AFRY

ÅF PÖYRY

KNOWLEDGE DISTILLATION

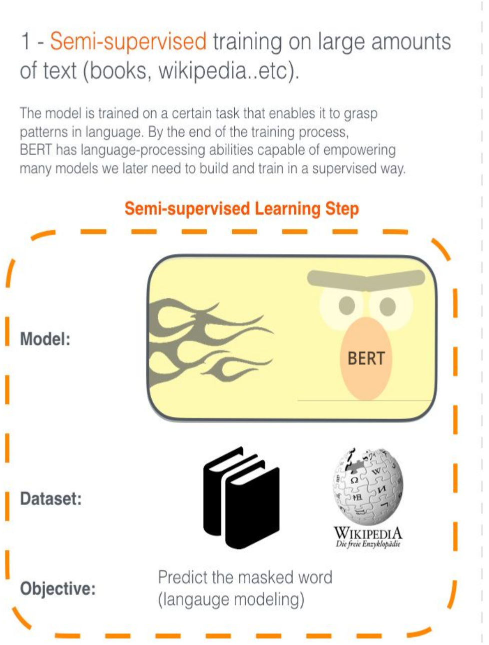
Strike balance between efficiency & performance

Hampus Londögård

TABLE OF CONTENTS

01 TRANSFORMER

What are they?


1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

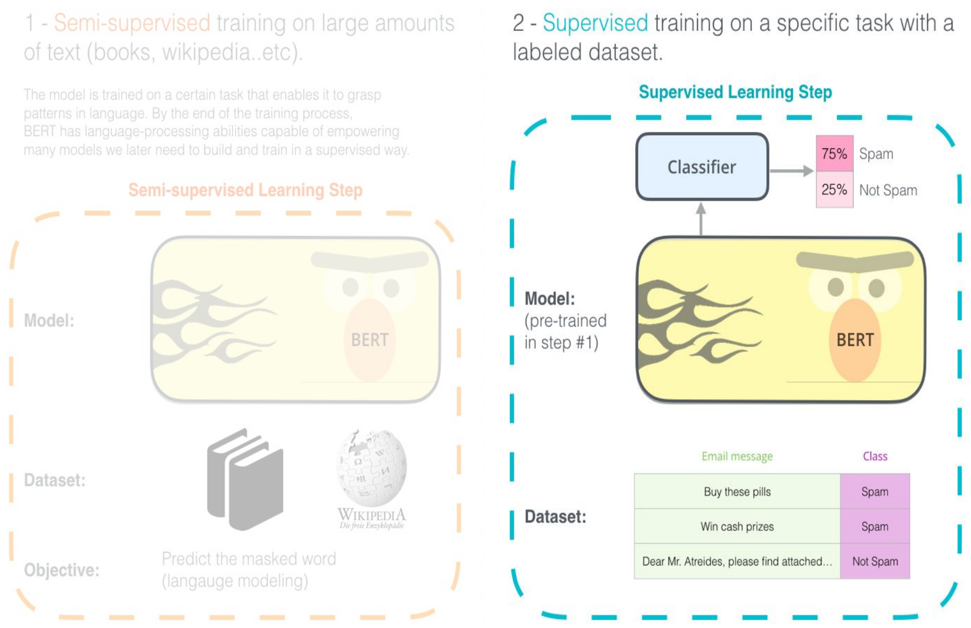
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

Semi-supervised Learning Step
Model:
BERT
Dataset:
WIKIPEDIA
Die freie Enzyklopädie
Objective:
Predict the masked word (langauge modeling)

TRANSFORMERS

- **Large Language Models**
  - Trained on massive data

blog.londogard.com/transformers-explained



1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

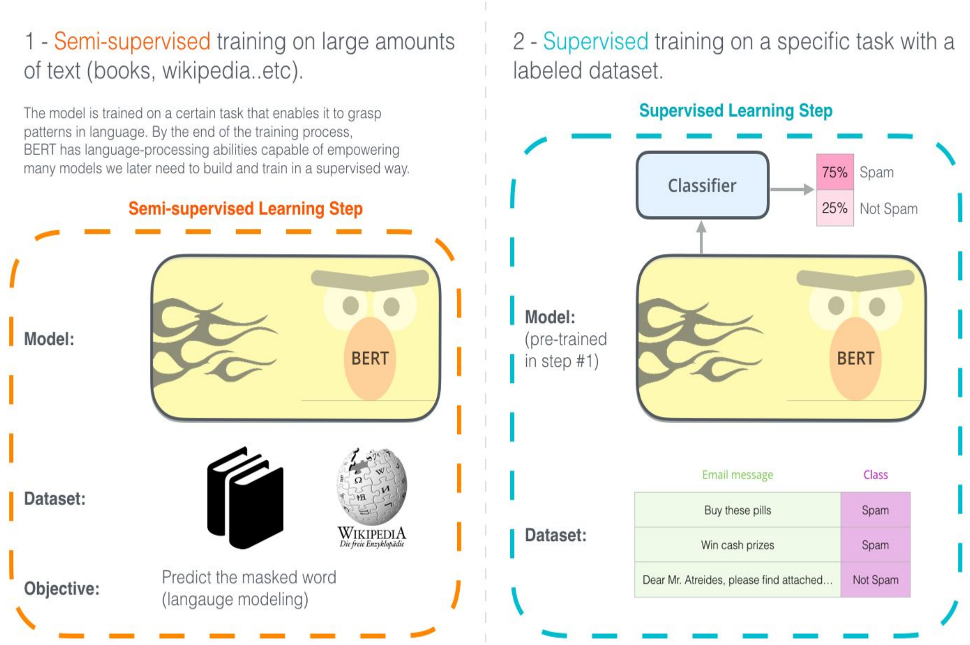
2 - Supervised training on a specific task with a labeled dataset.

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached…	Not Spam

TRANSFORMERS

- **Large Language Models**
    - Trained on massive data
- **Easily fine-tuned**
    - Little data enough

blog.londogard.com/transformers-explained


1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.
Semi-supervised Learning Step
Model:
BERT
Dataset:
Wikipedia
Die freie Enzyklopädie
Objective:
Predict the masked word (langauge modeling)
2 - Supervised training on a specific task with a labeled dataset.
Supervised Learning Step
Classifier
75% Spam
25% Not Spam
Model: (pre-trained in step #1)
BERT
Dataset:

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached…	Not Spam

TRANSFORMERS

- **Large Language Models**
  - Trained on massive data
- **Easily fine-tuned**
  - Little data enough
- **Very powerful**
  - >90 % of all State-of-the-Art (SotA) today in Text

