Efficient Transfer Learning using Pretrained Models

Hassan Sajjad

Transfer Learning

- Learn knowledge from one setting and apply it to another setting

https://ruder.io/state-of-transfer-learning-in-nlp/

Why Transfer Learning?

- Several tasks share similar properties
    - Humans learn a language
        - Morphology, syntax, etc.

Why Transfer Learning?

- Several tasks share similar properties
    - Humans learn a language
        - Morphology, syntax, etc.
    - Machines can also benefit from the related tasks
        - POS tagging and named entity tagging
        - A general task helps to learn a specific task

Why Transfer Learning?

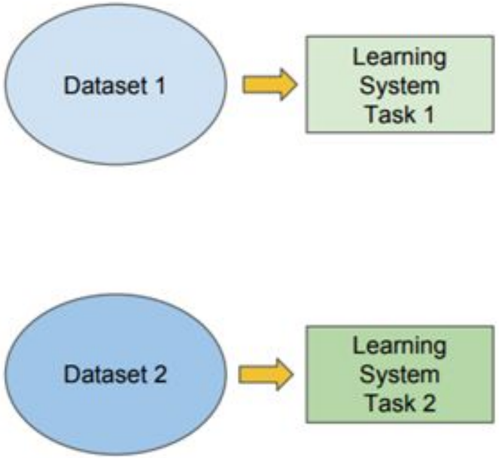
- Several tasks share similar properties
  - Humans learn a language
    - Morphology, syntax, etc.
  - Machines can also benefit from the related tasks
    - POS tagging and named entity tagging
    - A general task helps to learn a specific task

Practically,

- Labeled resources are not enough
- Generalization
  - Optimizing more than one tasks

Traditional ML

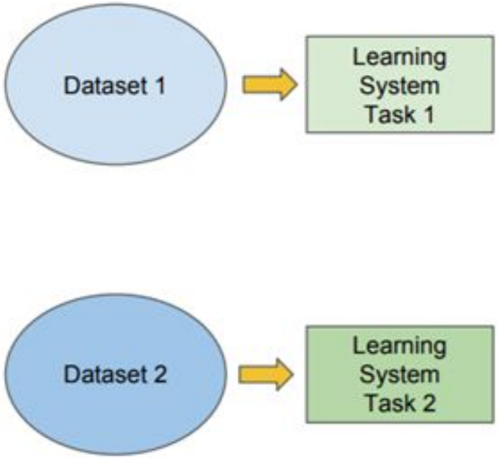
- Isolated, single task learning:
    - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



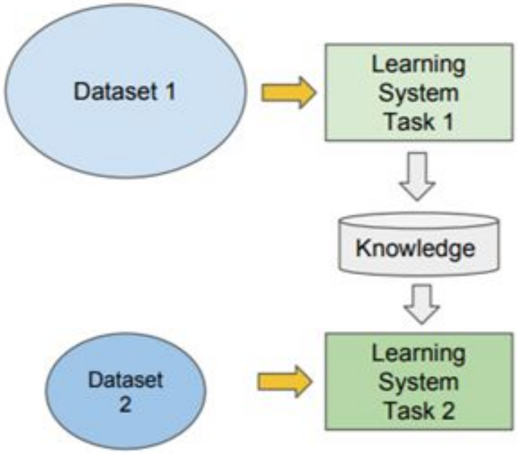
https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

