Transfer Learning

DSCC/LING 251/451: Machine Learning with Limited Data

C.M. Downey

Spring 2026

Roadmap

Roadmap

- Last few lectures: strategies for **obtaining or extrapolating labels**
  - (Semi-Supervised Learning, Active Learning, Weak Supervision)

Roadmap

- Last few lectures: strategies for **obtaining or extrapolating labels**
  - (Semi-Supervised Learning, Active Learning, Weak Supervision)
- Today: how can you leverage **distinct but related data?**
  - (or models trained on that data)

Roadmap

- Last few lectures: strategies for **obtaining or extrapolating labels**
  - (Semi-Supervised Learning, Active Learning, Weak Supervision)
- Today: how can you leverage **distinct but related data?**
  - (or models trained on that data)
- Approaches like this fall under the umbrella of **Transfer Learning**
  - This is a **very overloaded term**: refers to a **very wide range of techniques**
  - Today we'll examine a **taxonomy of approaches** in this space

What "Transfer Learning" Could Mean

What "Transfer Learning" Could Mean

- Take a **pre-trained vision model** (e.g. ResNet) and continue **self-supervised learning** on a **new domain** (e.g. medical X-rays)

What "Transfer Learning" Could Mean

- Take a **pre-trained vision model** (e.g. ResNet) and continue **self-supervised learning** on a **new domain** (e.g. medical X-rays)
- Take a **pre-trained LM** (e.g. BERT) and **fine-tune** it for a **supervised task**

What "Transfer Learning" Could Mean

- Take a **pre-trained vision model** (e.g. ResNet) and continue **self-supervised learning** on a **new domain** (e.g. medical X-rays)
- Take a **pre-trained LM** (e.g. BERT) and **fine-tune** it for a **supervised task**
- Fine-tune a **pre-trained multilingual LM** for a supervised task on English, then **directly apply it to Swahili** ("cross-lingual transfer")

What "Transfer Learning" Could Mean

- Take a **pre-trained vision model** (e.g. ResNet) and continue **self-supervised learning** on a **new domain** (e.g. medical X-rays)
- Take a **pre-trained LM** (e.g. BERT) and **fine-tune** it for a **supervised task**
- Fine-tune a **pre-trained multilingual LM** for a supervised task on English, then **directly apply it to Swahili** ("cross-lingual transfer")
- Train an LLM on **30 tasks simultaneously** (e.g. T5 or FLAN models)

What "Transfer Learning" Could Mean

- Take a **pre-trained vision model** (e.g. ResNet) and continue **self-supervised learning** on a **new domain** (e.g. medical X-rays)
- Take a **pre-trained LM** (e.g. BERT) and **fine-tune** it for a **supervised task**
- Fine-tune a **pre-trained multilingual LM** for a supervised task on English, then **directly apply it to Swahili** ("cross-lingual transfer")
- Train an LLM on **30 tasks simultaneously** (e.g. T5 or FLAN models)
- **Distill** GPT-4 into a **compact model** (e.g. to fit on a phone)

Transfer Formally (Pan and Yang 2010)

Transfer Formally (Pan and Yang 2010)

- A **domain** $D = \{\mathcal{X}, P(X)\}$ is a pair of a **feature space** $\mathcal{X}$ and a **probability distribution** over features $P(X)$

Transfer Formally (Pan and Yang 2010)

- A **domain** $D = \{\mathcal{X}, P(X)\}$ is a pair of a **feature space** $\mathcal{X}$ and a **probability distribution** over features $P(X)$
- A **task** $T = \{\mathcal{Y}, P(Y|X)\}$ is a pair of an **output/label space** $\mathcal{Y}$ and a **conditional probability distribution** of labels given (input) features

Transfer Formally (Pan and Yang 2010)

- A **domain** $D = \{\mathcal{X}, P(X)\}$ is a pair of a **feature space** $\mathcal{X}$ and a **probability distribution** over features $P(X)$
- A **task** $T = \{\mathcal{Y}, P(Y|X)\}$ is a pair of an **output/label space** $\mathcal{Y}$ and a **conditional probability distribution** of labels given (input) features
- Consider a **source domain/task** $\{D_S, T_S\}$ as well as a **target** $\{D_T, T_T\}$

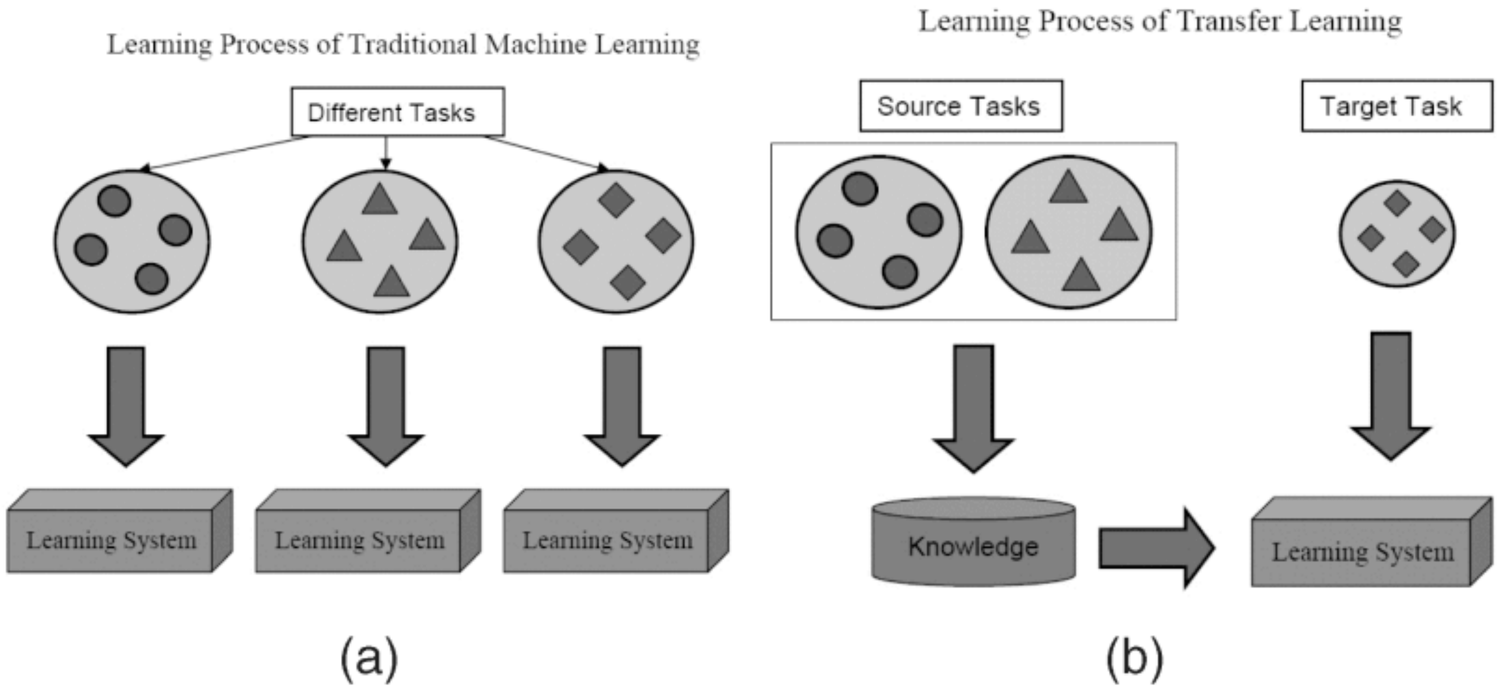
Transfer Formally (Pan and Yang 2010)

- A **domain** $D = \{\mathcal{X}, P(X)\}$ is a pair of a **feature space** $\mathcal{X}$ and a **probability distribution** over features $P(X)$
- A **task** $T = \{\mathcal{Y}, P(Y|X)\}$ is a pair of an **output/label space** $\mathcal{Y}$ and a **conditional probability distribution** of labels given (input) features
- Consider a **source domain/task** $\{D_S, T_S\}$ as well as a **target** $\{D_T, T_T\}$
- Transfer Learning: $D_S \neq D_T$, $T_S \neq T_T$, or **both**
  - ...but we can **leverage the knowledge** in D_S, T_S to help for D_T, T_T

Transfer Formally (Pan and Yang 2010)

- A **domain** $D = \{\mathcal{X}, P(X)\}$ is a pair of a **feature space** $\mathcal{X}$ and a **probability distribution** over features $P(X)$
- A **task** $T = \{\mathcal{Y}, P(Y|X)\}$ is a pair of an **output/label space** $\mathcal{Y}$ and a **conditional probability distribution** of labels given (input) features
- Consider a **source domain/task** $\{D_S, T_S\}$ as well as a **target** $\{D_T, T_T\}$
- Transfer Learning: $D_S \neq D_T$, $T_S \neq T_T$, or **both**
  - ...but we can **leverage the knowledge** in D_S, T_S to help for D_T, T_T
- This definition is **still very broad!**

Transfer Learning Schematic



Pan and Yang (2010)

Axes of Transfer

Axis 1: What differs?

	Same Task	Different Task
Same Domain	Traditional ML	Inductive Transfer
Different Domain	Domain Adaptation (no common name)	

Axis 1: What differs?

- **"Inductive Transfer"**: same domain (data distribution), different task

	Same Task	Different Task
Same Domain	Traditional ML	Inductive Transfer
Different Domain	Domain Adaptation (no common name)	

Axis 1: What differs?

- **"Inductive Transfer"**: same domain (data distribution), different task
- **Domain Adaptation** ("Transductive Transfer"): different domain, same task

	Same Task	Different Task
Same Domain	Traditional ML	Inductive Transfer
Different Domain	Domain Adaptation (no common name)	

Axis 1: What differs?

- **"Inductive Transfer"**: same domain (data distribution), different task
- **Domain Adaptation** ("Transductive Transfer"): different domain, same task
- **No formal name** for the case where **both** domain and task differ

	Same Task	Different Task
Same Domain	Traditional ML	Inductive Transfer
Different Domain	Domain Adaptation (no common name)	

Axis 1: What differs?

- **"Inductive Transfer"**: same domain (data distribution), different task
- **Domain Adaptation** ("Transductive Transfer"): different domain, same task
- **No formal name** for the case where **both** domain and task differ
- (Names in quotes come from Pan and Yang)

	Same Task	Different Task
Same Domain	Traditional ML	Inductive Transfer
Different Domain	Domain Adaptation (no common name)	

Axis 2: What transfers?

Axis 2: What transfers?

- **Instance transfer:** select or up-weight relevant source datapoints
  - I.e. "find the source data that looks most like the target"
  - **Caveat:** this tends to not be used as much anymore (at least on its own)

Axis 2: What transfers?

- **Instance transfer:** select or up-weight relevant source datapoints
  - I.e. "find the source data that looks most like the target"
  - **Caveat:** this tends to not be used as much anymore (at least on its own)
- **Representation transfer:** learn a shared feature space for source/target
  - **Very common** in modern ML/DL; example: multilingual or multitask models

Axis 2: What transfers?

- **Instance transfer:** select or up-weight relevant source datapoints
  - I.e. "find the source data that looks most like the target"
  - **Caveat:** this tends to not be used as much anymore (at least on its own)
- **Representation transfer:** learn a shared feature space for source/target
  - **Very common** in modern ML/DL; example: multilingual or multitask models
- **Parameter transfer:** re-use and adapt weights trained for source
  - Also **extremely common**, probably the standard form of transfer today

Axis 2: What transfers?

- **Instance transfer:** select or up-weight relevant source datapoints
  - I.e. "find the source data that looks most like the target"
  - **Caveat:** this tends to not be used as much anymore (at least on its own)
- **Representation transfer:** learn a shared feature space for source/target
  - **Very common** in modern ML/DL; example: multilingual or multitask models
- **Parameter transfer:** re-use and adapt weights trained for source
  - Also **extremely common**, probably the standard form of transfer today
- **Knowledge transfer:** transfer model behavior/decisions rather than parameters
  - Most common: **"knowledge distillation"**, where a small "student" mimics a "teacher" model

Axis 2: What transfers?

- **Instance transfer:** select or up-weight relevant source datapoints
  - I.e. "find the source data that looks most like the target"
  - **Caveat:** this tends to not be used as much anymore (at least on its own)
- **Representation transfer:** learn a shared feature space for source/target
  - **Very common** in modern ML/DL; example: multilingual or multitask models
- **Parameter transfer:** re-use and adapt weights trained for source
  - Also **extremely common**, probably the standard form of transfer today
- **Knowledge transfer:** transfer model behavior/decisions rather than parameters
  - Most common: **"knowledge distillation"**, where a small "student" mimics a "teacher" model
- These are **not mutually exclusive:** fine-tuning a pre-trained NN is both representation and parameter transfer

Axis 3: When does transfer happen?

Axis 3: When does transfer happen?

- **Sequential:** train on source, then adapt to target
  - I.e. the **pre-train + fine-tune** paradigm

Axis 3: When does transfer happen?

- **Sequential:** train on source, then adapt to target
  - I.e. the **pre-train + fine-tune** paradigm
- **Simultaneous:** train on source and target at the **same time**
  - Example: **multi-task learning**

Axis 3: When does transfer happen?

