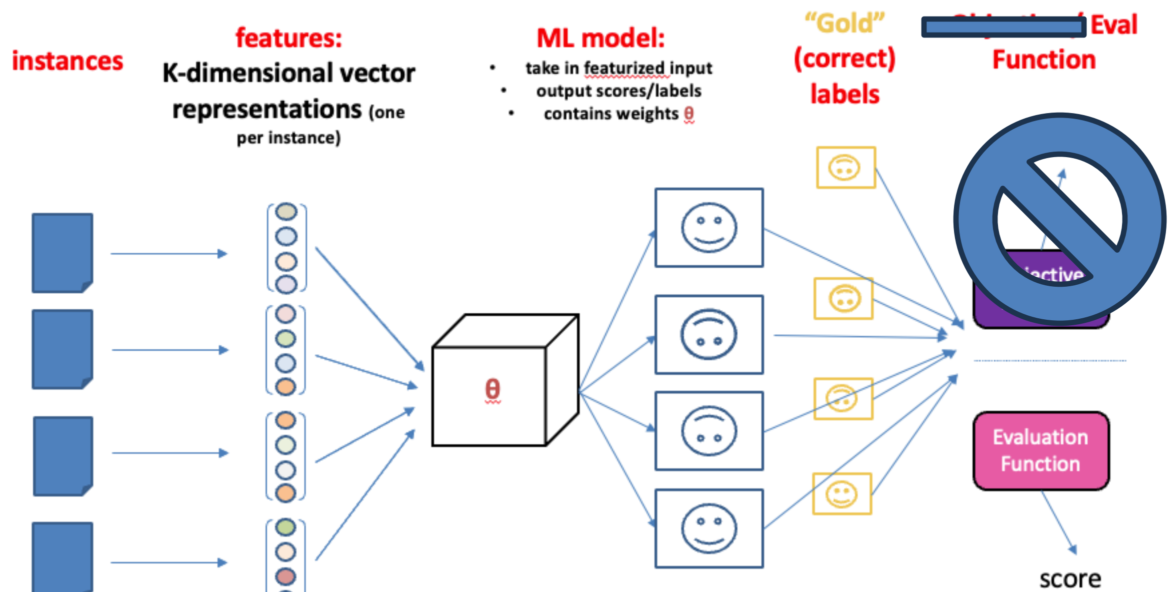


Terminology: Few-shot (k-shot)

- Given only *k* “training” examples, can the model generalize to test time. A few ways this can be realized:
  - *k*-shot fine-tuning
  - *k*-shot in-context learning (prompting)
  - *k*-shot prompt tuning
  - *…*

k-shot in-context learning (prompting)

- Effectively, your *entire* training set *only* has *k* labeled examples
- However, rather than using this training set to update model parameters…
- Prepend those *k* labeled examples to each test instance.
- Evaluate as normal



Inference only: No *further* learning / tuning of model parameters

k-shot in-context learning (prompting)