Traditional ML vs Transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks

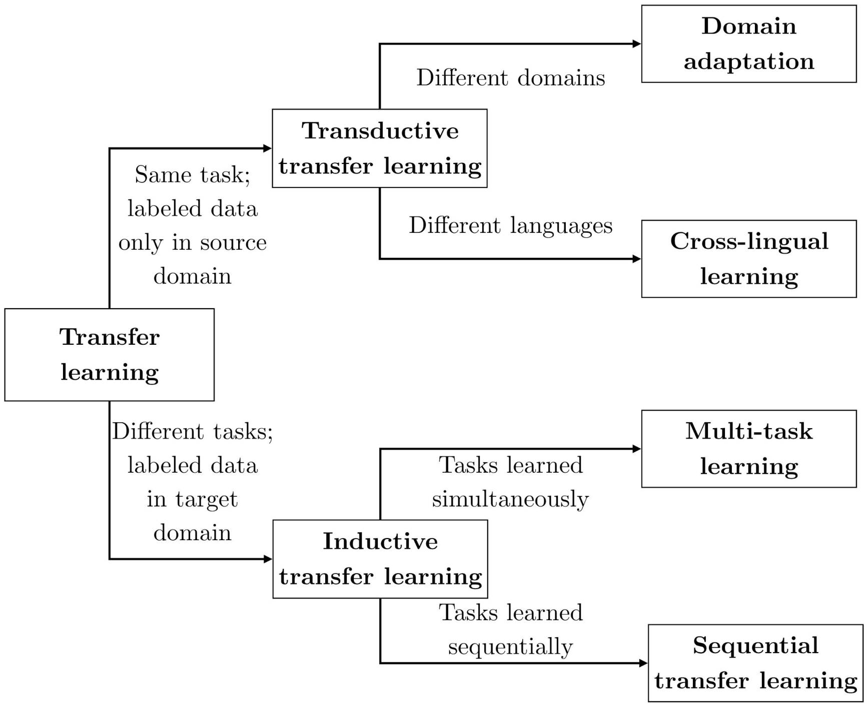


- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

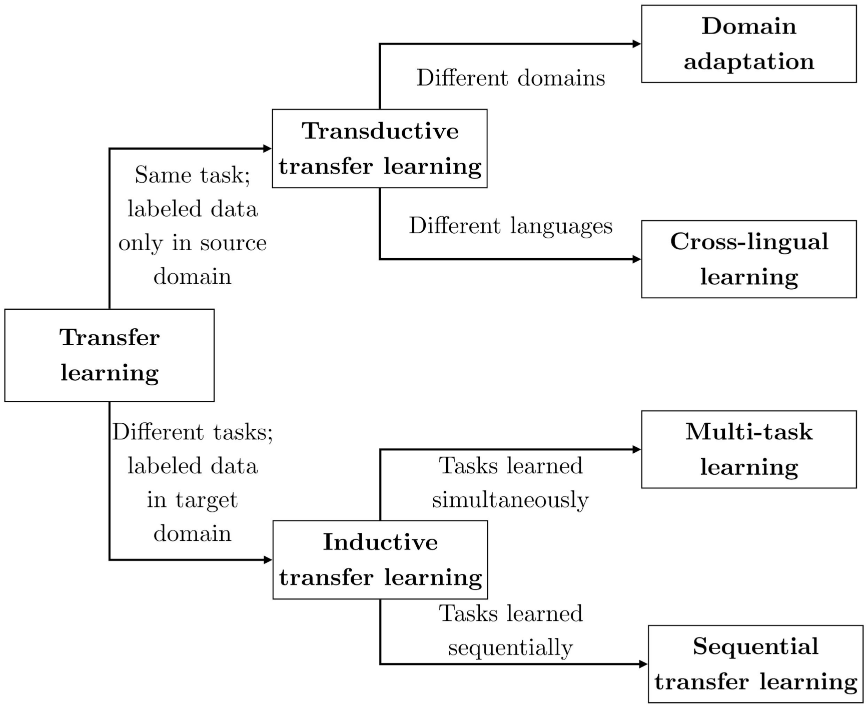
Types of Transfer Learning



https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf#page=64

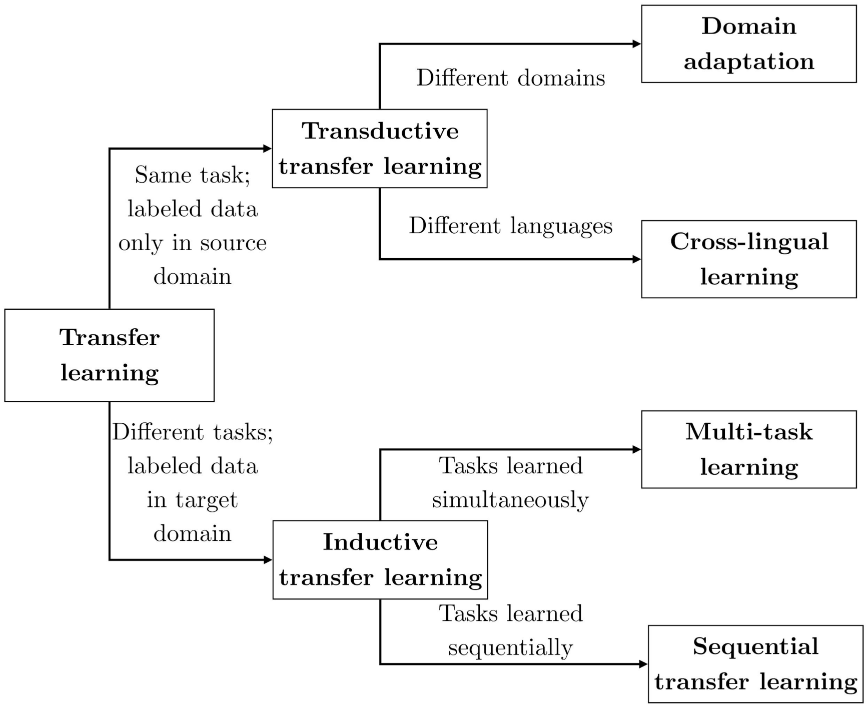
Types of Transfer Learning

Sentiment classification using News data while target domain is tweet



https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf#page=64

Types of Transfer Learning

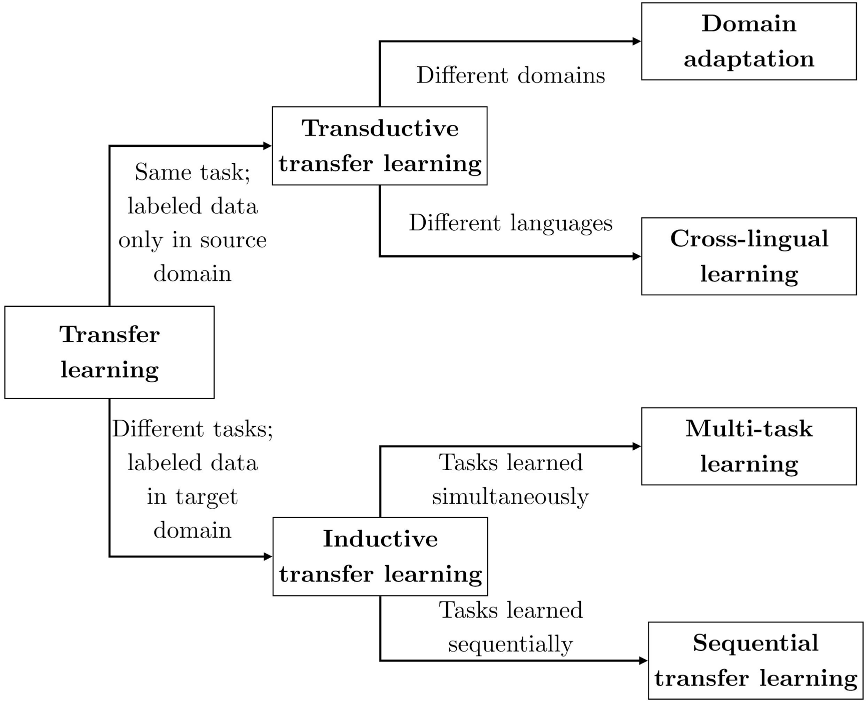


Sentiment classification using News data while target domain is tweet

Sentiment classification using English News data while target language is Spanish