- **Sequential:** train on source, then adapt to target
  - I.e. the **pre-train + fine-tune** paradigm
- **Simultaneous:** train on source and target at the **same time**
  - Example: **multi-task learning**
- **Continual:** keep transferring to **more tasks over time**
  - Can be hard to prevent **forgetting previous tasks**

Mapping the Opening Examples

Example	What Differs	What Transfers	When
ResNet > X-Rays	domain	params, features	sequential
BERT > task	task	params, features	sequential
English task > Swahili	domain (language)	params, features	sequential
T5/FLAN	tasks (multiple)	params, features	simultaneous
GPT-4 > small version	model	knowledge	sequential

Pre-training + Fine-tuning

(Sequential Transfer)

Pre-training as Transfer

Pre-training as Transfer

- We've already seen sequential transfer in **self-supervised pre-training**
  - Source task: the **pretext / surrogate task** (LM, SimCLR, etc.)
  - Target task: the **actual downstream problem** (classification, QA, etc.)
  - What transfers: the **parameters** and **representations of the data**

Pre-training as Transfer

- We've already seen sequential transfer in **self-supervised pre-training**
  - Source task: the **pretext / surrogate task** (LM, SimCLR, etc.)
  - Target task: the **actual downstream problem** (classification, QA, etc.)
  - What transfers: the **parameters** and **representations of the data**
- Before about 2018, "transfer learning" in NLP meant using **pre-trained word vectors** (word2vec, GloVe)
  - Howard & Ruder (2018), ULMFiT: pre-train an LM then **fine-tune on many tasks**
  - BERT (2019): shifted to **pre-train + fine-tune** as the **default paradigm**

Why does it work?

Why does it work?

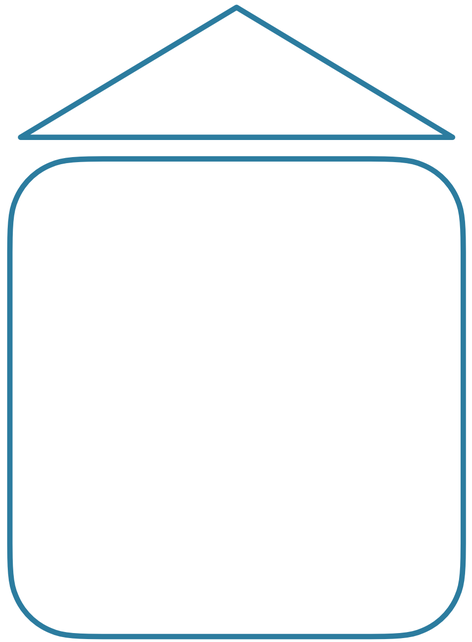
- For well-constructed pretext tasks, model has to learn **general-purpose representations of the data**
  - E.g. LMs learn about syntax, semantics, world knowledge
  - These features give a **head-start for solving downstream tasks**

Why does it work?

- For well-constructed pretext tasks, model has to learn **general-purpose representations of the data**
  - E.g. LMs learn about syntax, semantics, world knowledge
  - These features give a **head-start for solving downstream tasks**
- Sometimes fine-tuning involves **freezing the pre-trained model** and **only training task-specific layers**
  - Shows the main model functions as a **general-purpose feature extractor**
  - Different downstream tasks make use of the **shared feature space**