instances

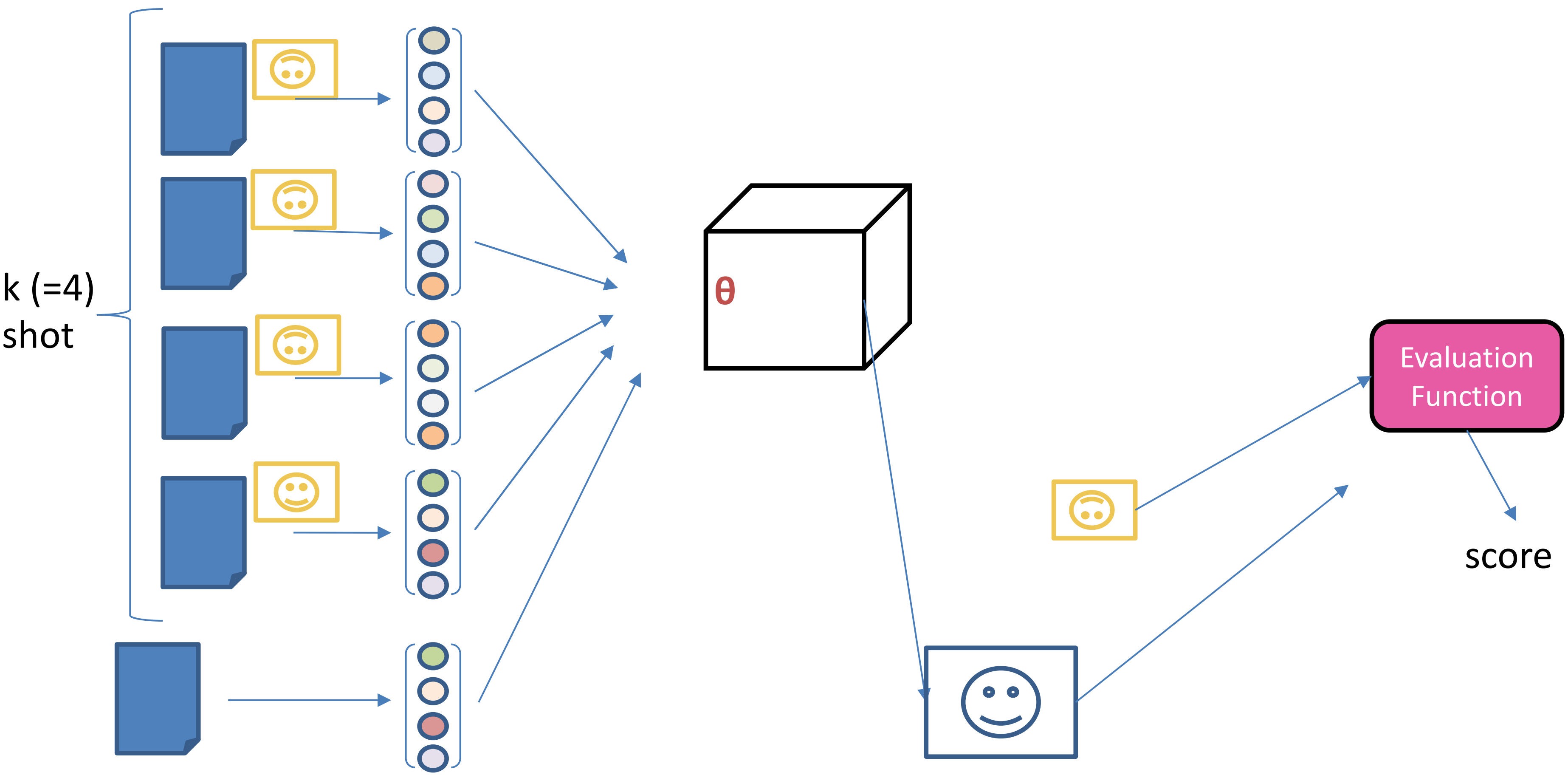
features:
K-dimensional vector representations (one per instance)

ML model:

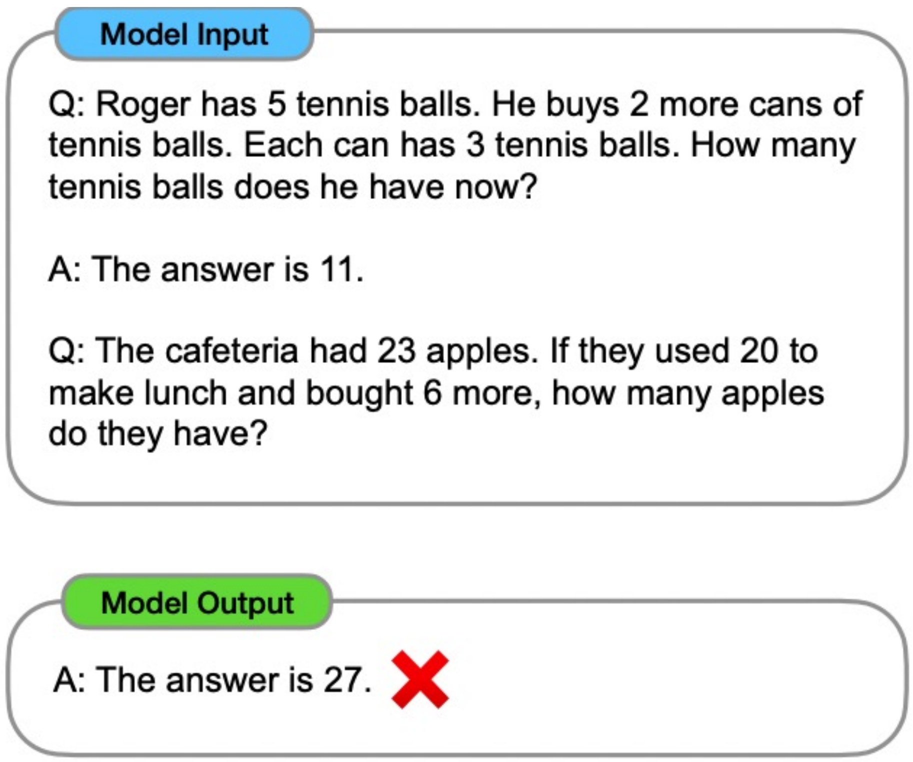
- take in featurized input
- output scores/labels
- contains weights θ