https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf#page=64

Types of Transfer Learning



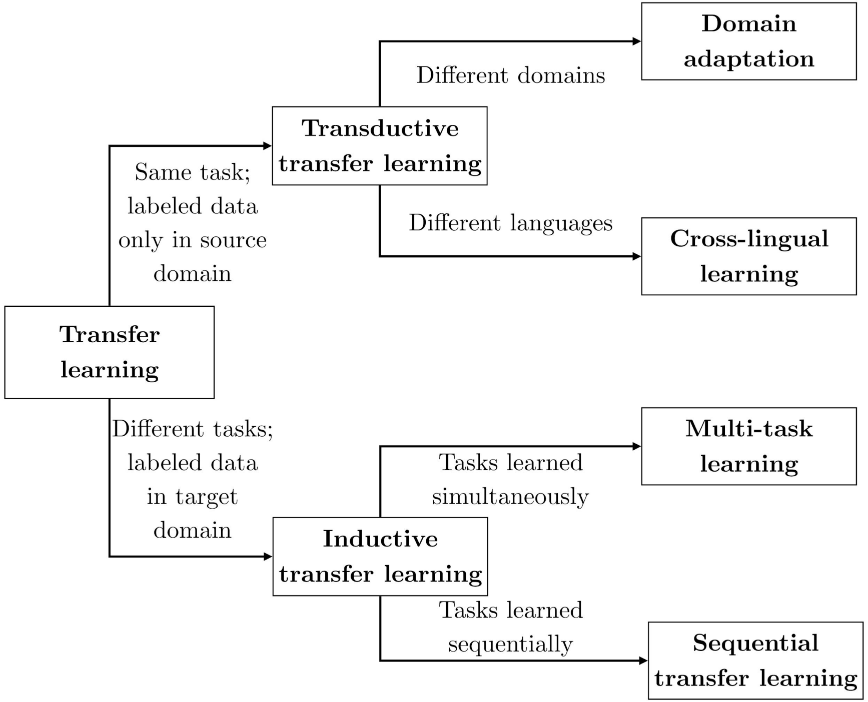
Sentiment classification using News data while target domain is tweet

Sentiment classification using English News data while target language is Spanish

POS and named-entity are related tasks that can help each other

https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf#page=64

Types of Transfer Learning



https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf#page=64

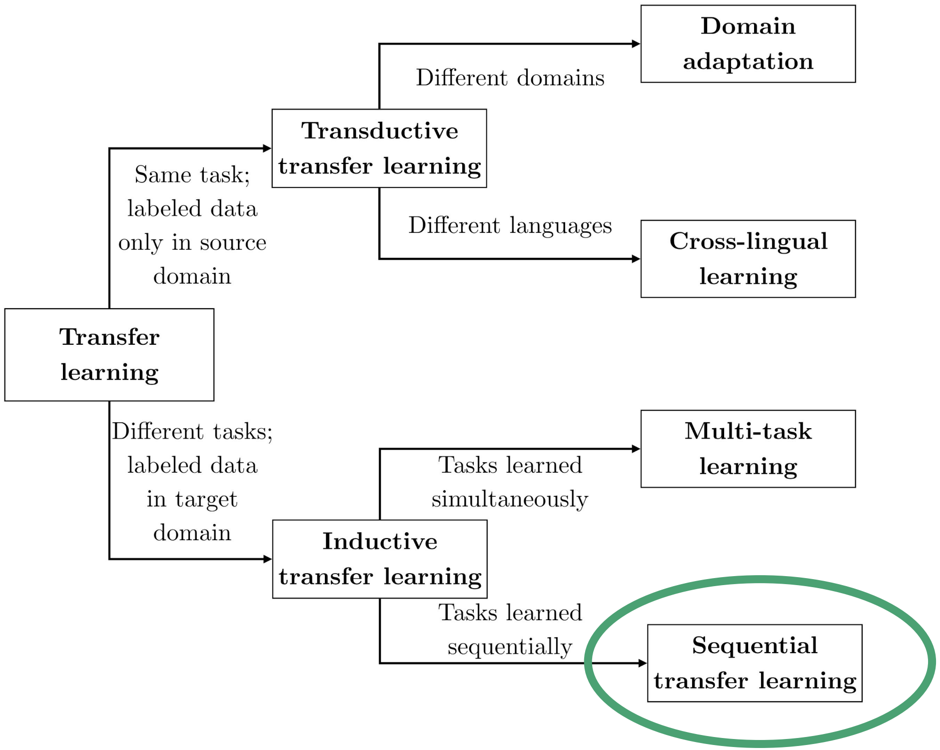
Sentiment classification using News data while target domain is tweet

Sentiment classification using English News data while target language is Spanish