blog.londogard.com/transformers-explained

02 EFFICIENCY

LARGE

GPU usually used for *inference*

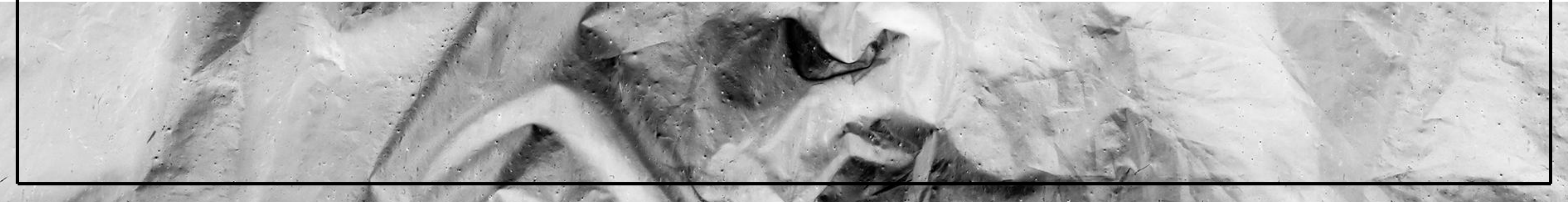
GROWING

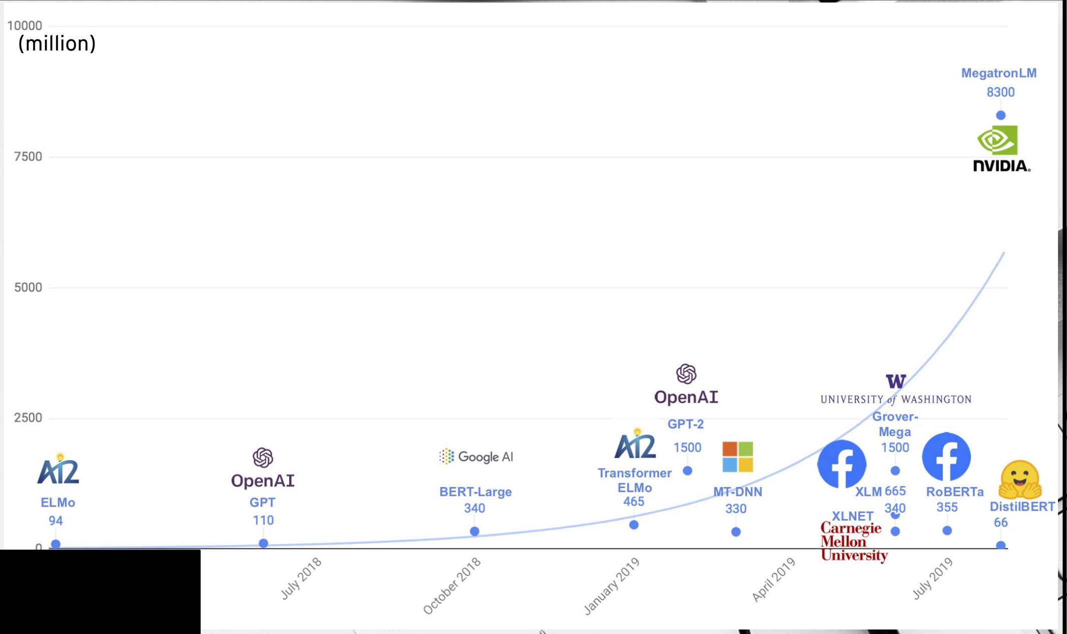
Scales incredibly well with data & size

8800%

PARAMETER INCREASE

(2018-2019)





“This is stupid and wonderful.”

—Hampus Londögård

03

QUANTIZATION

REDUCE PRECISION

To reduce the total layer size.

03 QUANTIZATION

QUANTIZATION

f32 → int8
~ 1/4th size
~ 4x faster
< 0.5% performance loss*
(* most of transformers)

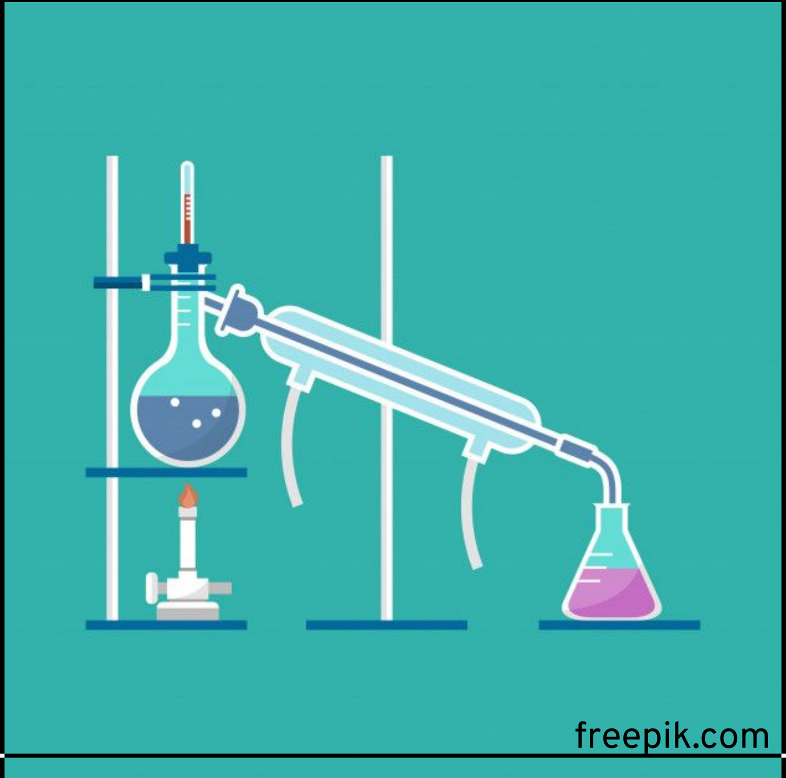
REDUCE PRECISION

To reduce the total layer size.

04 DISTILLATION

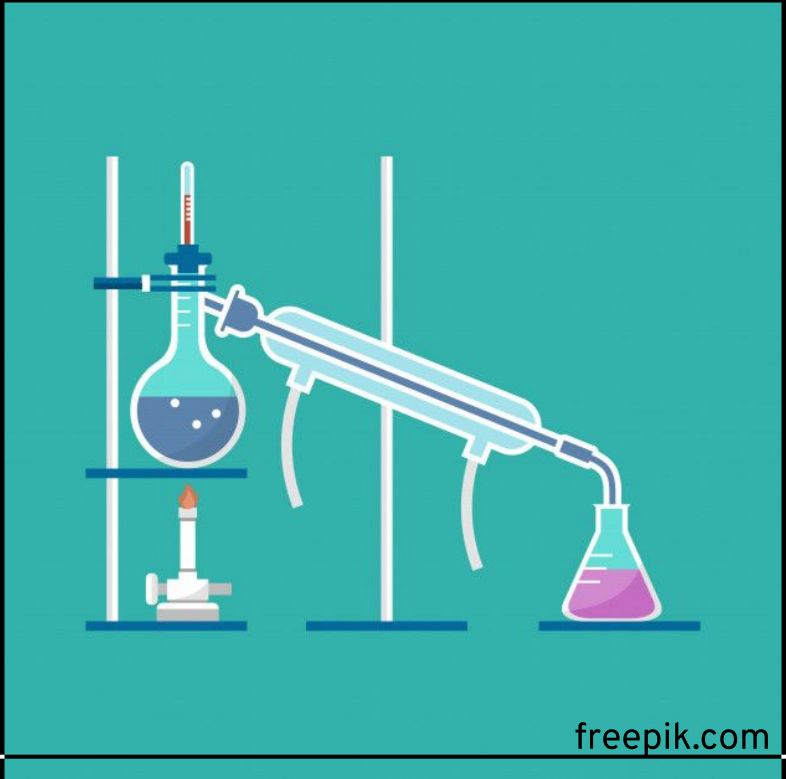
How to distill knowledge

KNOWLEDGE DISTILLATION



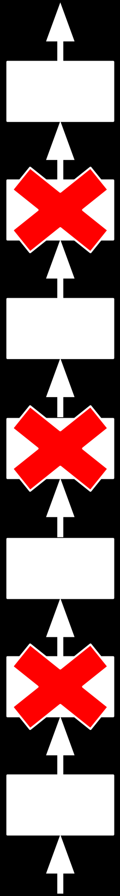
"Distilling the knowledge in a neural network"
Hinton, Geoffrey et. al 2015