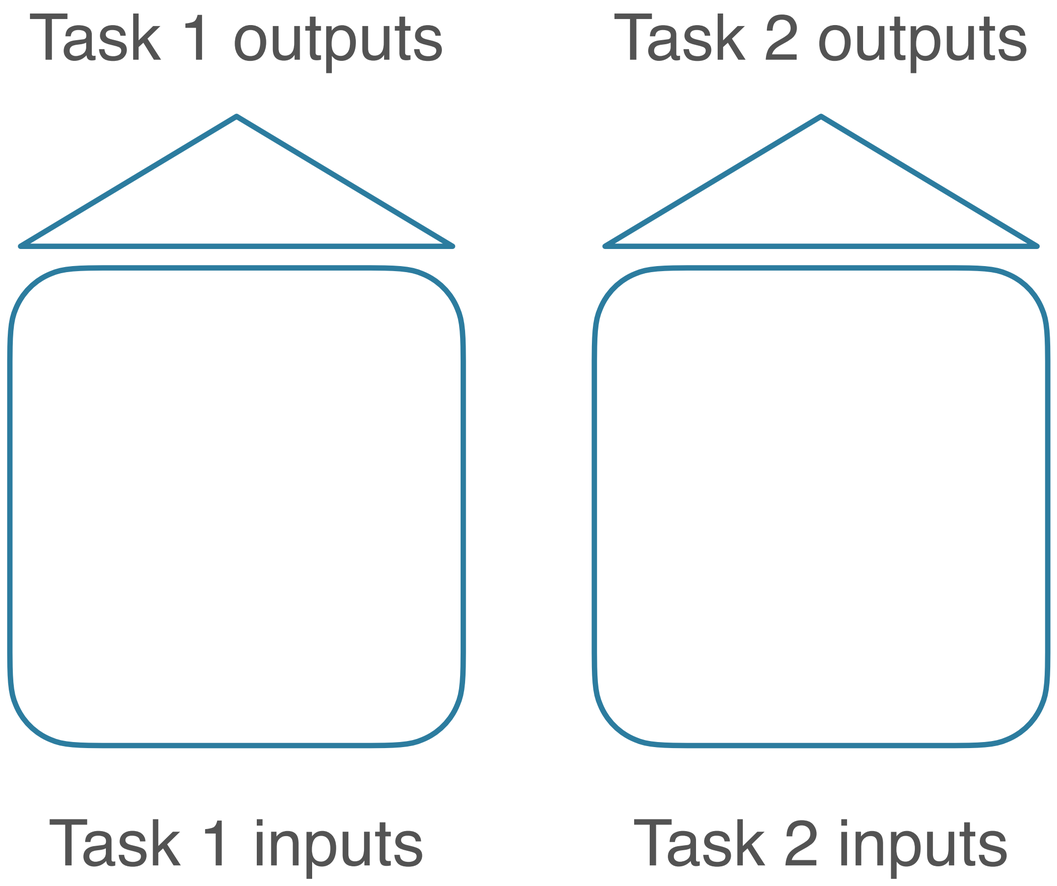
Traditional Learning

Task 1 outputs

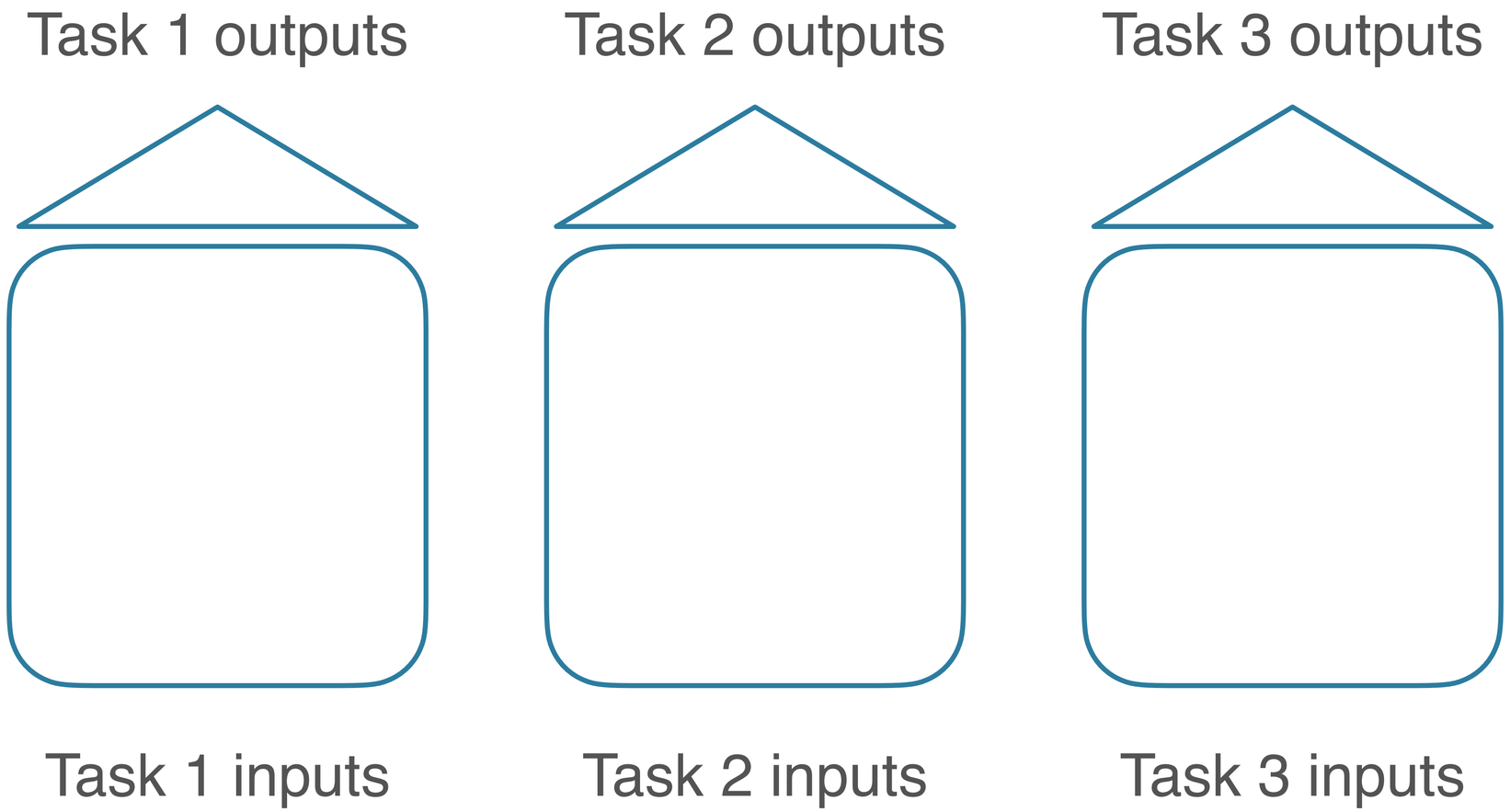


Task 1 inputs

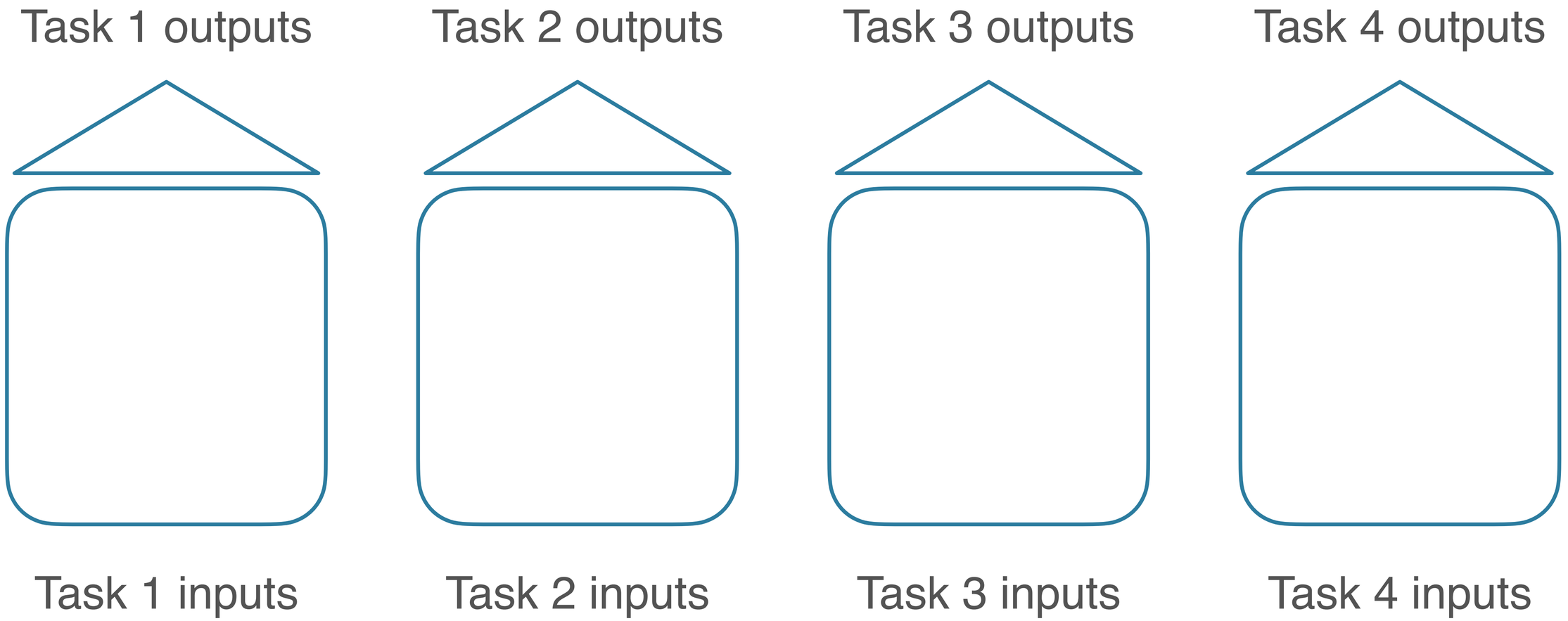
Traditional Learning



Traditional Learning

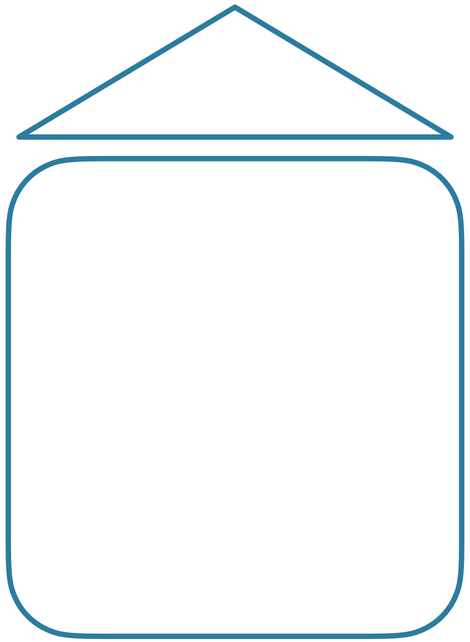


Traditional Learning



Pre-training / Representation Learning

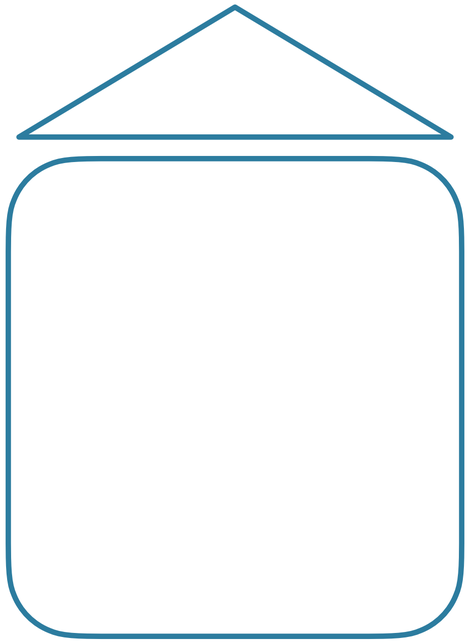
“pre-training” task outputs



“pre-training” task inputs

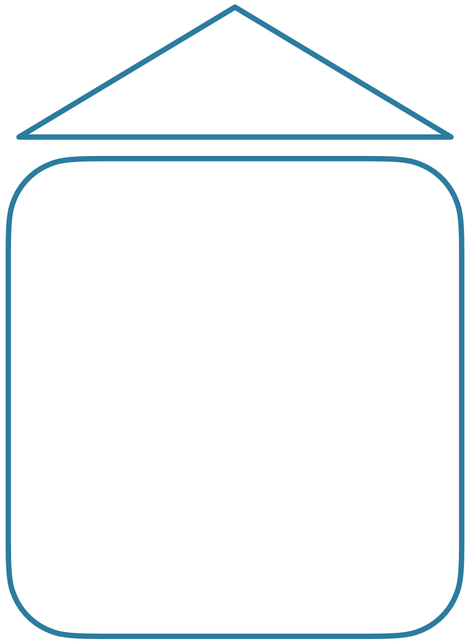
Pre-training / Representation Learning

“pre-training” task outputs



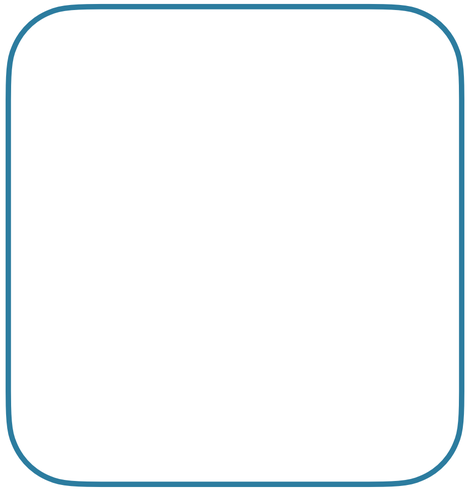
Pre-training / Representation Learning

“pre-training” task outputs



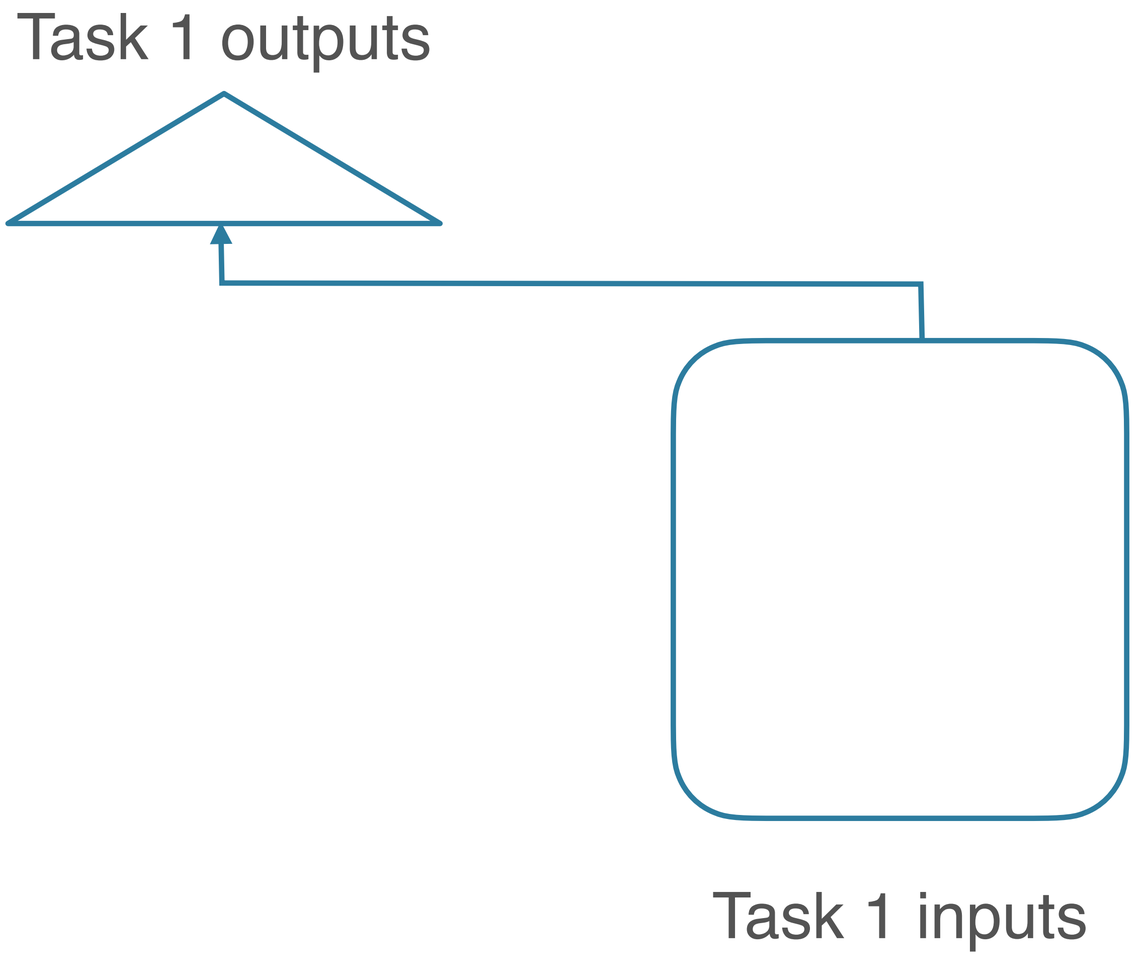
Task 1 inputs

Pre-training / Representation Learning

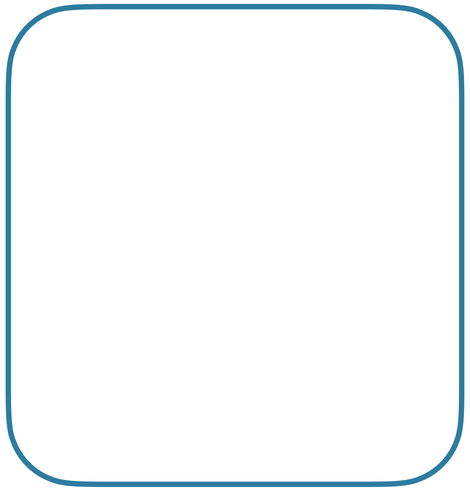


Task 1 inputs

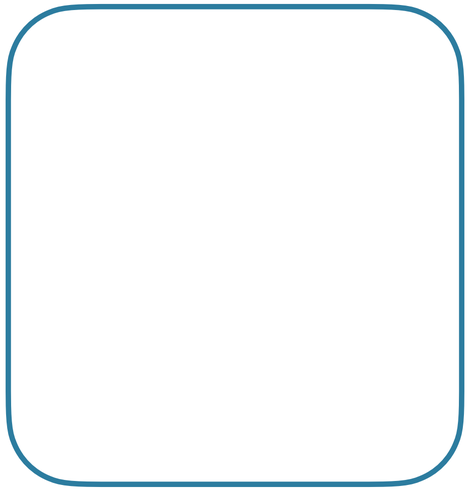
Pre-training / Representation Learning



Pre-training / Representation Learning

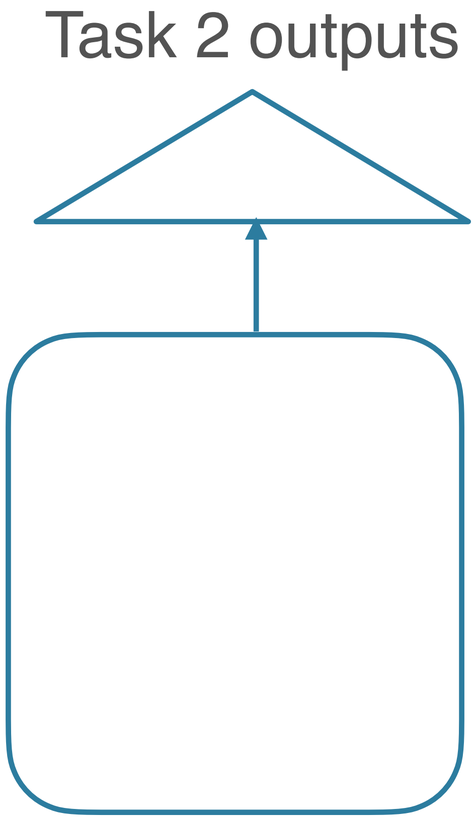


Pre-training / Representation Learning

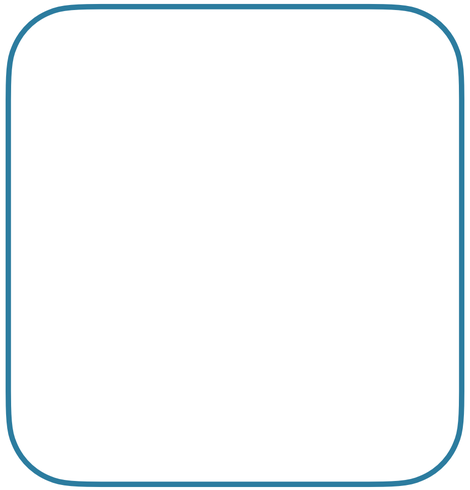


Task 2 inputs

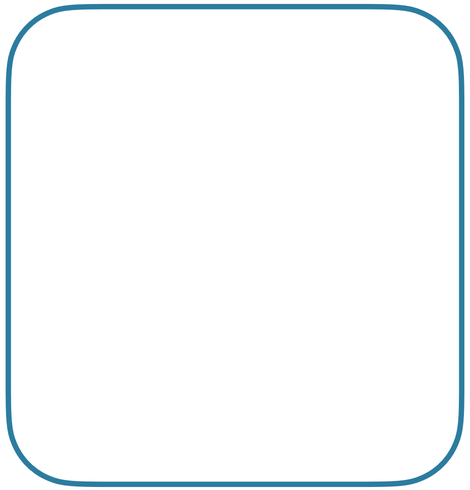
Pre-training / Representation Learning



Pre-training / Representation Learning

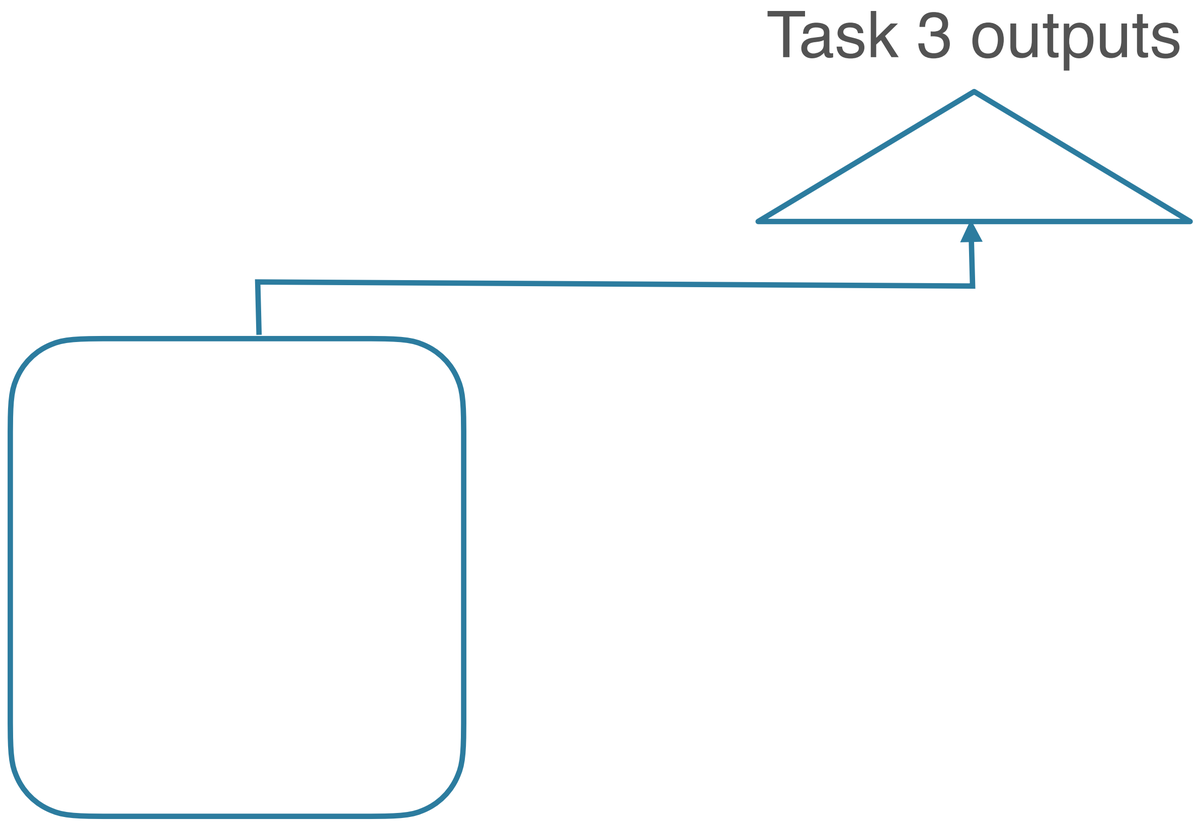


Pre-training / Representation Learning

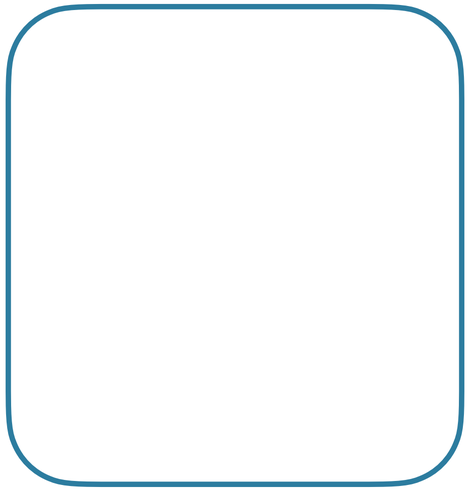


Task 3 inputs

Pre-training / Representation Learning



Pre-training / Representation Learning



When does it work well?

When does it work well?

- Source **data size** and **diversity** matter
  - More data leads to **more robust features**, improving transfer (why **pre-training scale has dominated** in recent years)

When does it work well?

- Source **data size** and **diversity** matter
  - More data leads to **more robust features**, improving transfer (why **pre-training scale has dominated** in recent years)
- **Domain match** between source and target
  - General pre-training data (e.g. Wikipedia, web text) transfers well to many tasks
  - **Domain-specific** pre-training (e.g. scientific texts) transfers better to **in-domain tasks** (but tends to be worse out of domain)

When does it work well?

- Source **data size** and **diversity** matter
  - More data leads to **more robust features**, improving transfer (why **pre-training scale has dominated** in recent years)
- **Domain match** between source and target
  - General pre-training data (e.g. Wikipedia, web text) transfers well to many tasks
  - **Domain-specific** pre-training (e.g. scientific texts) transfers better to **in-domain tasks** (but tends to be worse out of domain)
- Choice of **pretext task** may also have an effect (some tasks might match up more or less well)

When does it work well?

- Source **data size** and **diversity** matter
  - More data leads to **more robust features**, improving transfer (why **pre-training scale has dominated** in recent years)
- **Domain match** between source and target
  - General pre-training data (e.g. Wikipedia, web text) transfers well to many tasks
  - **Domain-specific** pre-training (e.g. scientific texts) transfers better to **in-domain tasks** (but tends to be worse out of domain)
- Choice of **pretext task** may also have an effect (some tasks might match up more or less well)
- Generally **relatedness** between source and target is important. But what kind?