POS and named-entity are related tasks that can help each other

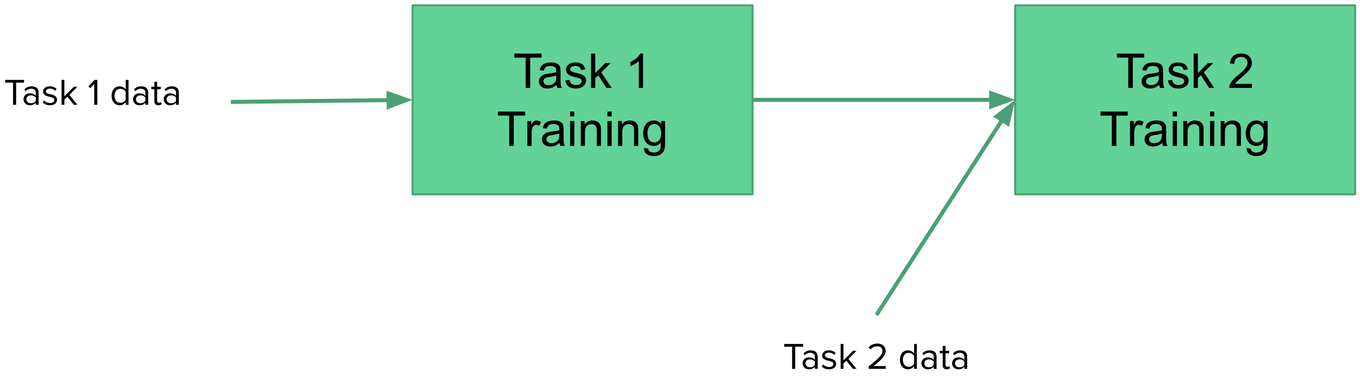
Learn task 1 first and then use the information in task 2

Types of Transfer Learning

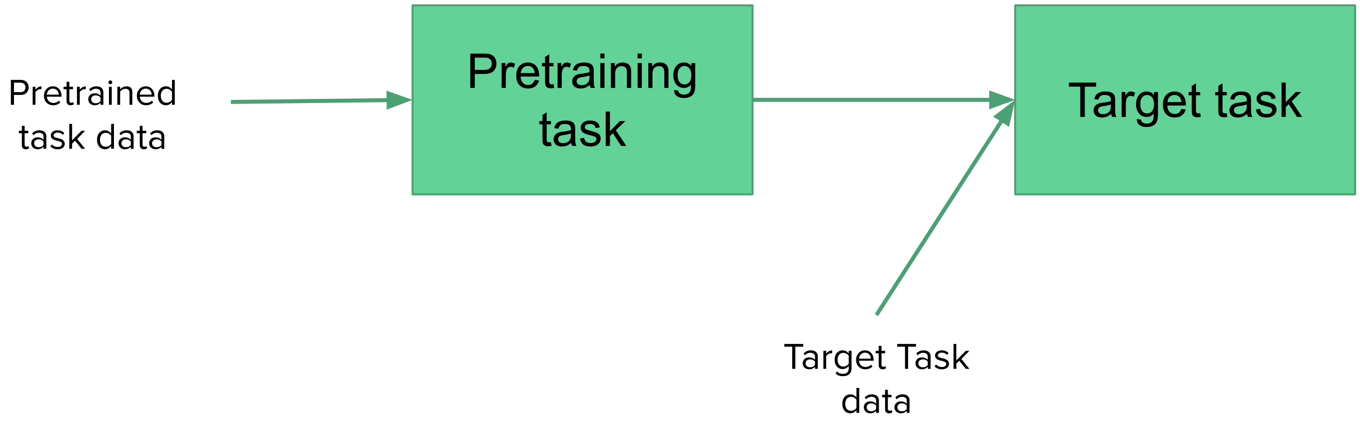


https://ruder.io/thesis/neural_transfer_learning_for_nlp.pdf#page=64

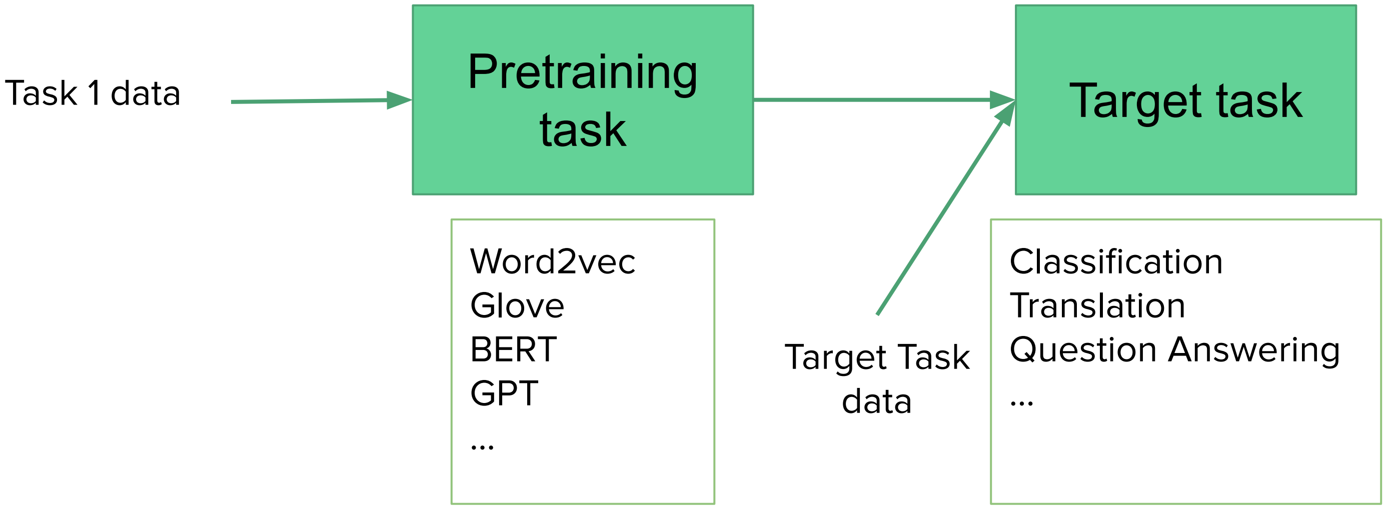
Sequential Transfer Learning



Sequential Transfer Learning



Sequential Transfer Learning



Pretraining Tasks vs. Target Tasks

Pretraining tasks

- General tasks or related to target tasks
- Unsupervised learning
- Large amount of available data
- Example: language modeling

Target Tasks

- Classification (sentiment classification)
- Word-level prediction (POS tagging)
- Generation (machine translation)

Transfer Learning Using Pretrained Model

Pretrained language models

- BERT, GPT, Elmo, XLNet
- Contextualized embeddings

Achieved state-of-the-art performance

- Sentiment analysis tasks
- Natural language inference tasks
- Text entailment tasks
- ...