"Gold" (correct) labels

Eval Function



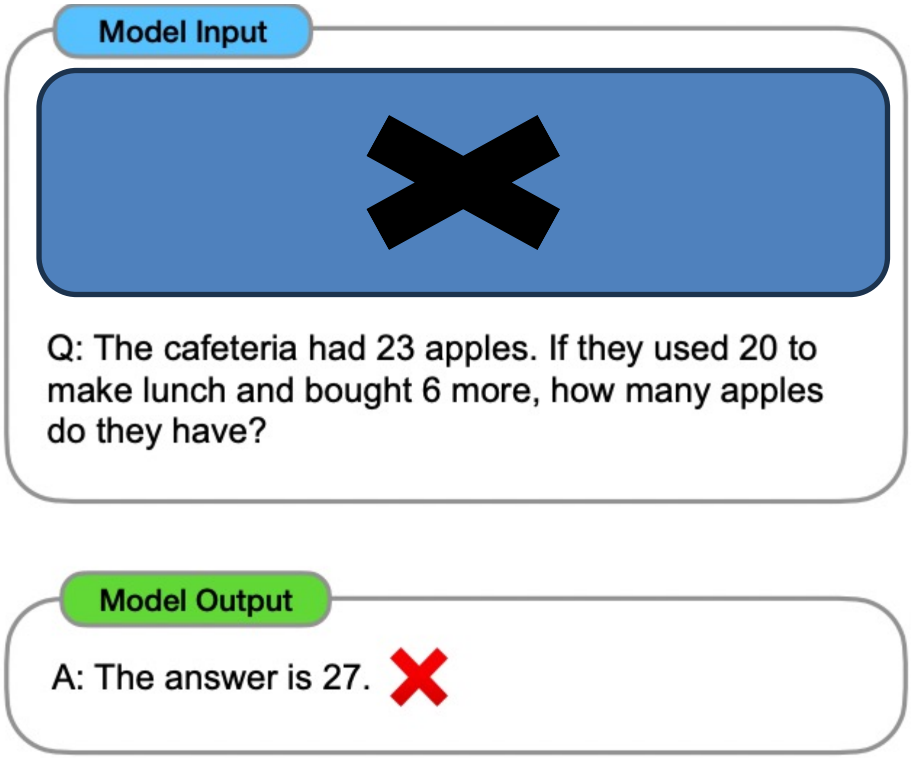
1-shot prompting



https://openreview.net/pdf?id=_VjQlMeSB_J, Fig. 1

0-shot prompting

- Describe the task (maybe).
- Or pose as a question.
- *No* examples as part of the prompt.



Does it work?

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	**89.0**	**91.0**	**96.9**	**93.9**	**94.8**	**92.5**
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	**76.1**	**93.8**	**62.3**	**88.2**	**92.5**	**93.3**
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

Table 3.5: Performance of GPT-3 on SuperGLUE compared to fine-tuned baselines and SOTA. All results are reported on the test set. GPT-3 few-shot is given a total of 32 examples within the context of each task and performs no gradient updates.

Does it work?