KNOWLEDGE DISTILLATION



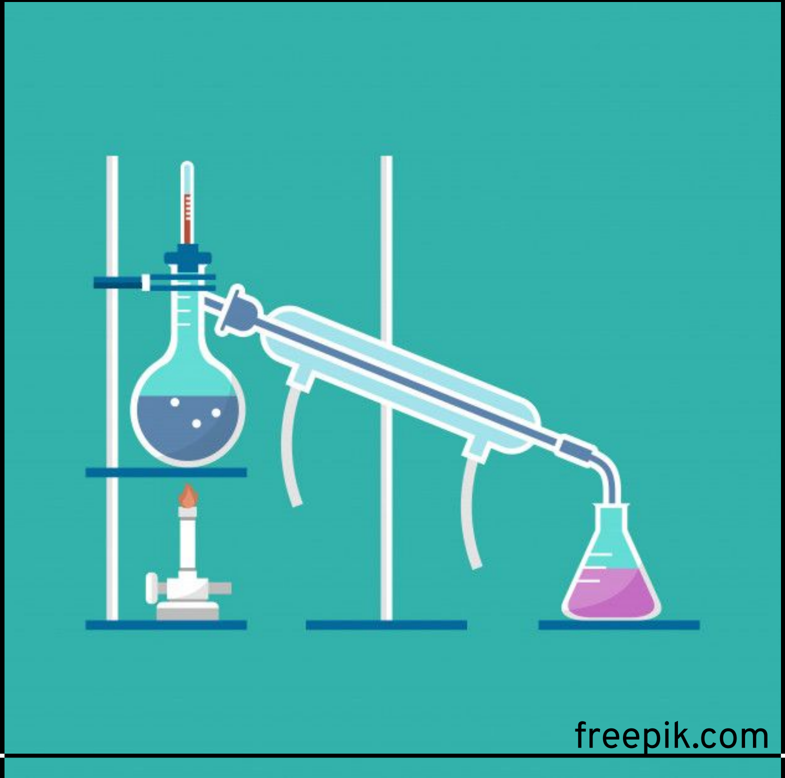
MODEL COMPRESSION

Compressing a model
E.g. 12 to 6 layers



"Distilling the knowledge in a neural network"
Hinton, Geoffrey et. al 2015

KNOWLEDGE DISTILLATION



MODEL COMPRESSION

Compressing a model
E.g. 12 to 6 layers

MODEL CHANGE

Transformer → RNN
(< 1/10th of size)

"Distilling the knowledge in a neural network"
Hinton, Geoffrey et. al 2015

PREPARATION

TEACHER: TRANSFORMER

Pre-trained on a large dataset

Available freely on huggingface.co 🤗

PREPARATION

TEACHER: TRANSFORMER

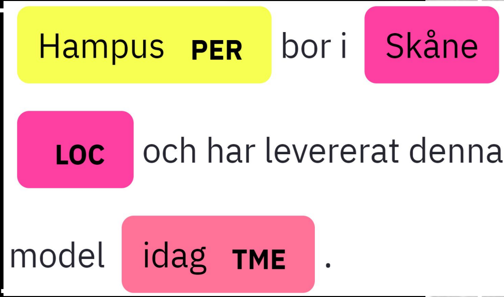
Pre-trained on a large dataset

Available freely on huggingface.co 🤗

TARGET TASK

Clear task

Dataset ready



Hampus PER bor i Skåne LOC och har levererat denna model idag TME .

PREPARATION

TEACHER: TRANSFORMER

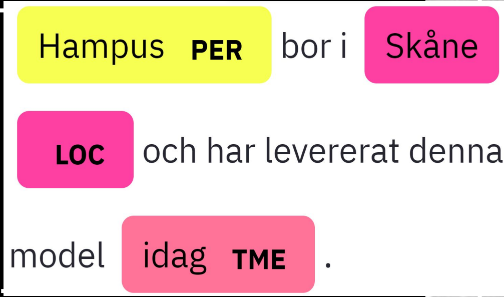
Pre-trained on a large dataset

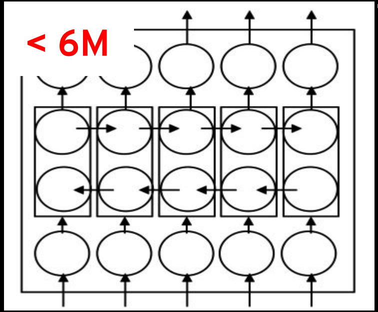
Available freely on huggingface.co 🤗

TARGET TASK

Clear task

Dataset ready



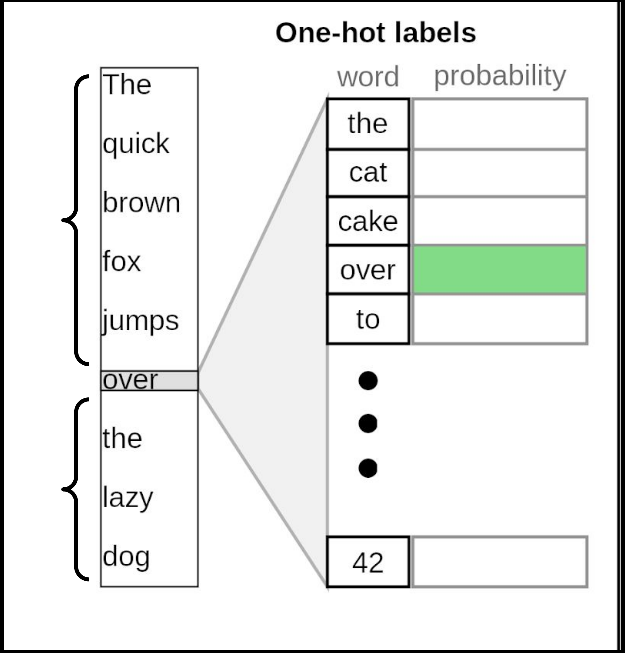


STUDENT ARCHITECTURE

What your end-goal is

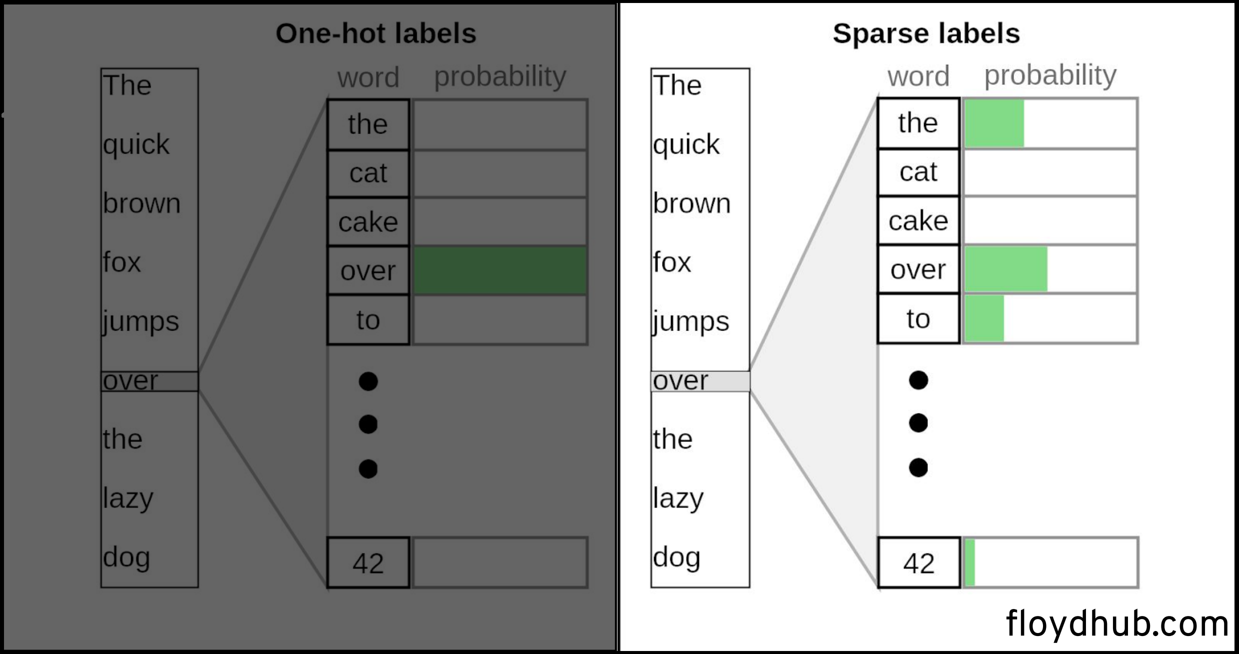
LABELS

Predict based on context



SOFT LABELS

Sometimes 'Sparse Labels'