Contextualized Embeddings

Word vectors

Cat = [0.8,-0.3,...]

We have two **cats**

Dog = [0.1,0.6,...]

It's raining **cats** and **dogs**

Embedding is independent of the context a word appears

Contextualized Embeddings

Word vectors

Cat = [0.8,-0.3,...]

Dog = [0.1,0.6,...]

We have two **cats**

It's raining **cats** and **dogs**

Embedding is independent of the context a word appears

Contextualized Word vectors

Cat = [0.8,-0.3,...]

Dog = [0.1,0.6,...]

We have two **cats**

It's raining **cats** and **dogs**

Cat = [0.5,0.1,...]

Dog = [-0.9,0.2,...]

Contextualized Embeddings

Word vectors

Cat = [0.8,-0.3,...]

Dog = [0.1,0.6,...]

We have two **cats**

It's raining **cats** and **dogs**

Embedding is independent of the context a word appears

Contextualized Word vectors

Cat = [0.8,-0.3,...]

Dog = [0.1,0.6,...]

We have two **cats**

Different vectors of "Cat" based on the context

It's raining **cats** and **dogs**

Cat = [0.5,0.1,...]

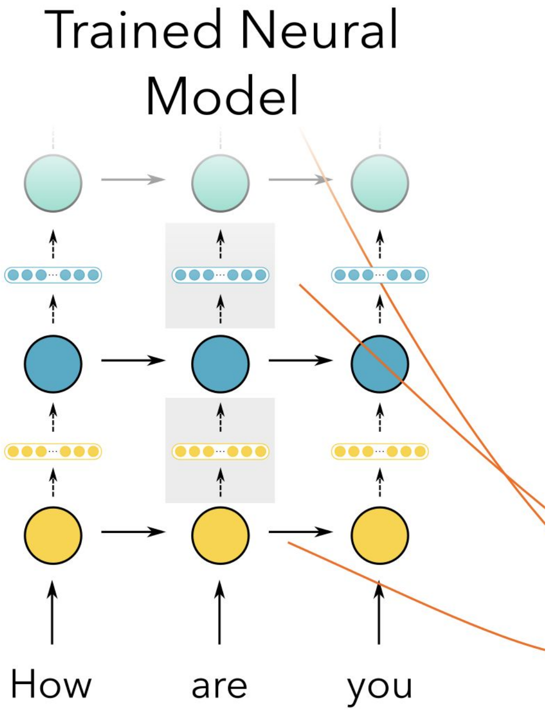
Dog = [-0.9,0.2,...]

Sequential Transfer Learning

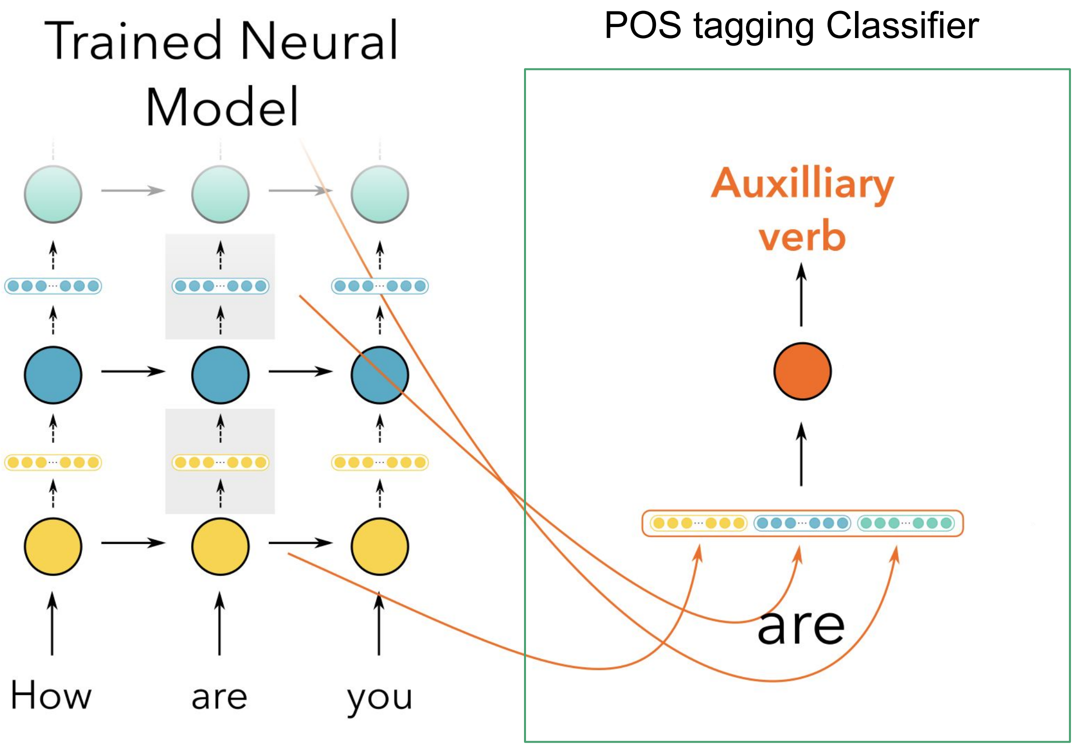
Given a pretrained model, there are two common ways to apply transfer learning