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0 [b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	<u>37.5</u>	34.9	28.3	35.2	<u>35.2</u>	33.1
mBART [LGG+20]	-	-	<u>29.8</u>	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	<u>39.2</u>	29.7	<u>40.6</u>	21.0	<u>39.5</u>

Table 3.4: Few-shot GPT-3 outperforms previous unsupervised NMT work by 5 BLEU when translating into English reflecting its strength as an English LM. We report BLEU

Let’s say you have more than *k* examples, but want to do *k*-shot prompting…

Different strategies for choosing the examples:

- randomly (fixed)
- randomly per instance
- “nearest neighbor”
- “expert selection”

Chain-of-Thought Prompting

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei Xuezhi Wang Dale Schuurmans Maarten Bosma
Brian Ichter Fei Xia Ed H. Chi Quoc V. Le Denny Zhou

Google Research, Brain Team
{jasonwei,dennyzhou}@google.com

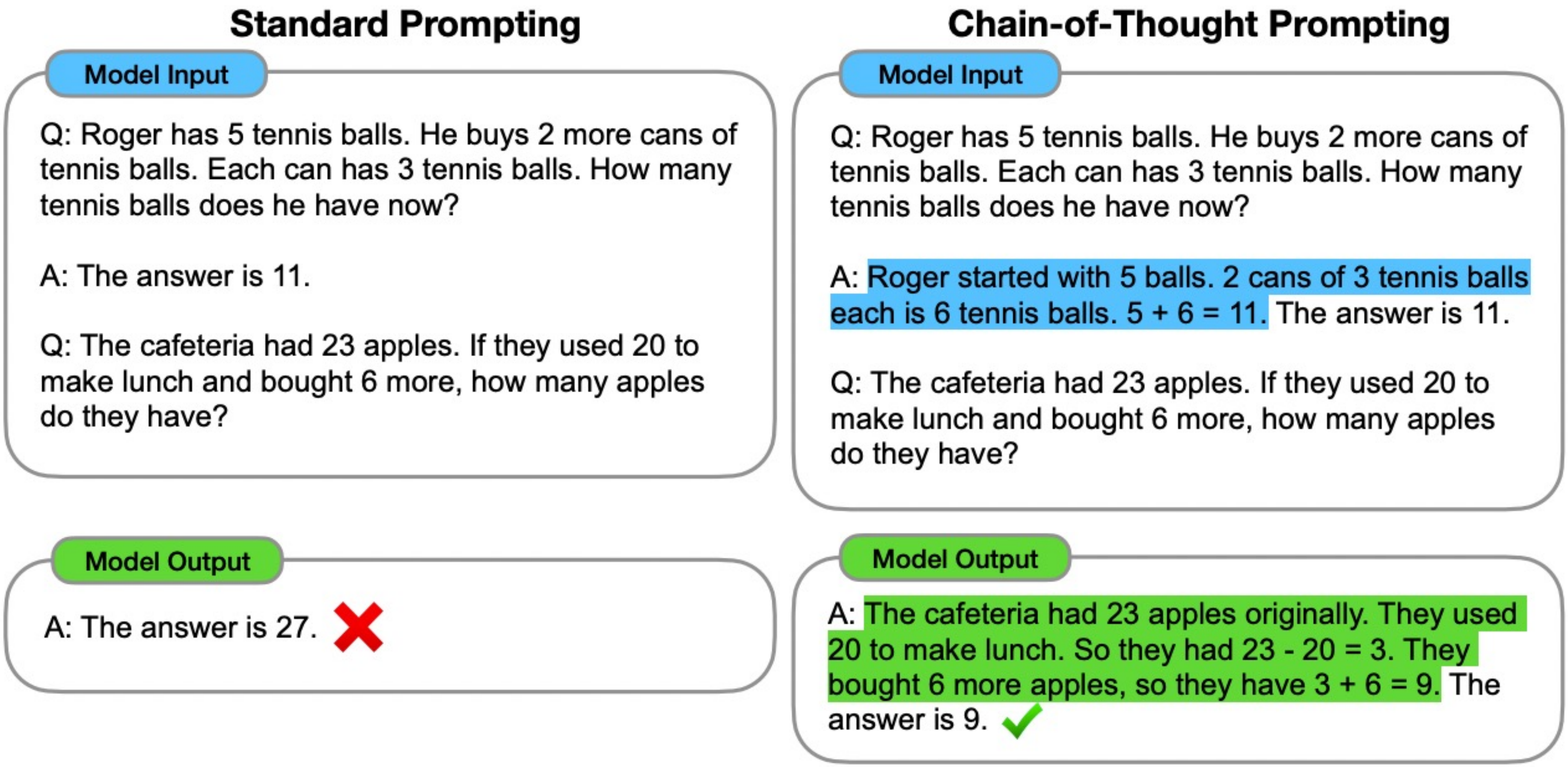
Abstract

We explore how generating a *chain of thought*—a series of intermediate reasoning steps—significantly improves the ability of large language models to perform complex reasoning. In particular, we show how such reasoning abilities emerge naturally in sufficiently large language models via a simple method called *chain-of-thought prompting*, where a few chain of thought demonstrations are provided as exemplars in prompting.