  - Picking the **wrong source** can sometimes hurt transfer (**negative transfer**)

Cross-lingual Transfer

Language Transfer

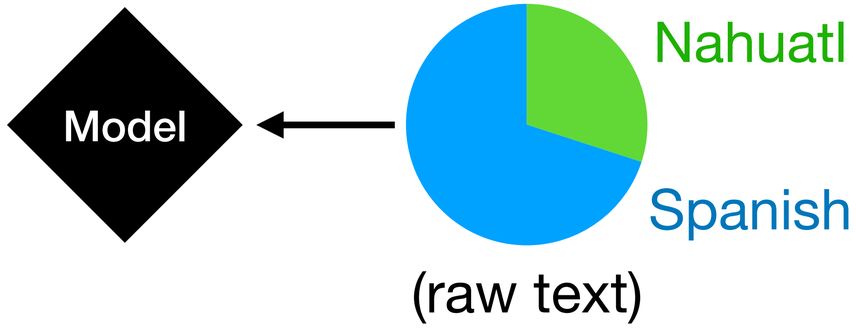
Language Transfer

- Recent approach: train **single LM** on text from **~50-500 languages**
  - No "parallel text" typically used for translation

Language Transfer

- Recent approach: train **single LM** on text from **~50-500 languages**
  - No "parallel text" typically used for translation

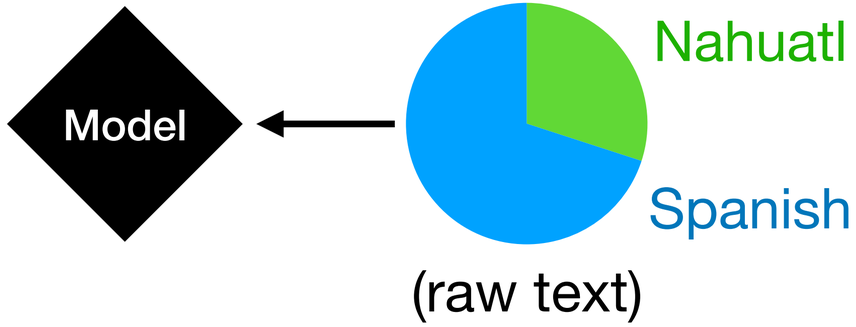
1. Bilingual language modeling



Language Transfer

- Recent approach: train **single LM** on text from **~50-500 languages**
  - No "parallel text" typically used for translation
- Resulting models show ability to **transfer task performance across languages**
  - (At least to a level far above chance)
  - How does it accomplish this **without translations?**
  - Usually attributed to **shared representation space** across languages

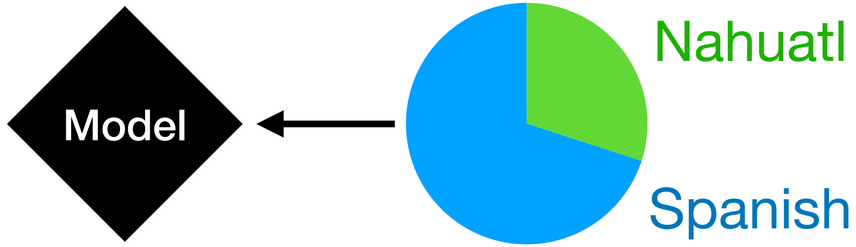
1. Bilingual language modeling



Language Transfer

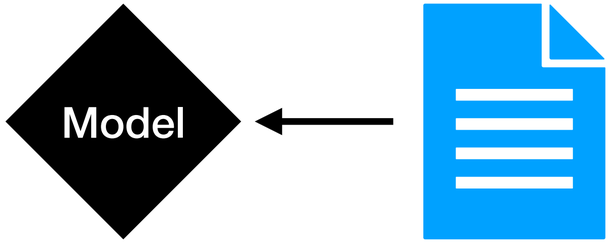
- Recent approach: train **single LM** on text from **~50-500 languages**
  - No "parallel text" typically used for translation
- Resulting models show ability to **transfer task performance across languages**
  - (At least to a level far above chance)
  - How does it accomplish this **without translations?**
  - Usually attributed to **shared representation space** across languages

1. Bilingual language modeling



(raw text)

2. Spanish task training

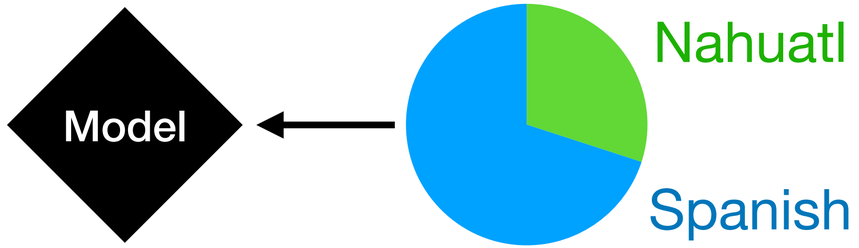


Spanish task training data

Language Transfer

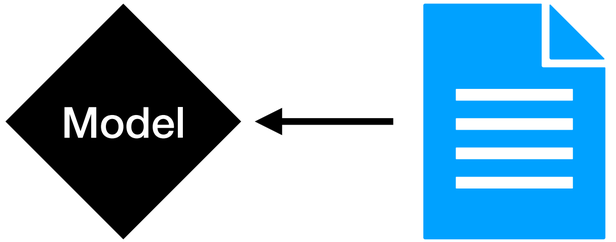
- Recent approach: train **single LM** on text from **~50-500 languages**
  - No "parallel text" typically used for translation
- Resulting models show ability to **transfer task performance across languages**
  - (At least to a level far above chance)
  - How does it accomplish this **without translations?**
  - Usually attributed to **shared representation space** across languages

1. Bilingual language modeling



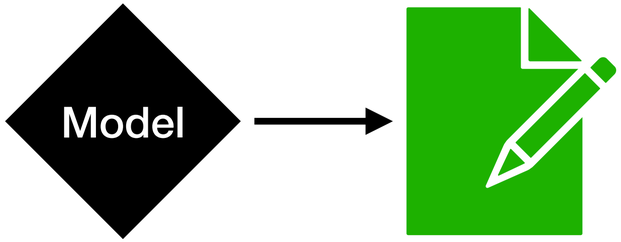
(raw text)

2. Spanish task training



Spanish task training data

3. Transfer to Nahuatl task



Nahuatl task test data

What enables this?

What enables this?

- **Shared (sub-word) vocabulary**: different languages share at least some units that the model uses to build text
  - Especially beneficial for languages with **shared scripts, cognates, borrowings**

What enables this?

- **Shared (sub-word) vocabulary**: different languages share at least some units that the model uses to build text
  - Especially beneficial for languages with **shared scripts, cognates, borrowings**
- **Shared model parameters:** all languages are processed by the same weights, model needs at least some **language-general representations**

What enables this?

- **Shared (sub-word) vocabulary**: different languages share at least some units that the model uses to build text
  - Especially beneficial for languages with **shared scripts, cognates, borrowings**
- **Shared model parameters:** all languages are processed by the same weights, model needs at least some **language-general representations**
- **Shared linguistic structure:** there is **some universal structure** to languages, and **related languages** share much more than the minimum

What enables this?