- Feature-based transfer learning
- Fine-tuning based transfer learning

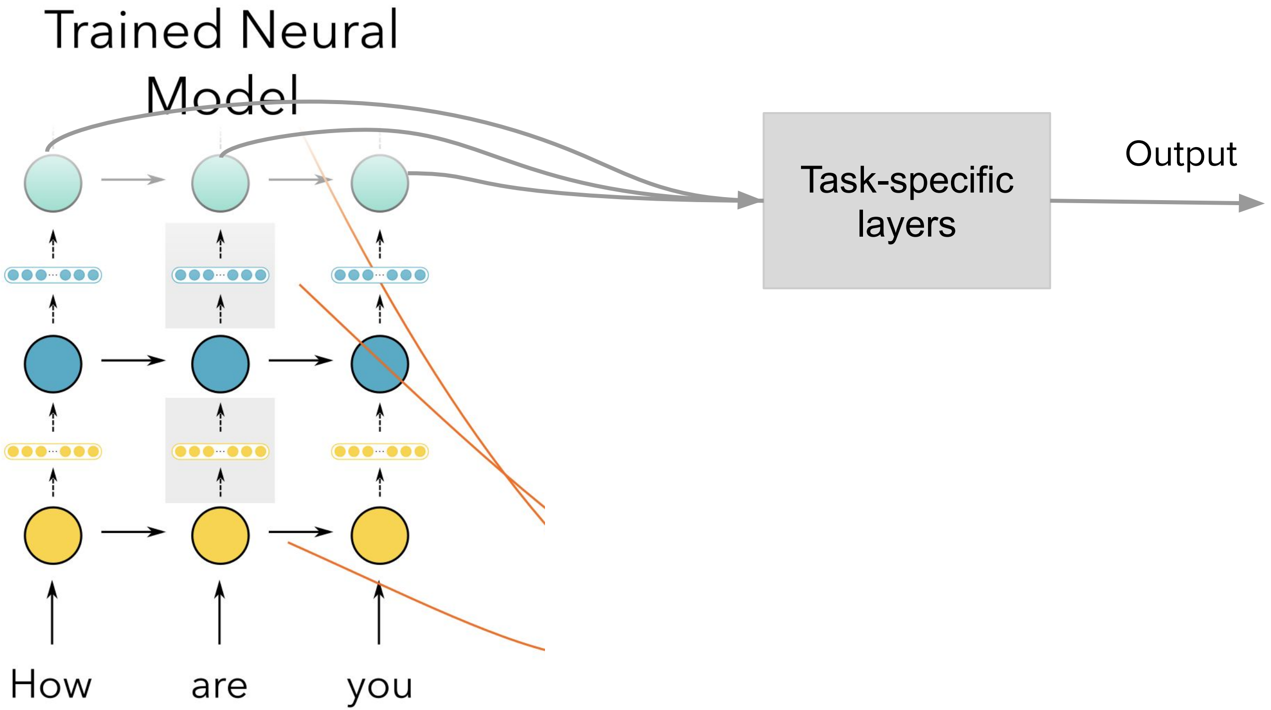
Feature-based Transfer Learning



Feature-based Transfer Learning



Fine-tuning based Transfer Learning



Sequential Transfer Learning

Feature based

- Choice of the classification model
- More control
  - Speed up
  - Memory requirement