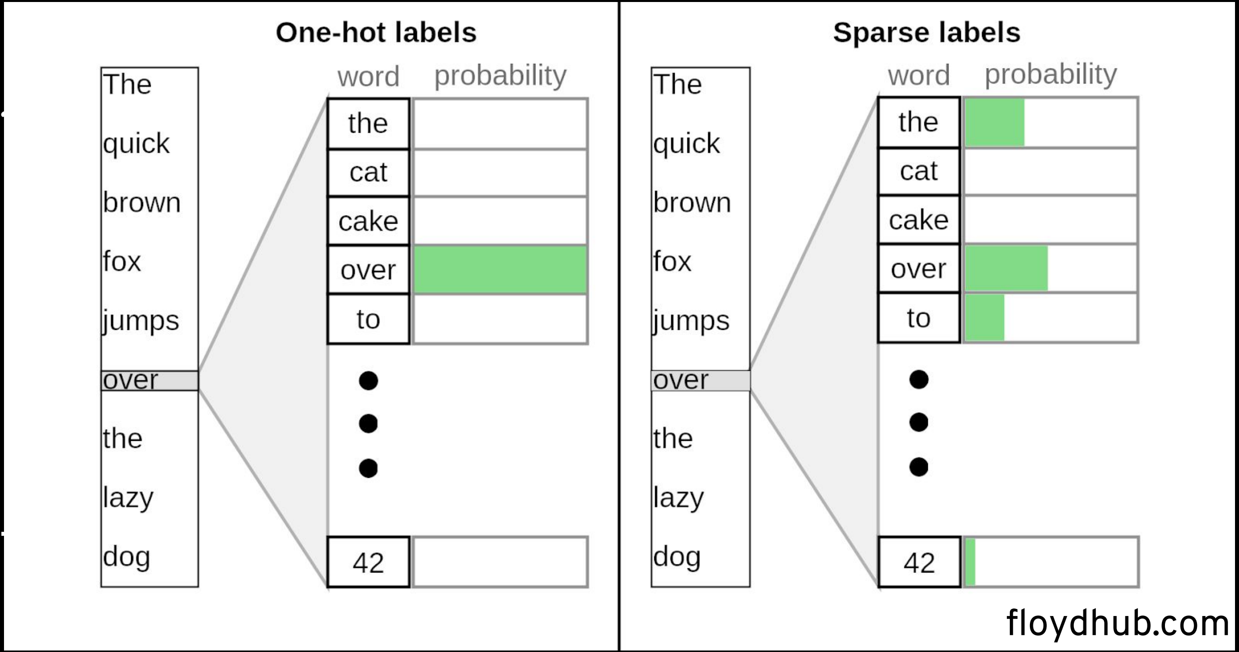


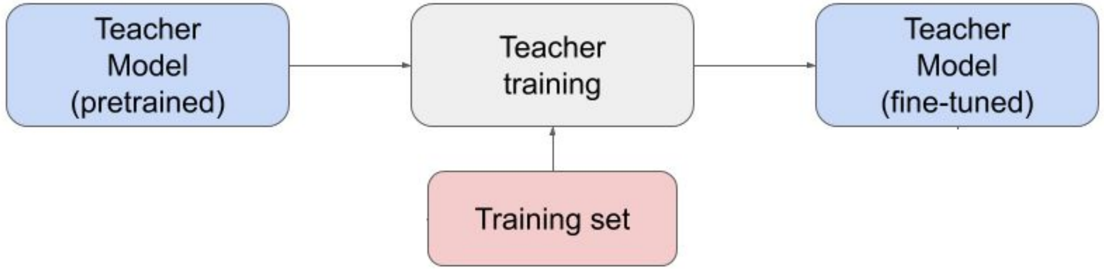
SOFT LABELS

Sometimes 'Sparse Labels'

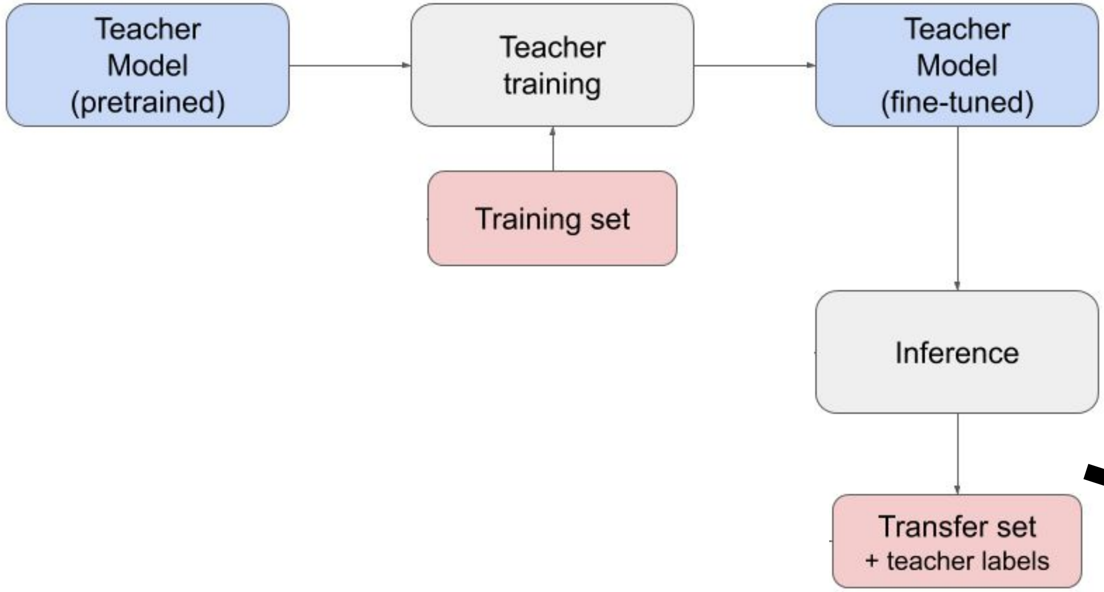
End-goal: Gather more information with less data

Teacher creates *Soft Labels*

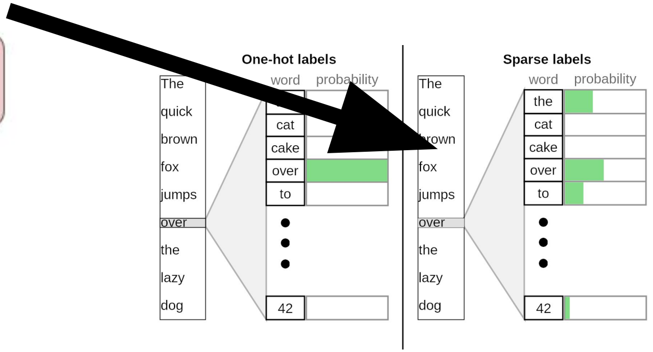



Teacher Model (pretrained)
Teacher training
Teacher Model (fine-tuned)
Training set