Experiments on three large language models show that chain-of-thought prompting improves performance on a range of arithmetic, commonsense, and symbolic reasoning tasks. The empirical gains can be striking. For instance, prompting a PaLM 540B with just eight chain-of-thought exemplars achieves state-of-the-art accuracy on the GSM8K benchmark of math word problems, surpassing even finetuned GPT-3 with a verifier.

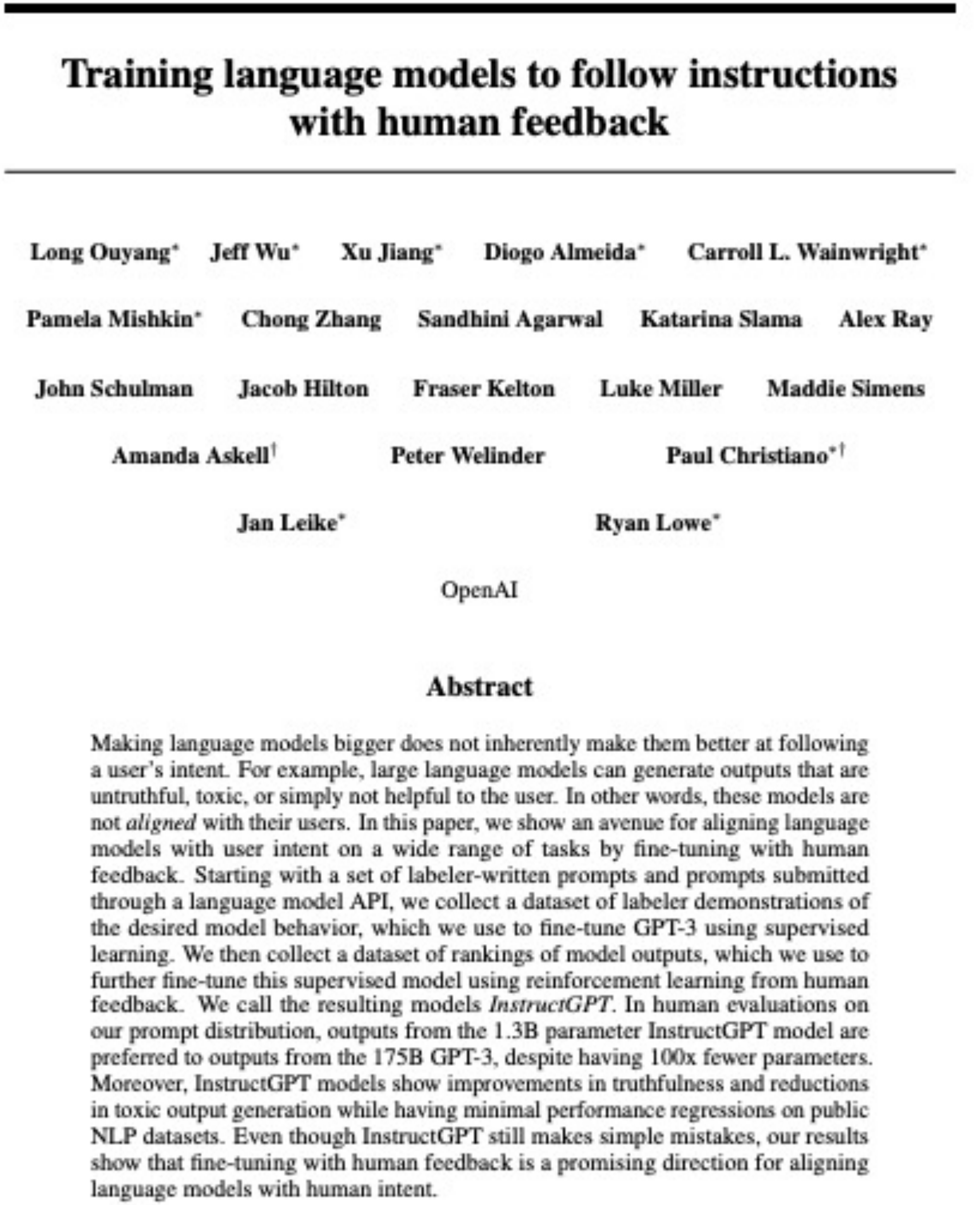
https://openreview.net/pdf?id=_VjQlMeSB_J