- **Shared (sub-word) vocabulary**: different languages share at least some units that the model uses to build text
  - Especially beneficial for languages with **shared scripts, cognates, borrowings**
- **Shared model parameters:** all languages are processed by the same weights, model needs at least some **language-general representations**
- **Shared linguistic structure:** there is **some universal structure** to languages, and **related languages** share much more than the minimum
- Where does it **not work?**
  - Typologically **distant/dissimilar languages**
  - Languages with **very little pre-training data** (which is most of them)

Multi-Task Learning

Multi-task Learning

$$\mathscr{L}_{\text{MTL}} = \sum_{t=1}^{T} \lambda_t \mathscr{L}_t(\theta_{\text{shared}}, \theta_t)$$

θ_{shared}: shared parameters

θ_t: task-specific parameters

λ_t: task weights

Multi-task Learning

- Instead of training on source then target, train on **multiple tasks at once**

$$\mathscr{L}_{\text{MTL}} = \sum_{t=1}^{T} \lambda_t \mathscr{L}_t(\theta_{\text{shared}}, \theta_t)$$

θ_{shared}: shared parameters

θ_t: task-specific parameters

λ_t: task weights

Multi-task Learning

- Instead of training on source then target, train on **multiple tasks at once**
- Key point: tasks **share model parameters** (sometimes all of them)

$$\mathcal{L}_{\text{MTL}} = \sum_{t=1}^{T} \lambda_t \mathcal{L}_t(\theta_{\text{shared}}, \theta_t)$$

θ_{shared}: shared parameters

θ_t: task-specific parameters

λ_t: task weights

Multi-task Learning

- Instead of training on source then target, train on **multiple tasks at once**
- Key point: tasks **share model parameters** (sometimes all of them)
- Why should this help?
  - **Implicit regularization:** each task is a different "view" of the data
  - **Attention focusing:** different tasks can help the model notice important features
  - A hard task might **benefit from easier task training**

$$\mathscr{L}_{\text{MTL}} = \sum_{t=1}^{T} \lambda_t \mathscr{L}_t(\theta_{\text{shared}}, \theta_t)$$

θ_{shared}: shared parameters

θ_t: task-specific parameters

λ_t: task weights

Traditional Approach: Task Heads

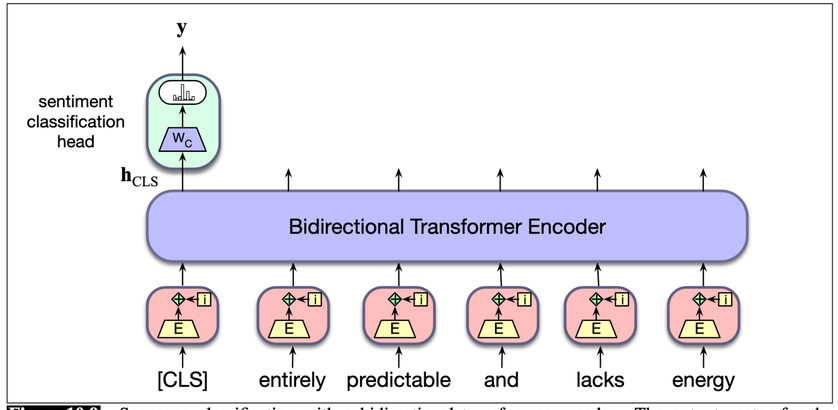


Figure 10.9 Sequence classification with a bidirectional transformer encoder. The output vector for the [CLS] token serves as input to a simple classifier.

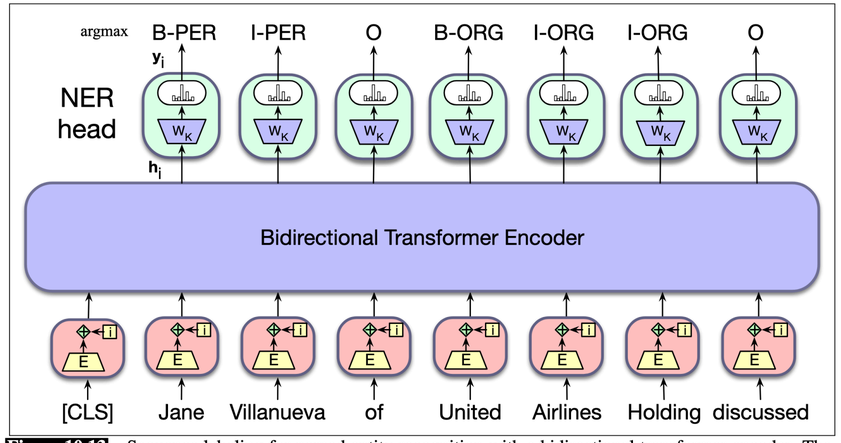


Figure 10.13 Sequence labeling for named entity recognition with a bidirectional transformer encoder. The output vector for each input token is passed to a simple k-way classifier.

Traditional Approach: Task Heads

- In the "BERT Era", classification tasks were done with **small task-specific layers** called "heads"

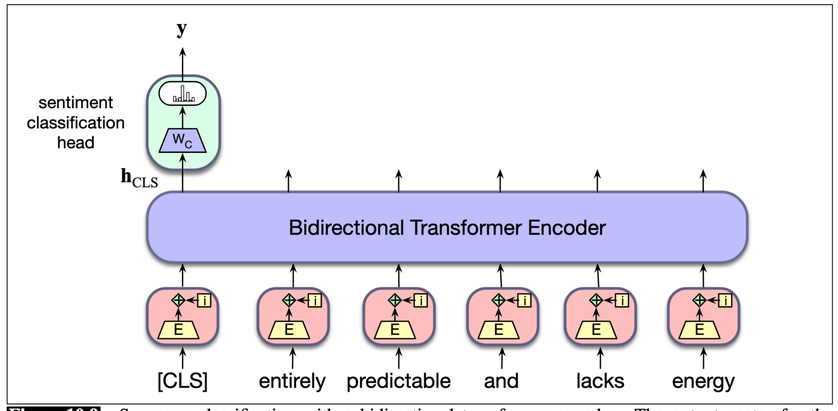


Figure 10.9 Sequence classification with a bidirectional transformer encoder. The output vector for the [CLS] token serves as input to a simple classifier.

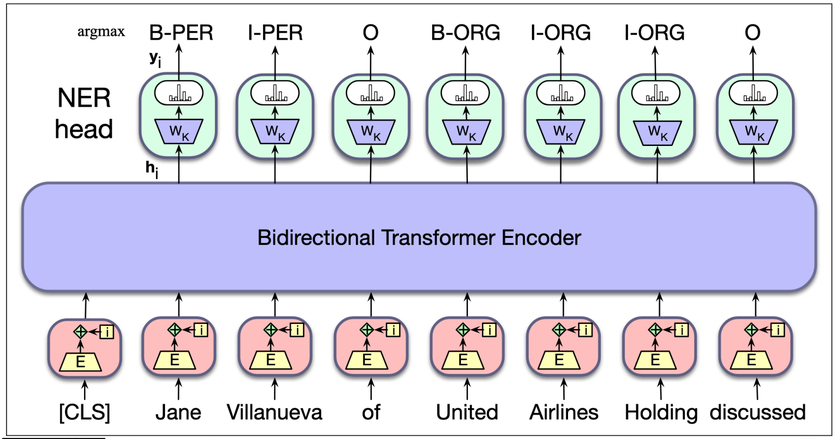


Figure 10.13 Sequence labeling for named entity recognition with a bidirectional transformer encoder. The output vector for each input token is passed to a simple k-way classifier.

Traditional Approach: Task Heads

- In the "BERT Era", classification tasks were done with **small task-specific layers** called "heads"
  - E.g. linear layer + softmax for **five-way sentiment classification**

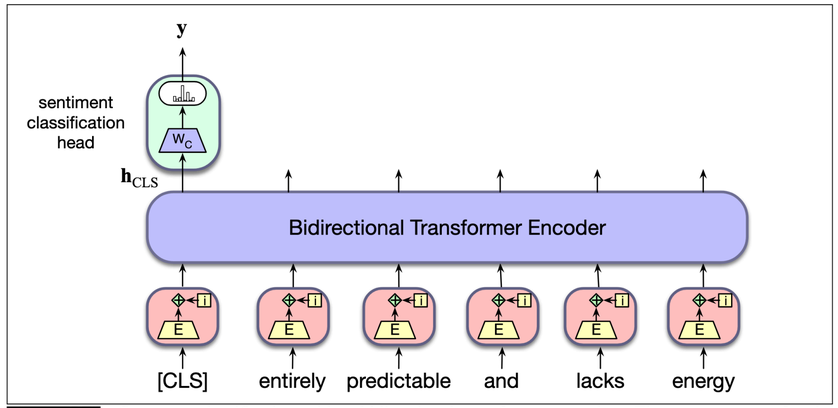


Figure 10.9 Sequence classification with a bidirectional transformer encoder. The output vector for the [CLS] token serves as input to a simple classifier.

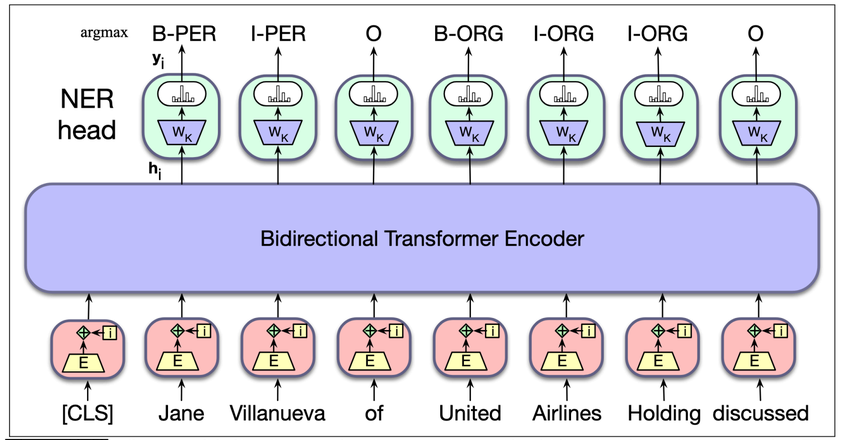


Figure 10.13 Sequence labeling for named entity recognition with a bidirectional transformer encoder. The output vector for each input token is passed to a simple k-way classifier.

Traditional Approach: Task Heads

- In the "BERT Era", classification tasks were done with **small task-specific layers** called "heads"
  - E.g. linear layer + softmax for **five-way sentiment classification**
  - Main model is sometimes **frozen** (only task head trains)

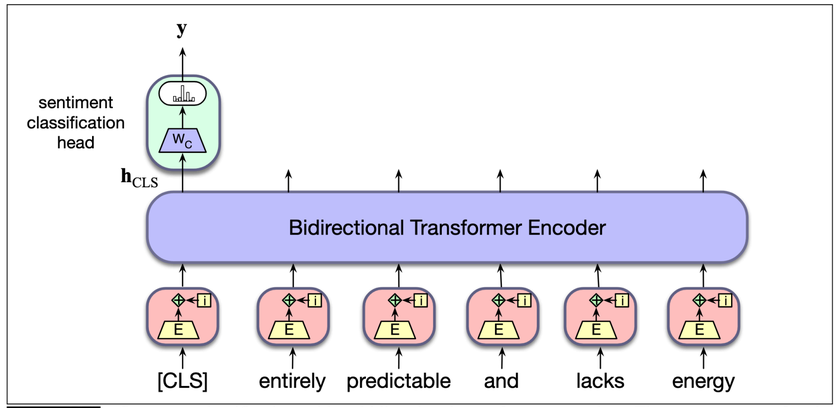


Figure 10.9 Sequence classification with a bidirectional transformer encoder. The output vector for the [CLS] token serves as input to a simple classifier.

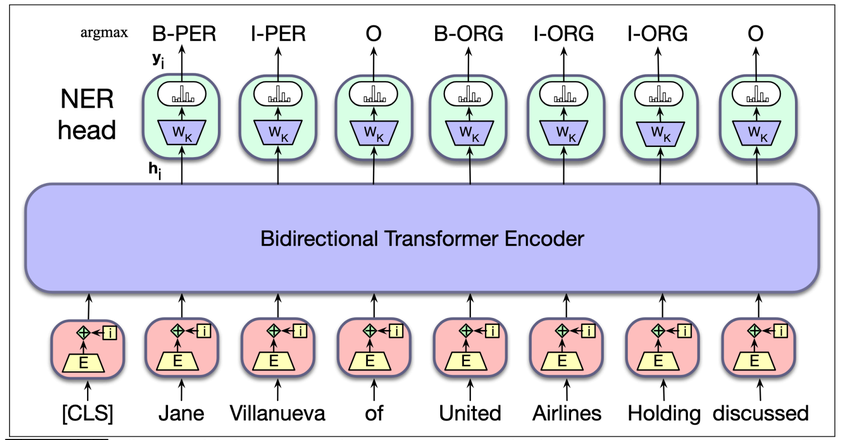


Figure 10.13 Sequence labeling for named entity recognition with a bidirectional transformer encoder. The output vector for each input token is passed to a simple k-way classifier.

Traditional Approach: Task Heads

- In the "BERT Era", classification tasks were done with **small task-specific layers** called "heads"
  - E.g. linear layer + softmax for **five-way sentiment classification**
  - Main model is sometimes **frozen** (only task head trains)
- Problem: each new task requires a **new task head**

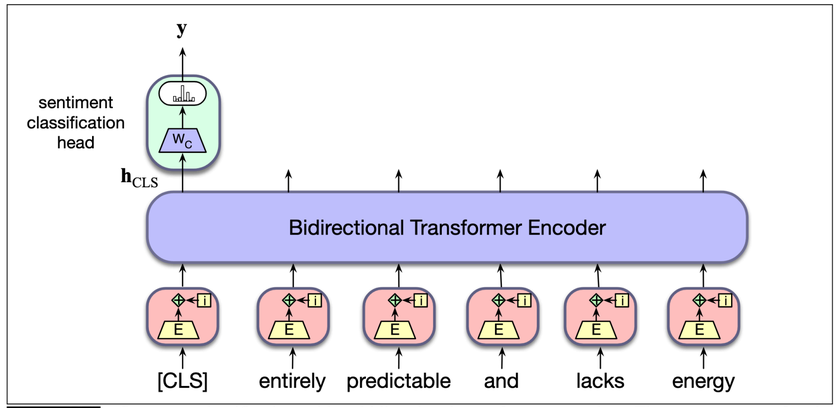


Figure 10.9 Sequence classification with a bidirectional transformer encoder. The output vector for the [CLS] token serves as input to a simple classifier.