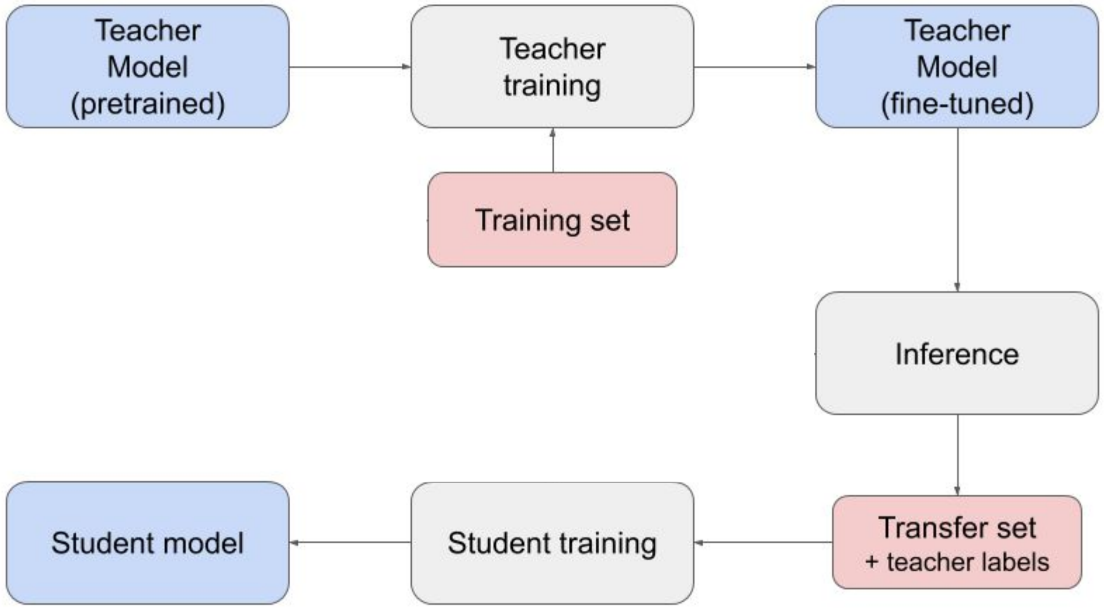
WORKFLOW


Teacher Model (pretrained)
Teacher training
Training set
Teacher Model (fine-tuned)
Inference
Transfer set + teacher labels

WORKFLOW


One-hot labels
Sparse labels
word
probability
The
quick
brown
fox
jumps
over
the
lazy
dog
the
cat
cake
over
to
42

WORKFLOW

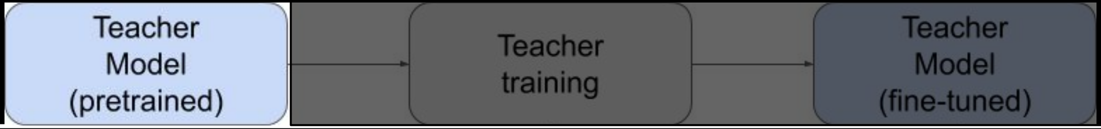

Teacher Model (pretrained)
Teacher training
Training set
Teacher Model (fine-tuned)
Inference
Transfer set + teacher labels
Student training
Student model

05 CODE

Let’s code 🎉

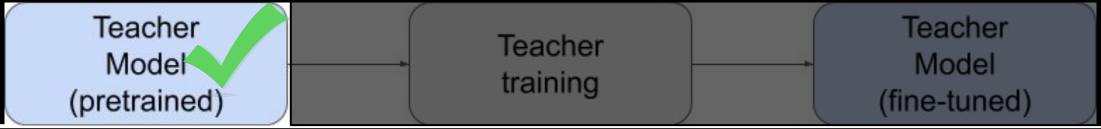
LOADING A TEACHER MODEL

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification



tokenizer = AutoTokenizer.from_pretrained("bert-large-cased")
```



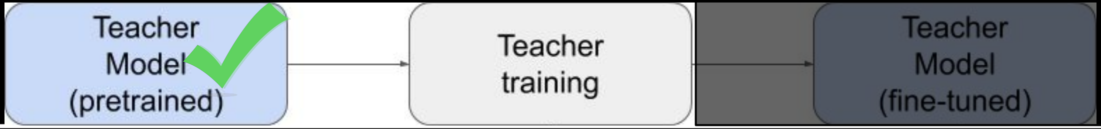
LOADING A TEACHER MODEL

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification



tokenizer = AutoTokenizer.from_pretrained("bert-large-cased")
model = AutoModelForSequenceClassification.from_pretrained("bert-large-cased")
```



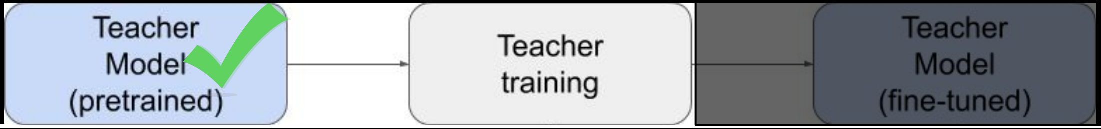
TRAINING A TEACHER MODEL

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification,
Trainer, TrainingArguments


tokenizer = AutoTokenizer.from_pretrained("bert-large-cased")
model = AutoModelForSequenceClassification.from_pretrained("bert-large-cased")
```



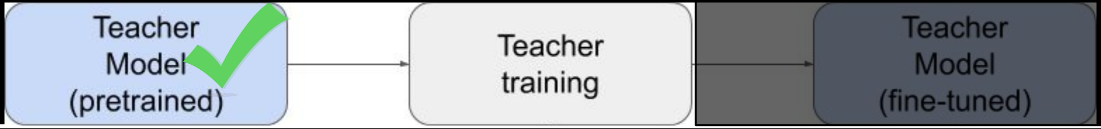
TRAINING A TEACHER MODEL

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification,
Trainer, TrainingArguments


tokenizer = AutoTokenizer.from_pretrained("bert-large-cased")
model = AutoModelForSequenceClassification.from_pretrained("bert-large-cased")

training_args = TrainingArguments(
    """... inserts params like batch-size, weight-decay etc ..."""
)
```



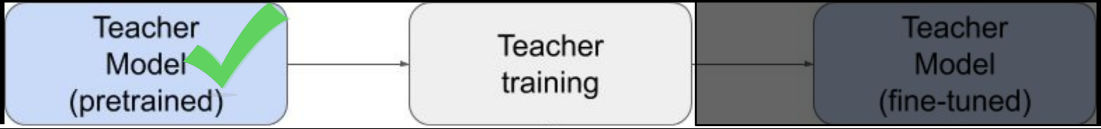
TRAINING A TEACHER MODEL

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification,
Trainer, TrainingArguments


tokenizer = AutoTokenizer.from_pretrained("bert-large-cased")
model = AutoModelForSequenceClassification.from_pretrained("bert-large-cased")

training_args = TrainingArguments(
    """... inserts params like batch-size, weight-decay etc ..."""
)

trainer = Trainer(model, training_args, train_dataset,  test_dataset)
```



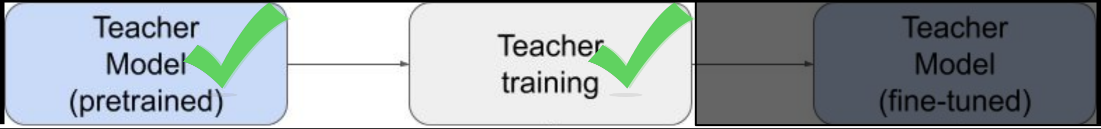
TRAINING A TEACHER MODEL

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification,
Trainer, TrainingArguments


tokenizer = AutoTokenizer.from_pretrained("bert-large-cased")
model = AutoModelForSequenceClassification.from_pretrained("bert-large-cased")

training_args = TrainingArguments(
    """... inserts params like batch-size, weight-decay etc ..."""
)

trainer = Trainer(model, training_args, train_dataset,  test_dataset)

trainer.train()
trainer.evaluate()
```