Chain-of-Thought



https://openreview.net/pdf?id=_VjQlMeSB_J, Fig. 1

InstructGPT


Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*
Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray
John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens
Amanda Askell† Peter Welinder Paul Christiano*†
Jan Leike* Ryan Lowe*
OpenAI

Abstract

Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not *aligned* with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through a language model API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback. We call the resulting models *InstructGPT*. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, our results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html

InstructGPT

Step 1

Collect demonstration data, and train a supervised policy.

Step 2

Collect comparison data, and train a reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

InstructGPT

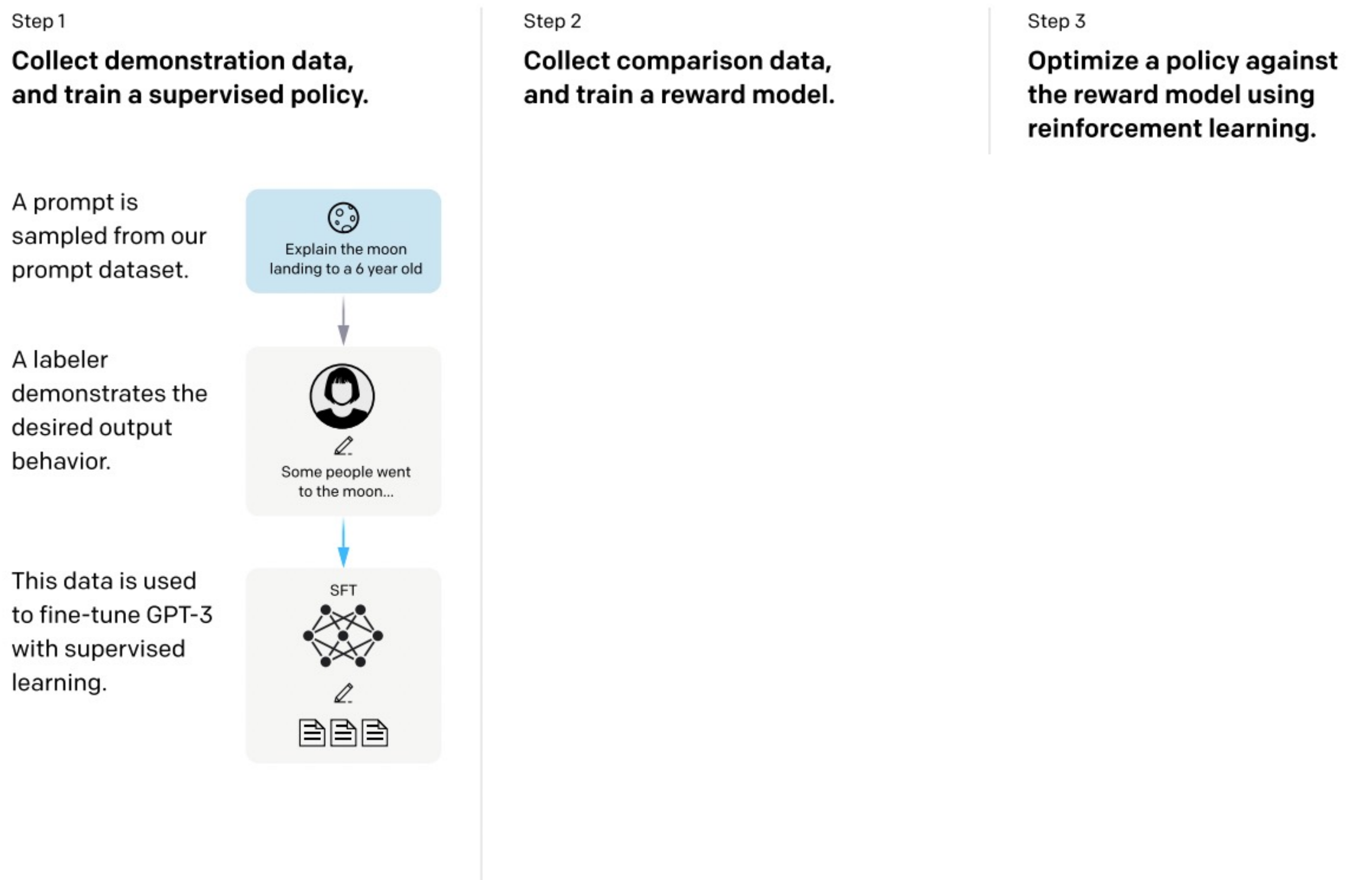


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers.