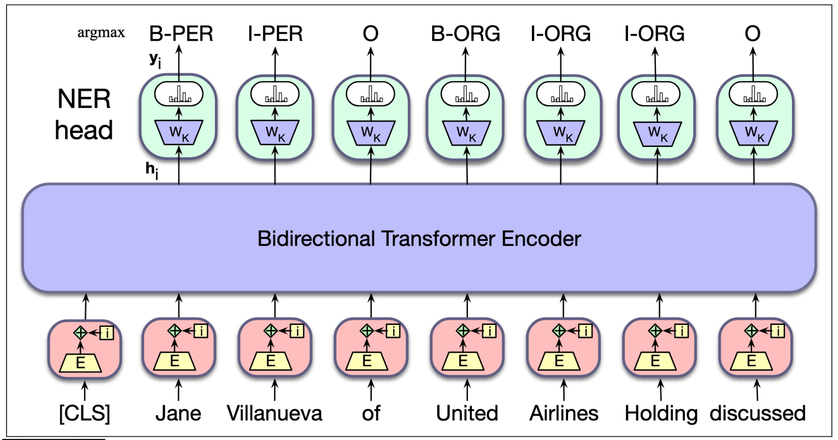
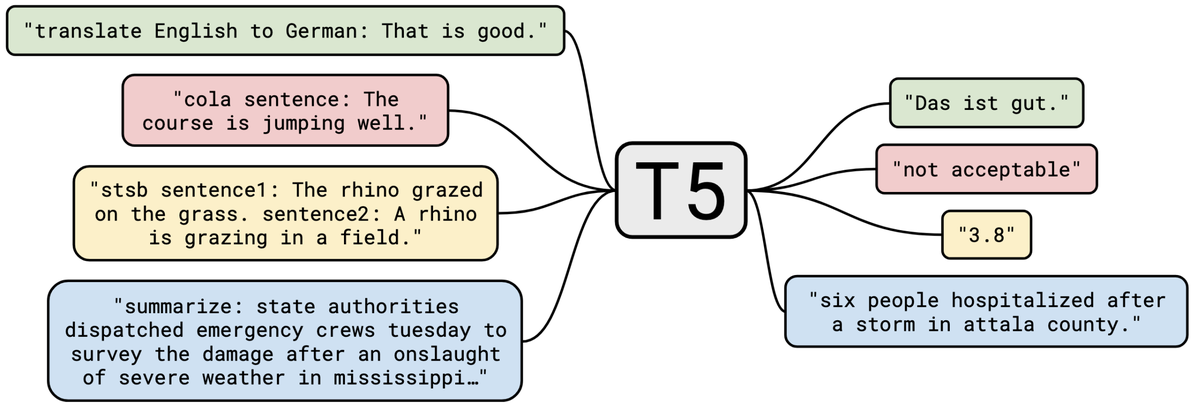


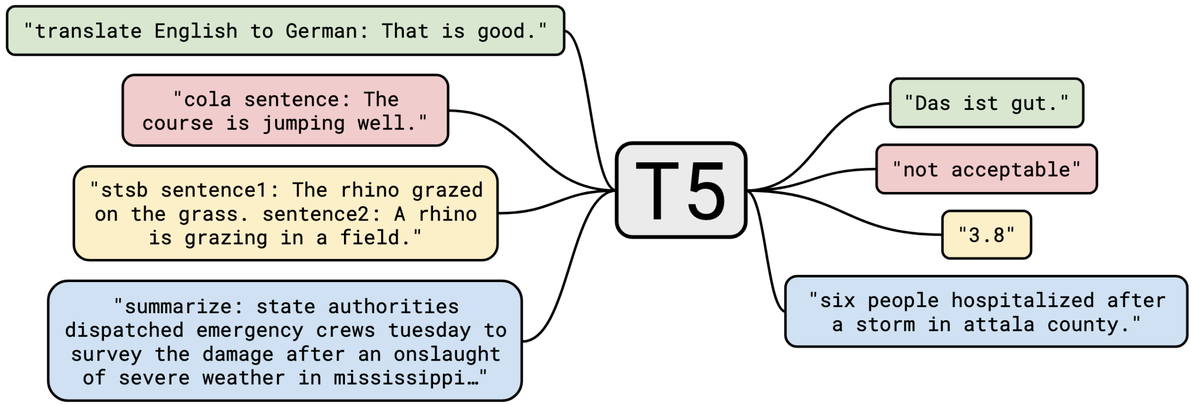
Figure 10.13 Sequence labeling for named entity recognition with a bidirectional transformer encoder. The output vector for each input token is passed to a simple k-way classifier.

Text-to-Text NLP



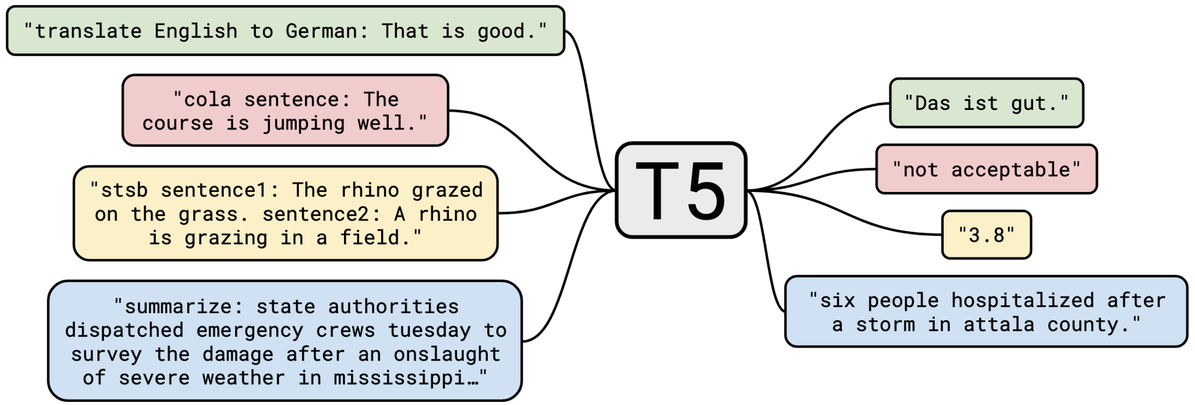
Text-to-Text NLP

- Train LMs to all tasks with **unified text output**
  - **Bypasses** classification heads
  - Well suited for **problems posed as text**



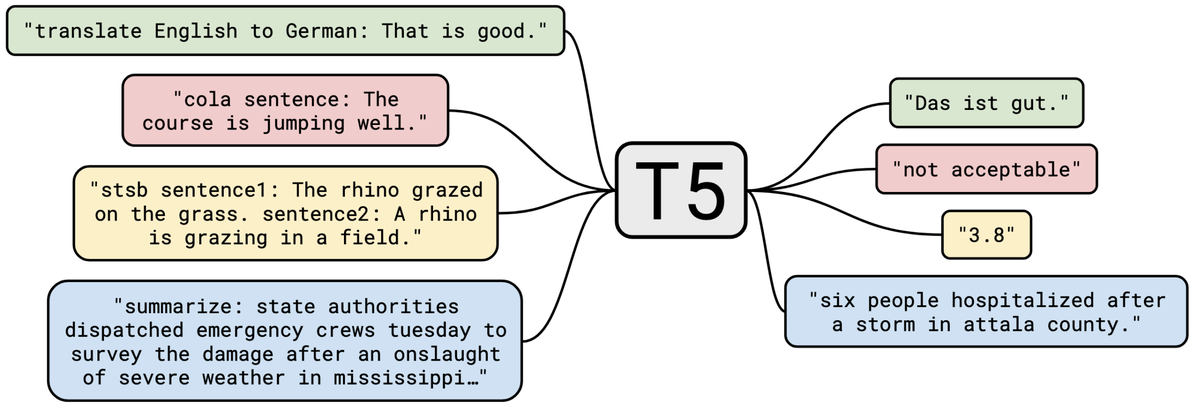
Text-to-Text NLP

- Train LMs to all tasks with **unified text output**
  - **Bypasses** classification heads
  - Well suited for **problems posed as text**
- **Explicit training** (does gradient updates)



Text-to-Text NLP

- Train LMs to all tasks with **unified text output**
  - **Bypasses** classification heads
  - Well suited for **problems posed as text**
- **Explicit training** (does gradient updates)
- Popularized by Google's **T5 model**
  - "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer"



Converting Tasks to Language Modeling

D.2 RTE

Original input:

Sentence 1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians).

Sentence 2: Slovenia has 3,000 inhabitants.

Processed input: rte sentence1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians). sentence2: Slovenia has 3,000 inhabitants.

Original target: 1

Processed target: not_entailment

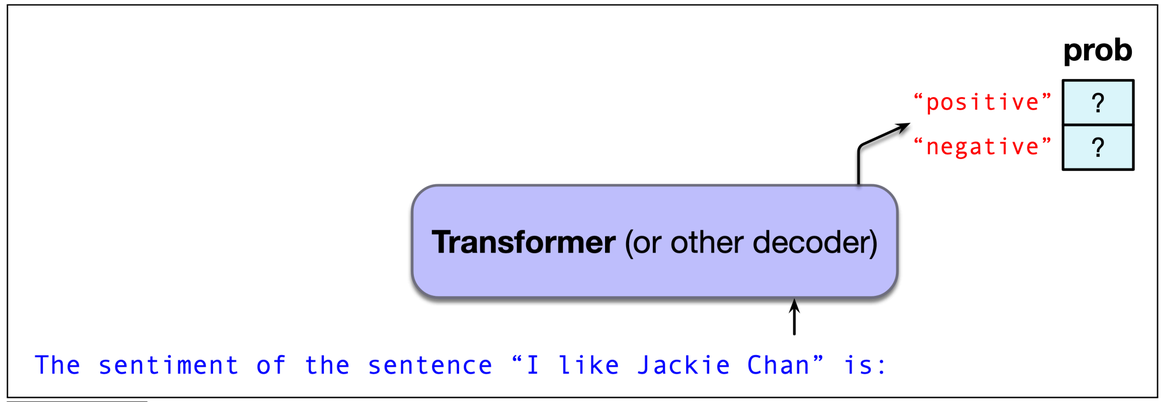


Figure 7.4 Computing the probabilities of the tokens `positive` and `negative` occurring after this prefix.

Converting Tasks to Language Modeling

- Take a supervised task and find a way to **pose it as text**

D.2 RTE

Original input:

Sentence 1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians).

Sentence 2: Slovenia has 3,000 inhabitants.

Processed input: rte sentence1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians). sentence2: Slovenia has 3,000 inhabitants.

Original target: 1

Processed target: not_entailment

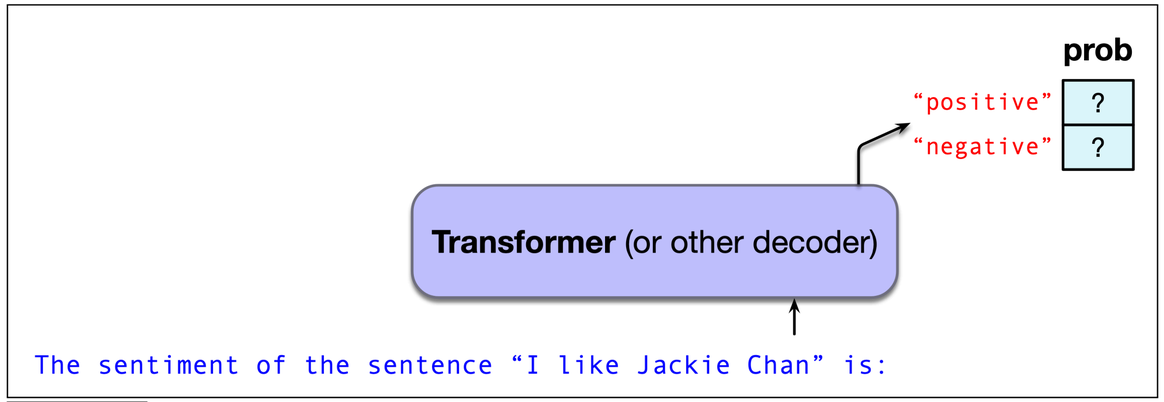


Figure 7.4 Computing the probabilities of the tokens `positive` and `negative` occurring after this prefix.

Converting Tasks to Language Modeling

- Take a supervised task and find a way to **pose it as text**
- For T5, the input is **prefixed with the task name** (e.g. `rte`)

D.2 RTE

Original input:

Sentence 1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians).

Sentence 2: Slovenia has 3,000 inhabitants.

Processed input: rte sentence1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians). sentence2: Slovenia has 3,000 inhabitants.

Original target: 1

Processed target: not_entailment

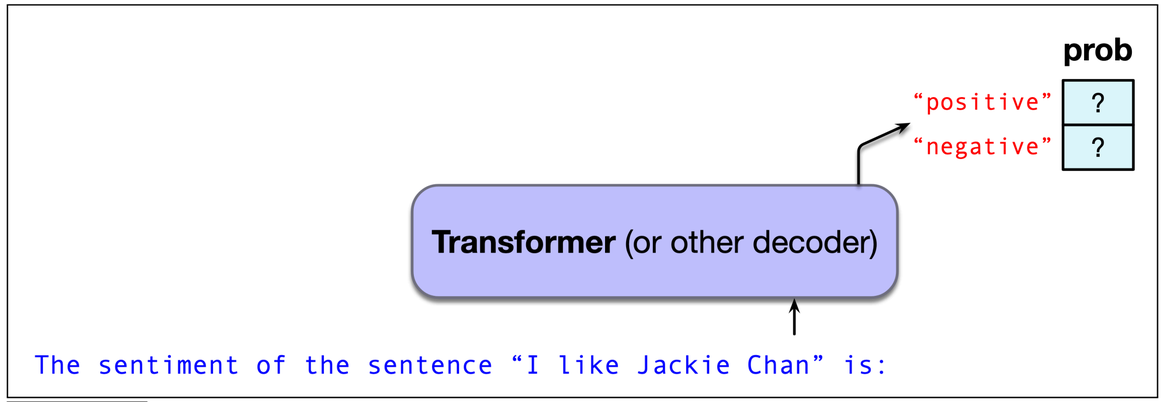


Figure 7.4 Computing the probabilities of the tokens `positive` and `negative` occurring after this prefix.