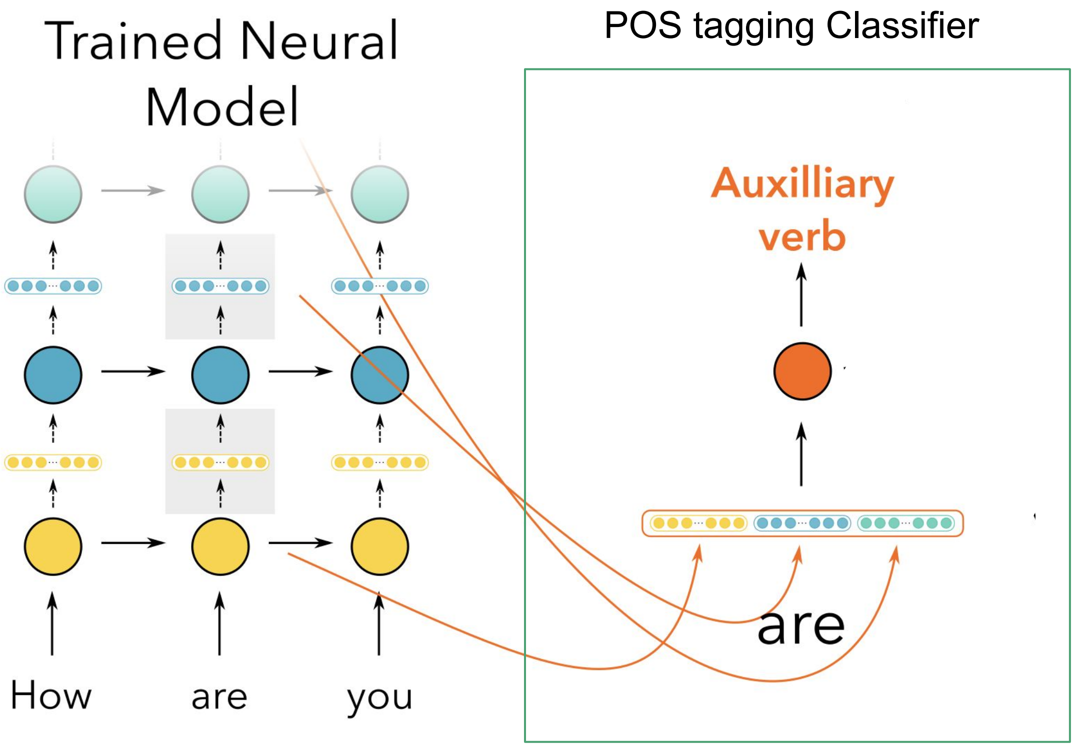
Fine-tuning based

- End-to-end learning
- Better in performance on some tasks

Efficient Feature-based Transfer Learning

Efficient Feature-based Transfer Learning

- Take a pretrained model
- Choose a target task
- For every input in the target task, extract contextualized embeddings
- Use them as feature in the task-specific classifier



Efficient Feature-based Transfer Learning

Pretrained models

- BERT-base
  - 13 layers including embedding layers
  - Layer size: 768
  - Features for transfer learning: 13 x 768 = 9984
- BERT-large
  - 25 layers including embedding layers
  - Layer size: 1024
  - Features for transfer learning: 25 * 1024 = 25600

Efficient Feature-based Transfer Learning

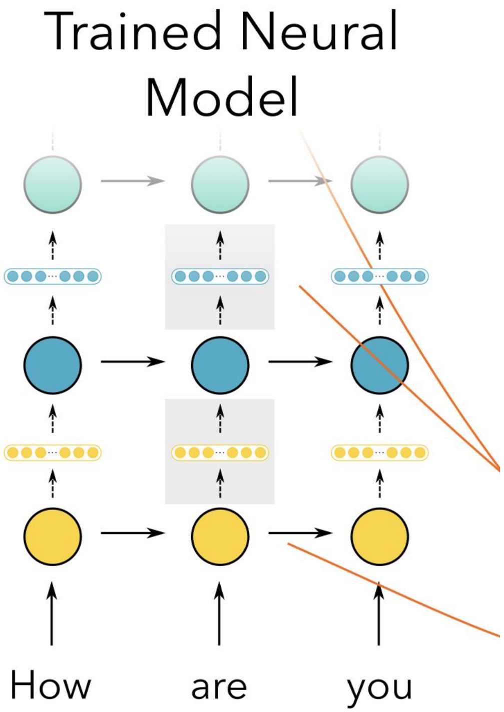
- BERT-base
  - 13 layers including embedding layers
  - 768 each layer size
  - Features for transfer learning: 13 x 768 = **9984**
- BERT-large
  - 25 layers including embedding layers
  - 1024 layer size
  - Features for transfer learning: 25 * 1024 = **25600**

Problem

- A large number of feature (compared to 300 embedding size of word2vec)
- Slow training and inference time
- Model overfits

Efficiency Bottlenecks

- Contextualized embedding extraction
  - Requires a full forward pass of the pretrained model
- Large number of features
  - Slower training
  - Slower inference time
  - Sub-optimum performance



Efficient Feature-based Transfer Learning

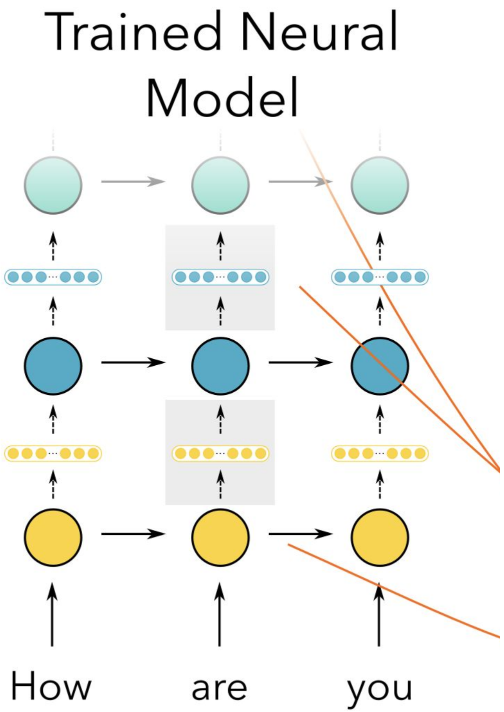
Hypothesis

1. The distributive nature of the pretrained models causes information **redundancy at both layer level and feature level**
2. Since pretrained models can be used as universal feature extractors, **not all features are equally relevant** for a downstream task

Hypothesis 1

Questions

- Is it necessary to extract contextualized embeddings from all layers of the network?
  - If knowledge about a task is mostly learned up to certain layers, we can limit the forward pass up to those layers



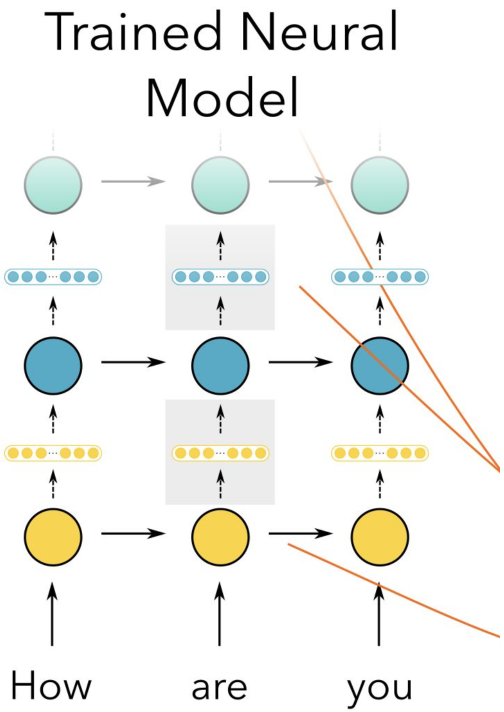
Hypothesis 1

Questions

- Is it necessary to extract contextualized embeddings from all layers of the network?
    - If knowledge about a task is mostly learned up to certain layers, we can limit the forward pass up to those layers