InstructGPT

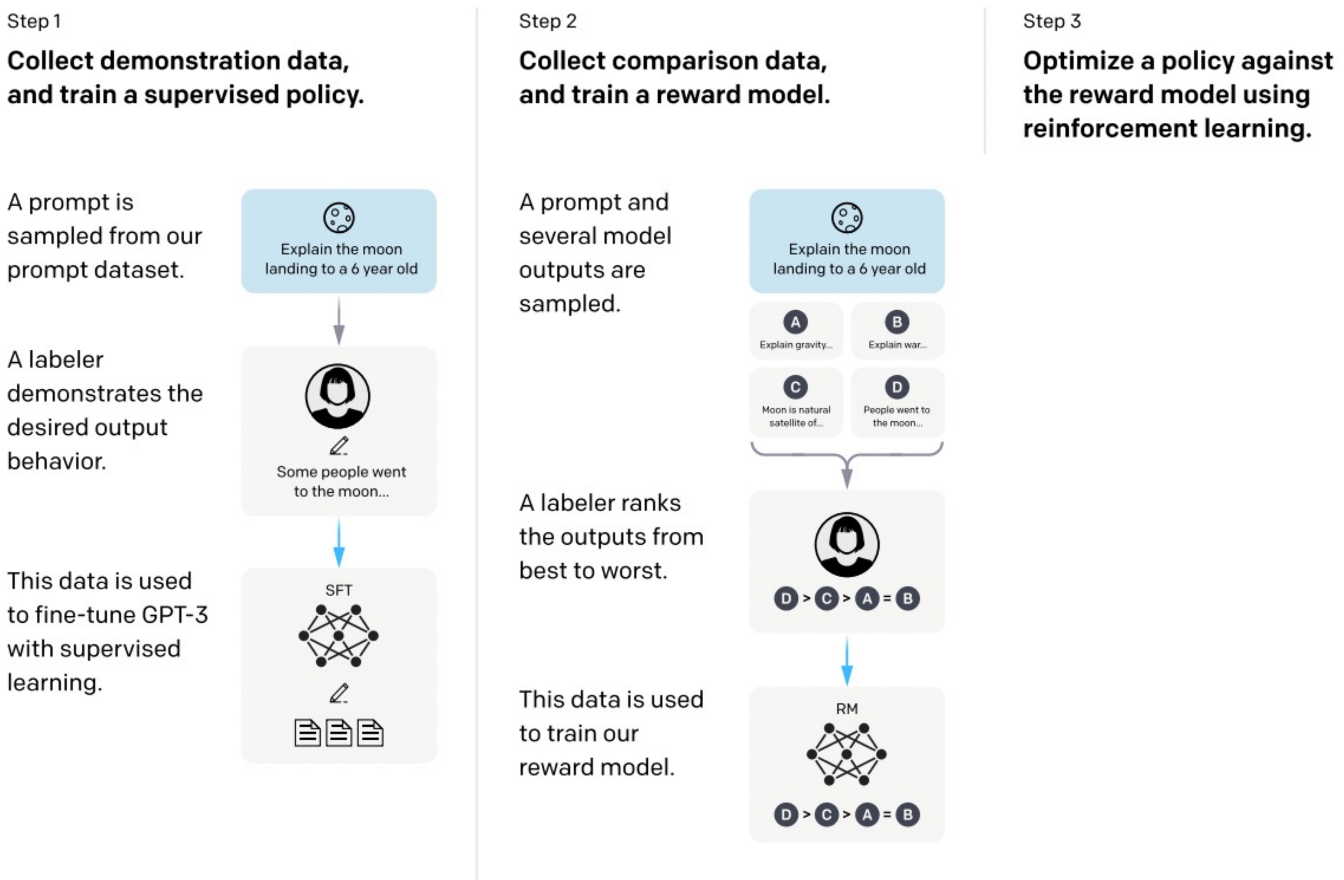


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers.