Converting Tasks to Language Modeling

- Take a supervised task and find a way to **pose it as text**
- For T5, the input is **prefixed with the task name** (e.g. `rte`)
- The task is now to **generate the tokens** of the **correct answer**

D.2 RTE

Original input:

Sentence 1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians).

Sentence 2: Slovenia has 3,000 inhabitants.

Processed input: rte sentence1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians). sentence2: Slovenia has 3,000 inhabitants.

Original target: 1

Processed target: not_entailment

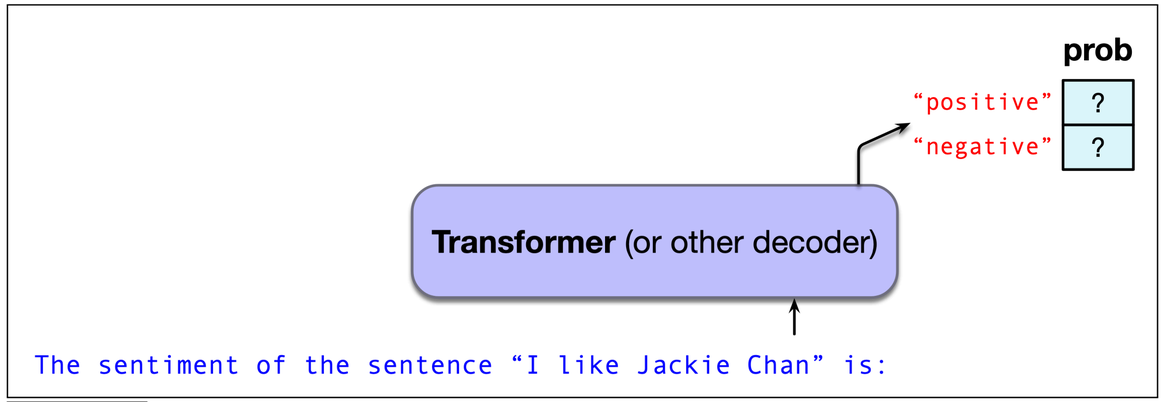


Figure 7.4 Computing the probabilities of the tokens `positive` and `negative` occurring after this prefix.

Converting Tasks to Language Modeling

- Take a supervised task and find a way to **pose it as text**
- For T5, the input is **prefixed with the task name** (e.g. `rte`)
- The task is now to **generate the tokens** of the **correct answer**
- Sometimes with **naturalistic prompting** like "the sentiment of the sentence is..."

D.2 RTE

Original input:

Sentence 1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians).

Sentence 2: Slovenia has 3,000 inhabitants.

Processed input: rte sentence1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians). sentence2: Slovenia has 3,000 inhabitants.

Original target: 1

Processed target: not_entailment

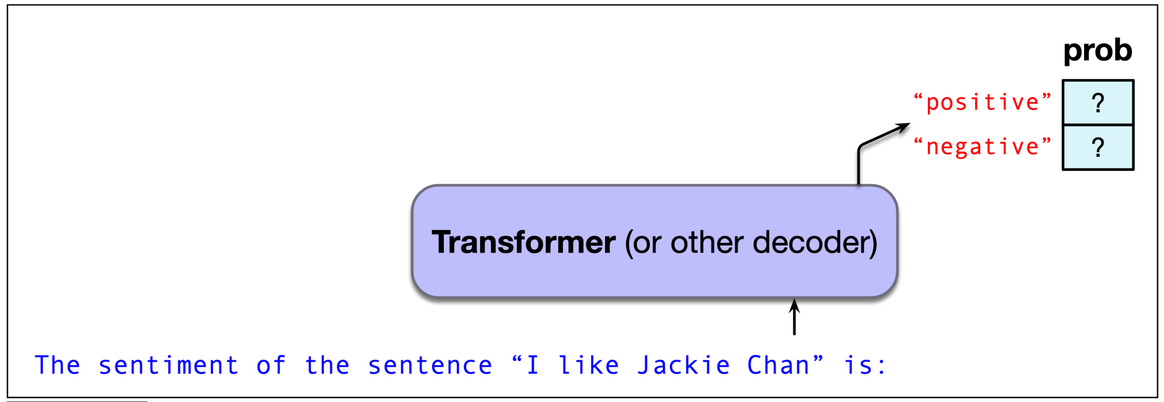


Figure 7.4 Computing the probabilities of the tokens `positive` and `negative` occurring after this prefix.

Converting Tasks to Language Modeling

- Take a supervised task and find a way to **pose it as text**
- For T5, the input is **prefixed with the task name** (e.g. `rte`)
- The task is now to **generate the tokens** of the **correct answer**
- Sometimes with **naturalistic prompting** like "the sentiment of the sentence is..."
- Other times this is with more **"meta"** text like `sentence1:` , `sentence2:`

D.2 RTE

Original input:

Sentence 1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians).

Sentence 2: Slovenia has 3,000 inhabitants.

Processed input: rte sentence1: A smaller proportion of Yugoslavia's Italians were settled in Slovenia (at the 1991 national census, some 3000 inhabitants of Slovenia declared themselves as ethnic Italians). sentence2: Slovenia has 3,000 inhabitants.

Original target: 1

Processed target: not_entailment

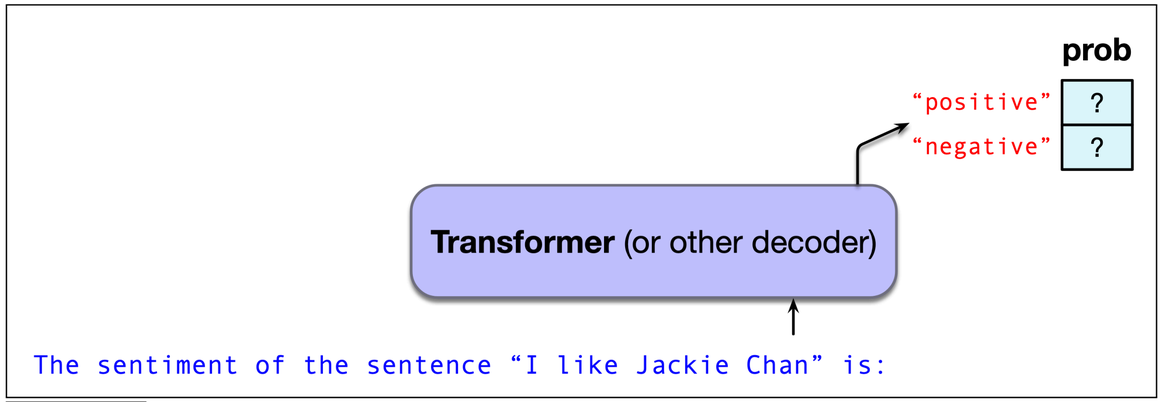
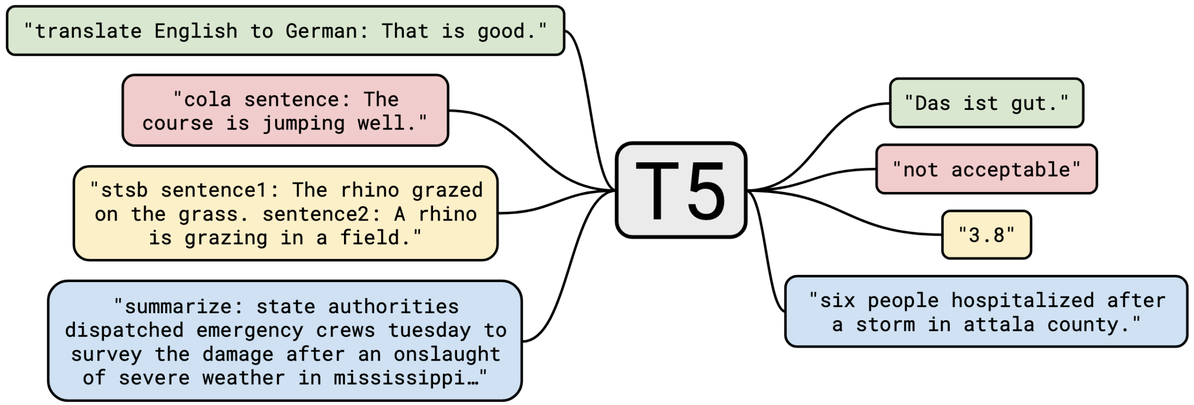


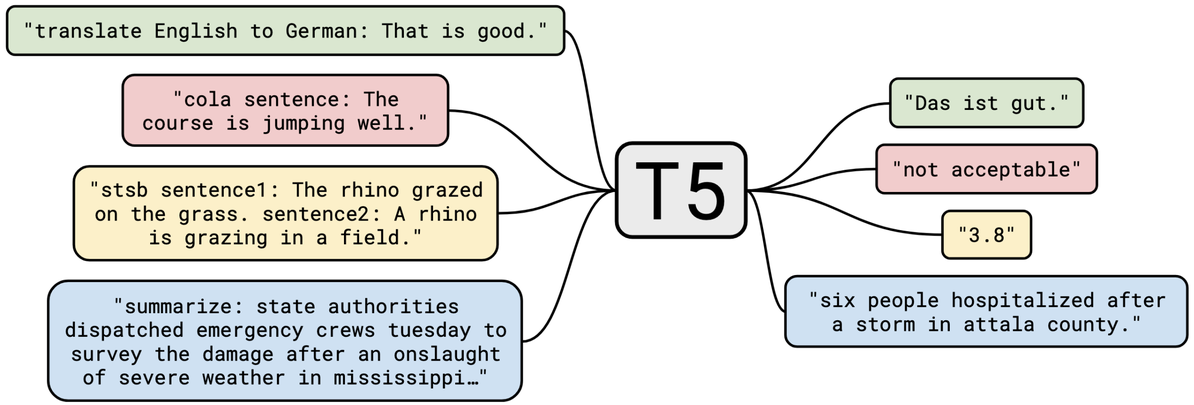
Figure 7.4 Computing the probabilities of the tokens `positive` and `negative` occurring after this prefix.

Why Text-to-Text?



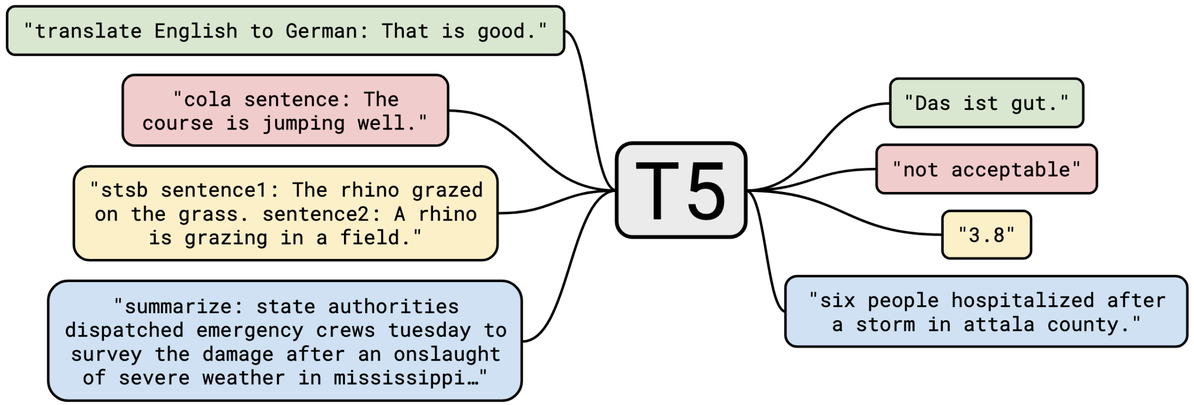
Why Text-to-Text?