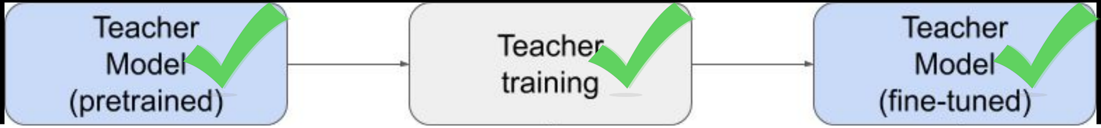
TRAINING A TEACHER MODEL

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification,
Trainer, TrainingArguments


tokenizer = AutoTokenizer.from_pretrained("bert-large-cased")
model = AutoModelForSequenceClassification.from_pretrained("bert-large-cased")

training_args = TrainingArguments(
    """... inserts params like batch-size, weight-decay etc ..."""
)

trainer = Trainer(model, training_args, train_dataset,  test_dataset)

trainer.train()
trainer.evaluate()
```



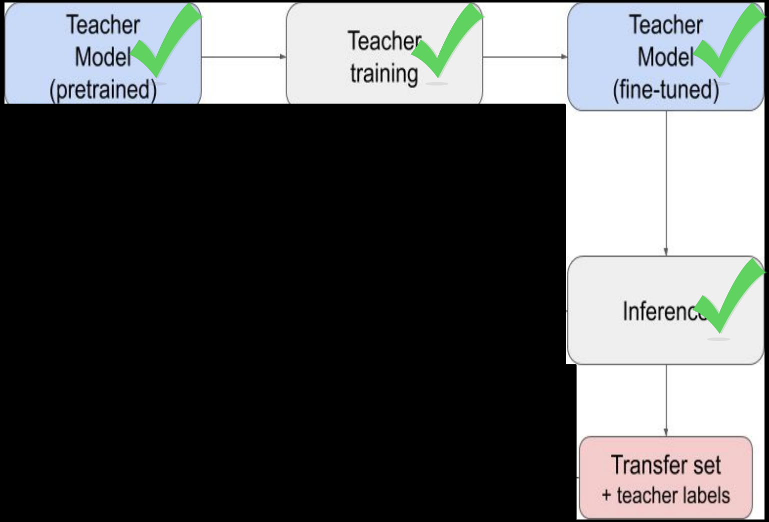
TRAINING A TEACHER MODEL

```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification,
Trainer, TrainingArguments


tokenizer = AutoTokenizer.from_pretrained("bert-large-cased")
model = AutoModelForSequenceClassification.from_pretrained("bert-large-cased")

training_args = TrainingArguments(
    """... inserts params like batch-size, weight-decay etc ..."""
)

trainer = Trainer(model, training_args, train_dataset,  test_dataset)

trainer.train()
trainer.evaluate()
```



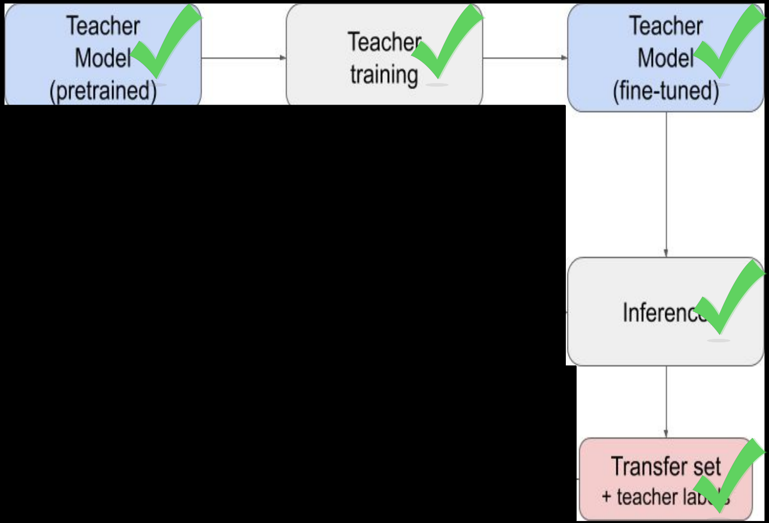
CREATING TRANSFER SET

```
def make_teacher_labels(dataset: Dataset, trainer: Trainer):
    preds = trainer.predict(dataset) # Batch predictions
    pandas_dataset = dataset.to_pandas()
    pandas_dataset[['label_1','label_2']] = preds.predictions
    return dataset.from_pandas(pandas_dataset)
```


Teacher Model (pretrained)
Teacher training
Teacher Model (fine-tuned)
Inference
Transfer set + teacher labels

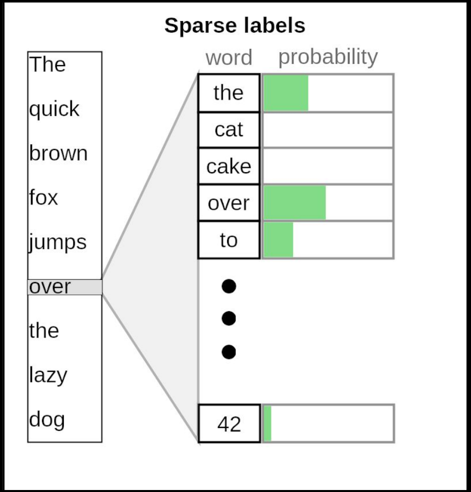
CREATING TRANSFER SET

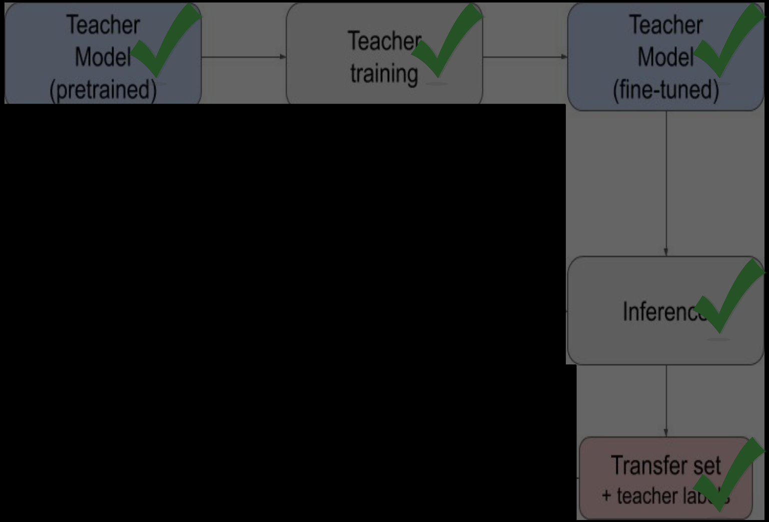
```
def make_teacher_labels(dataset: Dataset, trainer: Trainer):
    preds = trainer.predict(dataset) # Batch predictions
    pandas_dataset = dataset.to_pandas()
    pandas_dataset[['label_1','label_2']] = preds.predictions
    return dataset.from_pandas(pandas_dataset)
```


Teacher Model (pretrained)
Teacher training
Teacher Model (fine-tuned)
Inference
Transfer set + teacher labels

CREATING TRANSFER SET