InstructGPT

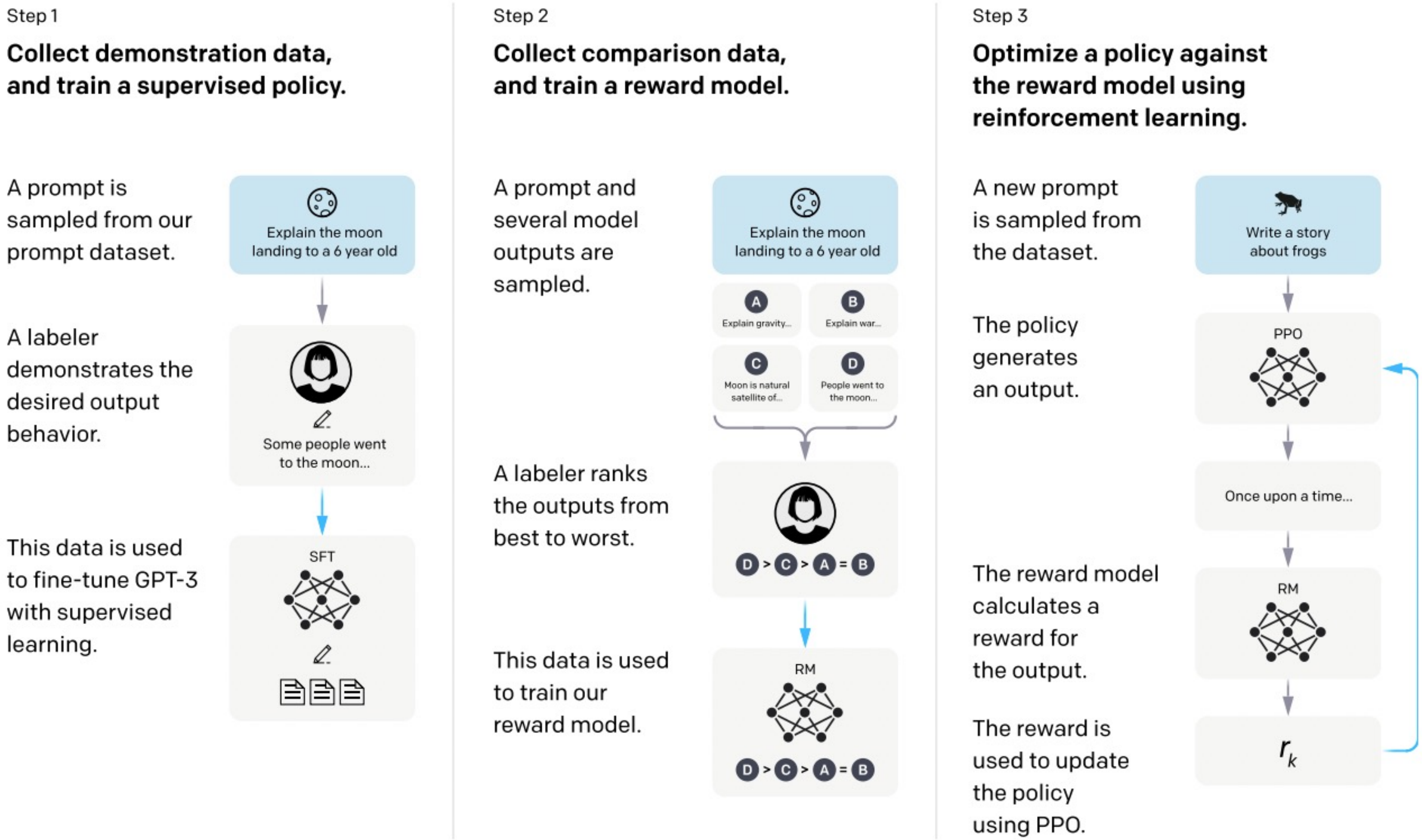


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers.

Terminology: Few-shot (k-shot)

- Given only *k* “training” examples, can the model generalize to test time. A few ways this can be realized:
  - *k*-shot fine-tuning
  - *k*-shot in-context learning (prompting)
  - *k*-shot prompt tuning
  - …

k-shot prompt tuning

- Like *k*-shot in-context prompting, keep *most* of the language model's parameters fixed / frozen ()
- But, learn *smaller* embedding models for the different tasks's prompts
- Still need a training step

Outline

Multi-task Learning

Prompting