LayerSelector Module

- It identifies the optimal number of layers required for a downstream task, i.e., reducing the size of contextualized embeddings



Hypothesis 1 & 2

Questions

- Do we need all the features for a downstream NLP task?
  - Due to distributive nature, the information among features may be redundant
  - Not all the features might be relevant equally important for a particular task

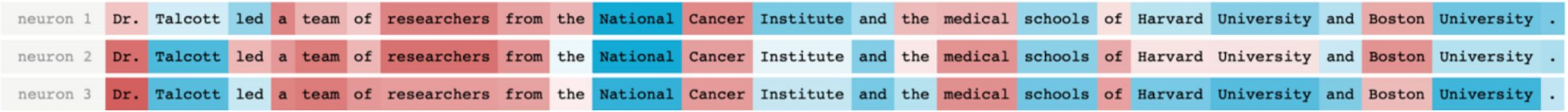
Hypothesis 1 & 2

Questions

- Do we need all the features for a downstream NLP task?
  - Due to distributive nature, the information among features may be redundant
  - Not all the features might be relevant equally important for a particular task

CCFS -- Multivariate feature ranking

- Correlation clustering
  - Identify redundant features (neurons)
  - Task independent



Hypothesis 1 & 2

Questions

- Do we need all the features for a downstream NLP task?
  - Due to distributive nature, the information among features may be redundant
  - Not all the features might be relevant equally important for a particular task

CCFS -- Multivariate feature ranking

- Correlation clustering
  - Identify redundant features (neurons)
  - Task independent
- Multivariate feature ranking
  - Elastic-net based feature ranking
  - Identify relevant features with respect to a task

$$\mathcal{L}(\theta) = -\sum_i \log P_\theta(\mathbf{l}_i|x_i) + \lambda_1\|\theta\|_1 + \lambda_2\|\theta\|_2^2$$

Overall Procedure

1. Use LayerSelector to select optimal number of layers L
2. Extract contextualized embedding up to the layer L
3. Run correlation clustering to identify and filter out redundant features
4. Run task-specific feature selector on the reduced feature set
5. Select the top features with respect to the task

Goal - reduce the feature set while maintaining performance within a threshold

Evaluation

- BERT and XLNet pretrained models
- Sequence labeling tasks
  - POS tagging, Semantic tagging, CCG tagging, Chunking, Named-entity tagging
- Sequence classification tasks (GLUE benchmarks)
  - Sentiment analysis
  - Semantic equivalence classification
  - Semantic textual similarity
  - Natural language inference (MNLI, QNLI)
  - Question pairs similarity
  - Textual entailment

Results - Sequence labeling

		POS	SEM	CCG	Chunking	NER
	Oracle Features	95.2%	92.0%	90.0% 9984	94.6%	97.3%

Results - Sequence labeling

		POS	SEM	CCG	Chunking	NER
	Oracle	95.2%	92.0%	90.0%	94.6%	97.3%
	Features			9984		
BERT	LS	94.8%	91.2%	88.7%	93.5%	95.4%
	Layer#	2	2	6	4	2

Results - Sequence labeling

		POS	SEM	CCG	Chunking	NER
BERT	Oracle	95.2%	92.0%	90.0%	94.6%	97.3%
	Features			9984		
	LS	94.8%	91.2%	88.7%	93.5%	95.4%
	Layer#	2	2	6	4	2
	`CCFS`	94.0%	90.1%	89.8%	92.3%	95.5%
	Features	300	400	400	600	300
	% Reduct.	97%↓	96%↓	96%↓	94%↓	97%↓

Results - Sequence Classification

- Using all features

	SST-2	MRPC	MNLI	QNLI	QQP	RTE	STS-B
Oracle Features	90.6%	86.0%	81.7%	90.2% 9984	91.2%	69.3%	89.7%

Results - Sequence Classification

- Using LayerSelector

		SST-2	MRPC	MNLI	QNLI	QQP	RTE	STS-B
BERT	Oracle Features	90.6%	86.0%	81.7%	90.2% 9984	91.2%	69.3%	89.7%
	LS Layer#	85.6% 6	86.0% 11	81.6% 11	89.9% 11	90.9% 11	69.3% 12	89.1% 11

Results - Sequence Classification

- As few as 10 features vs. 9984 features

		SST-2	MRPC	MNLI	QNLI	QQP	RTE	STS-B
BERT	Oracle Features	90.6%	86.0%	81.7%	90.2% 9984	91.2%	69.3%	89.7%
	LS Layer#	85.6% 6	86.0% 11	81.6% 11	89.9% 11	90.9% 11	69.3% 12	89.1% 11
	CCFS Features % Reduction	86.1% 100 (99%↓)	85.5% 100 (99%↓)	81.0% 20 (99.8%↓)	89.2% 10 (99.9%↓)	90.2% 30 (99.7%↓)	69.0% 30 (99.7%↓)	88.5% 400 (96%↓)

Summary

- Transfer learning enables use of various tasks for the benefit of a target task
  - Labeled data is always limited
  - Models trained for a particular tasks do not generalize well
- Sequential transfer learning consists of a pretrained model and a task-specific model
  - Pretrained models are mainly self-learning modules
- The state-of-the-art models come with the issue of requiring large memory and high inference time
- Our proposed method reduces the feature set size to as low as 1% while maintaining 97% of the original performance

Thank you