```
def make_teacher_labels(dataset: Dataset, trainer: Trainer):
    preds = trainer.predict(dataset) # Batch predictions
    pandas_dataset = dataset.to_pandas()
    pandas_dataset[['label_1','label_2']] = preds.predictions
    return dataset.from_pandas(pandas_dataset)
```

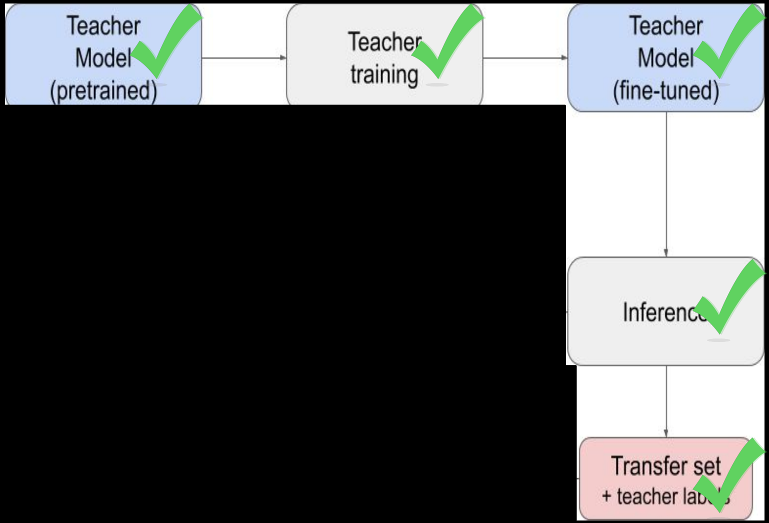




Teacher Model (pretrained)
Teacher training
Teacher Model (fine-tuned)
Inference
Transfer set + teacher labels

CREATING TRANSFER SET

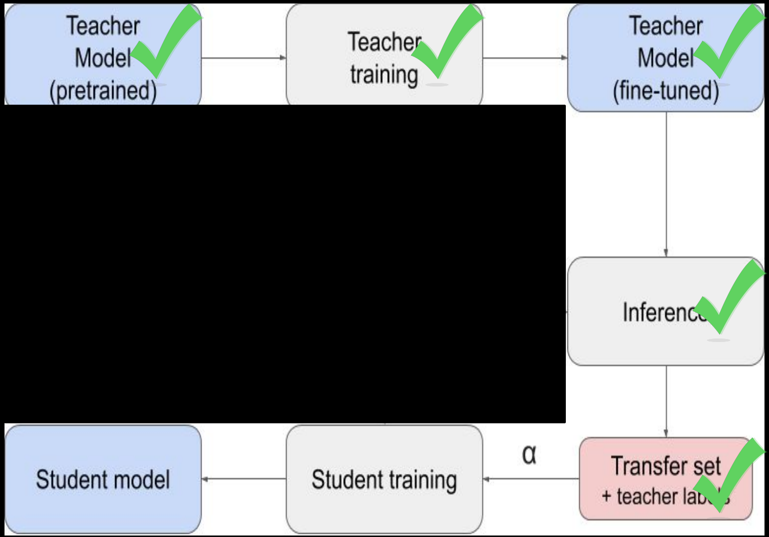
```
def make_teacher_labels(dataset: Dataset, trainer: Trainer):
    preds = trainer.predict(dataset) # Batch predictions
    pandas_dataset = dataset.to_pandas()
    pandas_dataset[['label_1','label_2']] = preds.predictions
    return dataset.from_pandas(pandas_dataset)
```



Teacher Model (pretrained)
Teacher training
Teacher Model (fine-tuned)
Inference
Transfer set + teacher lab

TRAINING STUDENT

Won’t go into details.


Teacher Model (pretrained)
Teacher training
Teacher Model (fine-tuned)
Inference
Transfer set + teacher labels
α
Student training
Student model

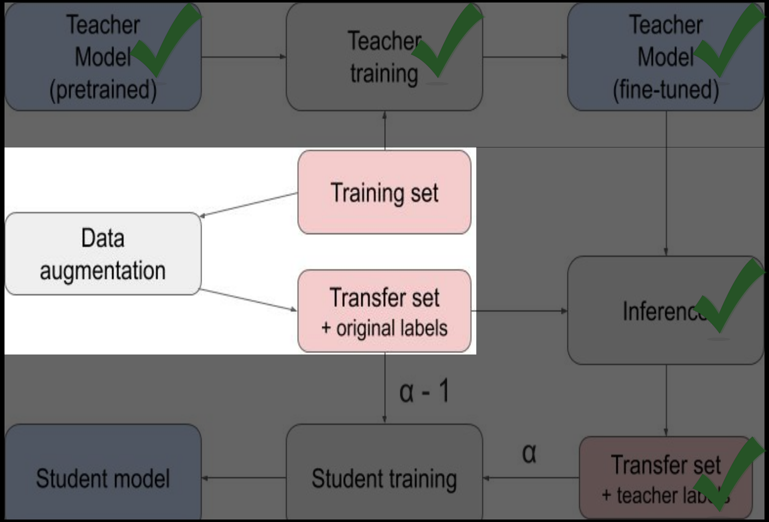
IMPROVING SIGNALS

IMPROVING SIGNALS

Classic: More data

This time through Data Augmentation via Teacher

(Tang et al.)


Teacher Model (pretrained)
Teacher training
Teacher Model (fine-tuned)
Training set
Data augmentation
Transfer set + original labels
Inference
α - 1
α
Student model
Student training
Transfer set + teacher labels

AUGMENTING DATA

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

AUGMENTING DATA

```python
def make_sample(input_sentence, pos_dict, p_mask=0.1):
    sentence = []
    for word in input_sentence:
        x = random.uniform()
```

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

AUGMENTING DATA

```
def make_sample(input_sentence, pos_dict, p_mask=0.1):
    sentence = []
    for word in input_sentence:
        x = random.uniform()
        if x < p_mask:
            sentence.append(mask_token)
```

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

AUGMENTING DATA

```python
def make_sample(input_sentence, pos_dict, p_mask=0.1):
    sentence = []
    for word in input_sentence:
        x = random.uniform()
        if x < p_mask:
            sentence.append(mask_token)
        else:
            sentence.append(word)
```

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

AUGMENTING DATA

```
def make_sample(input_sentence, pos_dict, p_mask=0.1):
    sentence = []
    for word in input_sentence:
        x = random.uniform()
        if x < p_mask:
            sentence.append(mask_token)
        else:
            sentence.append(word)
```

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

2: Replace Word by Equal POS

Replace one noun with another noun, or a verb with another verb

AUGMENTING DATA

```
def make_sample(input_sentence, pos_dict, p_mask=0.1, p_pos=0.1):
    sentence = []
    for word in input_sentence:
        x = random.uniform()
        if x < p_mask:
            sentence.append(mask_token)
        elif x < (p_mask + p_pos):
            same_pos = pos_dict[word.pos]
            sentence.append(random.choice(same_pos))
        else:
            sentence.append(word)
```

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

2: Replace Word by Equal POS

Replace one noun with another noun, or a verb with another verb

AUGMENTING DATA