- No need to train a **separate task head** for each new task



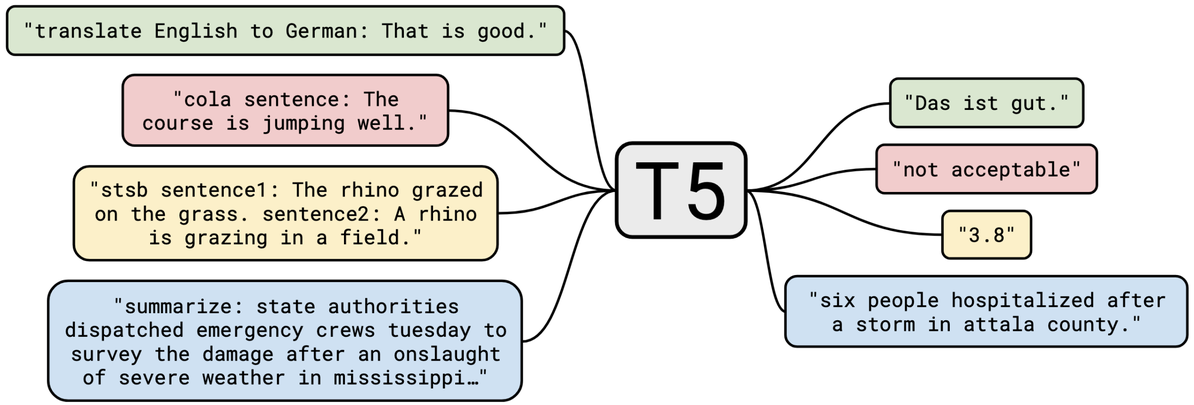
Why Text-to-Text?

- No need to train a **separate task head** for each new task
  - **No specialized architecture** at all!



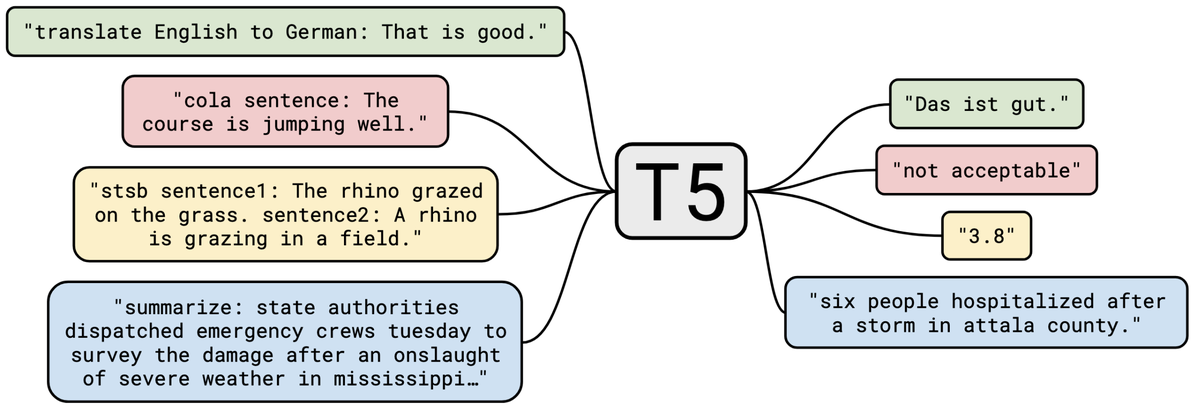
Why Text-to-Text?

- No need to train a **separate task head** for each new task
  - **No specialized architecture** at all!
- Builds on the model's **existing strength** in text prediction



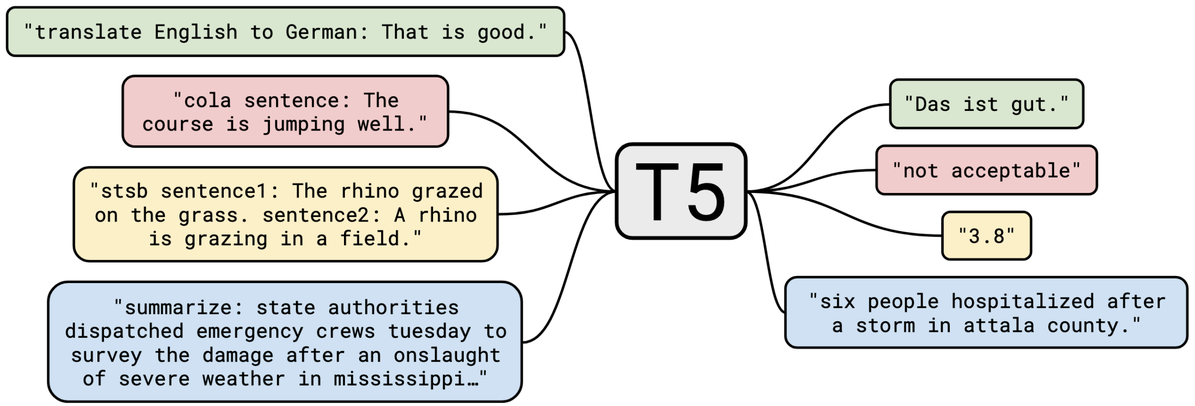
Why Text-to-Text?

- No need to train a **separate task head** for each new task
  - **No specialized architecture** at all!
- Builds on the model's **existing strength** in text prediction
- Allows models to **answer questions more naturalistically**



Why Text-to-Text?

- No need to train a **separate task head** for each new task
  - **No specialized architecture** at all!
- Builds on the model's **existing strength** in text prediction
- Allows models to **answer questions more naturalistically**
- Prompts allow for **task specification and demonstration**



Task Inventories

(a) SUP-NATINST (this work)

(b) NATINST

(c) PROMPTSOURCE (T0 subset)

(d) FLAN

(e) INSTRUCTGPT

Figure 2: Compared to other datasets, SUP-NATINST covers a more diverse range of task types. InstructGPT reports a very coarse categorization of their task types. Bubble size represents the number of tasks of each type in log scale.

Where Multi-task Transfer Fails

Where Multi-task Transfer Fails

- **Task interference:** when tasks actively degrade each other

Where Multi-task Transfer Fails

- **Task interference:** when tasks actively degrade each other
- **Task weighting** can be a poblem
  - The task weight λ_t can be **critical in practice**
  - High-resource tasks can **drown out low-resource tasks**
  - No universally-right answer for weighting

Where Multi-task Transfer Fails

- **Task interference:** when tasks actively degrade each other
- **Task weighting** can be a poblem
  - The task weight λ_t can be **critical in practice**
  - High-resource tasks can **drown out low-resource tasks**
  - No universally-right answer for weighting
- This type of interference is called **Negative Transfer**
  - Important and active field of research

How much to transfer?

Fine-tuning Spectrum

Fine-tuning Spectrum

- **Full Fine-tuning**: train **all parameters** on new task/domain
  - Provides **maximum adaptation**, but the risk of **overfitting** or **"catastrophic forgetting"**

Fine-tuning Spectrum

- **Full Fine-tuning**: train **all parameters** on new task/domain
  - Provides **maximum adaptation**, but the risk of **overfitting** or **"catastrophic forgetting"**
- **Partial Fine-tuning:** only train **some layers/modules**, rest are **frozen**
  - Intuition: **preserve general-purpose features** while adapting **task-specific parameters**

Fine-tuning Spectrum

- **Full Fine-tuning**: train **all parameters** on new task/domain
  - Provides **maximum adaptation**, but the risk of **overfitting** or **"catastrophic forgetting"**
- **Partial Fine-tuning:** only train **some layers/modules**, rest are **frozen**
  - Intuition: **preserve general-purpose features** while adapting **task-specific parameters**
- **Feature Extraction:** (aside from an optional task head) **everything is frozen**
  - Pre-trained model used as a **fixed feature extractor**
  - **Low risk** of overfitting or forgetting

"Parameter-Efficient Transfer"

"Parameter-Efficient Transfer"

- Problem with full fine-tuning: **computational scale**
  - Unless doing multi-task, requires storing **millions of parameters per task**

"Parameter-Efficient Transfer"

- Problem with full fine-tuning: **computational scale**
  - Unless doing multi-task, requires storing **millions of parameters per task**
- **Adapter modules:** small trainable layers inserted **between pre-trained layers**
  - Layers of main model **remain frozen**, only small adapters train
  - Sometimes each **task or language** gets its **own adapter**

"Parameter-Efficient Transfer"

- Problem with full fine-tuning: **computational scale**
  - Unless doing multi-task, requires storing **millions of parameters per task**
- **Adapter modules:** small trainable layers inserted **between pre-trained layers**
  - Layers of main model **remain frozen**, only small adapters train
  - Sometimes each **task or language** gets its **own adapter**
- **LoRA (Low-Rank Adaptation):** essentially, learn a **compact offset to the pre-trained weights** $W' = W + BA$
  - B and A are **low-rank matrices**, containing **strictly less information** than W

Knowledge Distillation

Knowledge Distillation

Knowledge Distillation

- All previous techniques have **transferred across data or task**

Knowledge Distillation

- All previous techniques have **transferred across data or task**
- **Knowledge Distillation:** transfer from **large to small model** (same task)

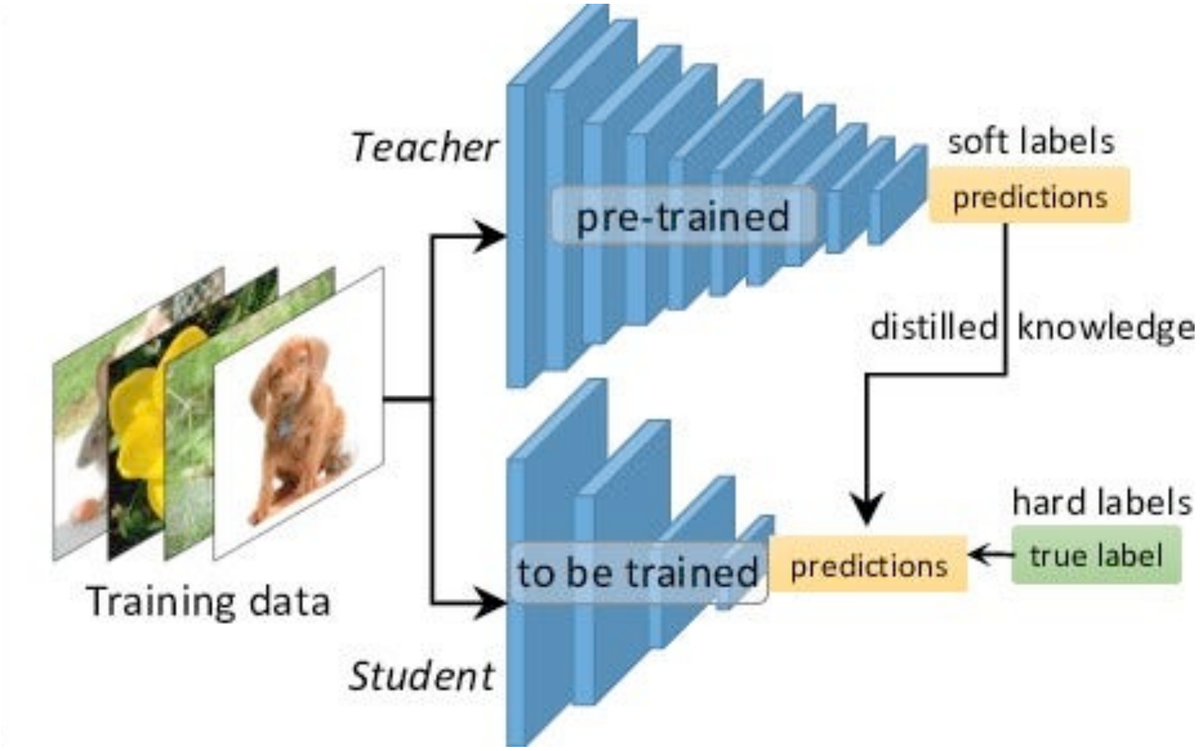
Knowledge Distillation

- All previous techniques have **transferred across data or task**
- **Knowledge Distillation:** transfer from **large to small model** (same task)
- Why not just train the small model directly?
  - Insight: the teacher model's **soft predictions / representations** contain **more information** than the original labels

Knowledge Distillation

- All previous techniques have **transferred across data or task**
- **Knowledge Distillation:** transfer from **large to small model** (same task)
- Why not just train the small model directly?
  - Insight: the teacher model's **soft predictions / representations** contain **more information** than the original labels
- $\mathscr{L}_{\text{distill}} = (1-\alpha)\cdot\mathscr{L}_{\text{CE}}(y,\hat{y}_S) + \alpha\cdot \text{KL}\left(\sigma(z_T/\tau)\|\sigma(z_S/\tau)\right)$
  - KL: **KL Divergence** between **teacher and student probability dists.**
  - L_CE: normal **classification loss**

Distillation Visualized



When use distillation?

When use distillation?

- Example: you have a **powerful model** (e.g. LLM), but you **can't afford to deploy it** (because of latency, cost, privacy, etc.)

When use distillation?

- Example: you have a **powerful model** (e.g. LLM), but you **can't afford to deploy it** (because of latency, cost, privacy, etc.)
- Distillation transfers the capability to a **smaller, deployable model**
  - For instance, many models that are **deployed to smartphones**

When use distillation?

- Example: you have a **powerful model** (e.g. LLM), but you **can't afford to deploy it** (because of latency, cost, privacy, etc.)
- Distillation transfers the capability to a **smaller, deployable model**
  - For instance, many models that are **deployed to smartphones**
- DistilBERT: 97% of BERT's performance at only **60% parameter size**

When use distillation?

- Example: you have a **powerful model** (e.g. LLM), but you **can't afford to deploy it** (because of latency, cost, privacy, etc.)
- Distillation transfers the capability to a **smaller, deployable model**
  - For instance, many models that are **deployed to smartphones**
- DistilBERT: 97% of BERT's performance at only **60% parameter size**
- Teacher predictions can be seen as **information-rich labeling functions**