```
def make_sample(input_sentence, pos_dict, p_mask=0.1, p_pos=0.1):
    sentence = []
    for word in input_sentence:
        x = random.uniform()
        if x < p_mask:
            sentence.append(mask_token)
        elif x < (p_mask + p_pos):
            same_pos = pos_dict[word.pos]
            sentence.append(random.choice(same_pos))
        else:
            sentence.append(word)
```

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

2: Replace Word by Equal POS

Replace one noun with another noun, or a verb with another verb

3: Ngram Sampling

I do enjoy a good pizza → a good pizza
(randomly keep only 1-5 words)

AUGMENTING DATA

```
def make_sample(input_sentence, pos_dict, p_mask=0.1, p_pos=0.1, p_ng=0.25):
    sentence = []
    for word in input_sentence:
        x = random.uniform()
        if x < p_mask:
            sentence.append(mask_token)
        elif x < (p_mask + p_pos):
            same_pos = pos_dict[word.pos]
            sentence.append(random.choice(same_pos))
        else:
            sentence.append(word)

    if random.uniform() < p_ng:
        n = random.choice(range(0, 5)) + 1
        sentence = sample(sentence, n)

    return sentence
```

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

2: Replace Word by Equal POS

Replace one noun with another noun, or a verb with another verb

3: Ngram Sampling

I do enjoy a good pizza → a good pizza
(randomly keep only 1-5 words)

AUGMENTING DATA

```
def make_sample(input_sentence, pos_dict, p_mask=0.1, p_pos=0.1, p_ng=0.25):
    sentence = []
    for word in input_sentence:
        x = random.uniform()
        if x < p_mask:
            sentence.append(mask_token)
        elif x < (p_mask + p_pos):
            same_pos = pos_dict[word.pos]
            sentence.append(random.choice(same_pos))
        else:
            sentence.append(word)

    if random.uniform() < p_ng:
        n = random.choice(range(0, 5)) + 1
        sentence = sample(sentence, n)

    return sentence
```

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

2: Replace Word by Equal POS

Replace one noun with another noun, or a verb with another verb

3: Ngram Sampling

I do enjoy a good pizza → a good pizza
(randomly keep only 1-5 words)

AUGMENTING DATA

```
def make_sample(input_sentence, pos_dict, p_mask=0.1, p_pos=0.1, p_ng=0.25):
    sentence = []
    for word in input_sentence:
        x = random.uniform()
        if x < p_mask:
            sentence.append(mask_token)
        elif x < (p_mask + p_pos):
            same_pos = pos_dict[word.pos]
            sentence.append(random.choice(same_pos))
        else:
            sentence.append(word)

    if random.uniform() < p_ng: # Alt. Mask ngram
        n = random.choice(range(0, 5)) + 1
        start = random.choice(len(sentence) - n)
        for idx in range(start, start + n):
            sentence[idx] = mask_token
    return sentence
```

1: Mask Random Words

I enjoy pizza → I [MASK] pizza

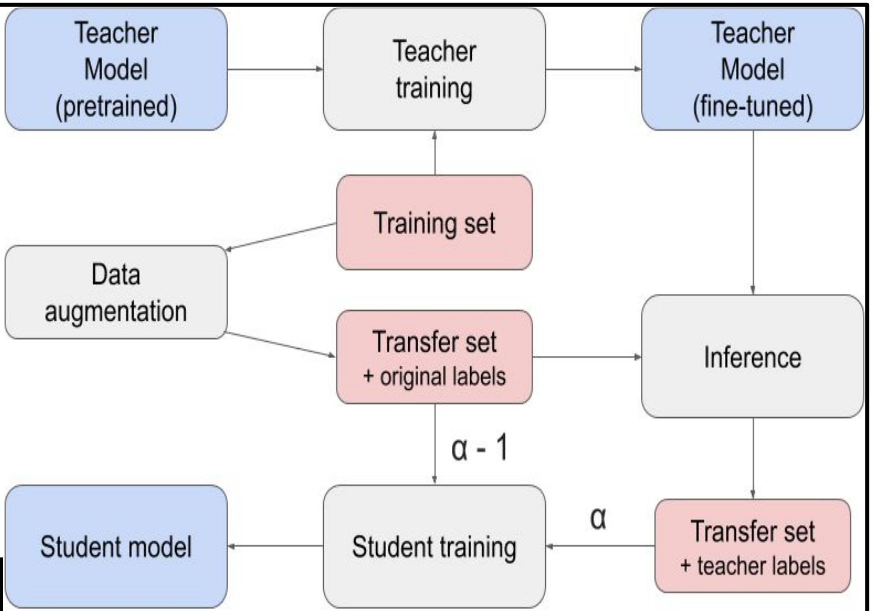
2: Replace Word by Equal POS

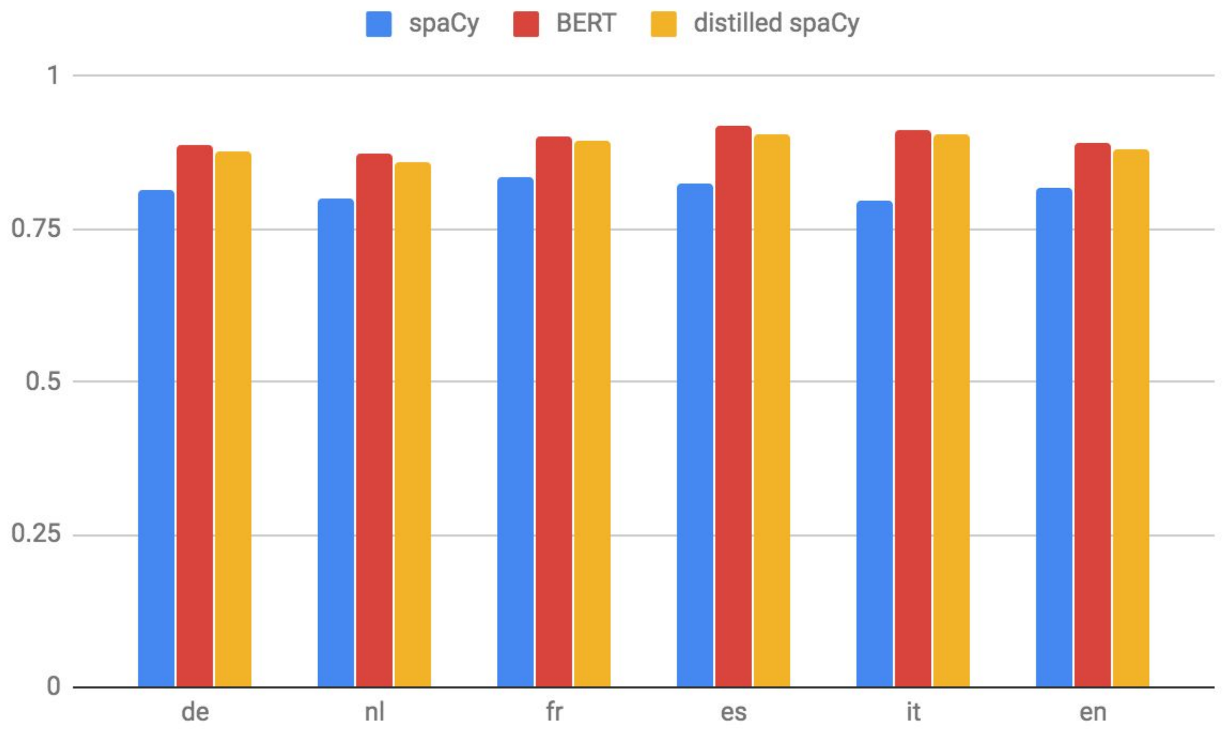
Replace one noun with another noun, or a verb with another verb

3: Ngram Sampling

I do enjoy a good pizza → a good pizza
(randomly keep only 1-5 words)

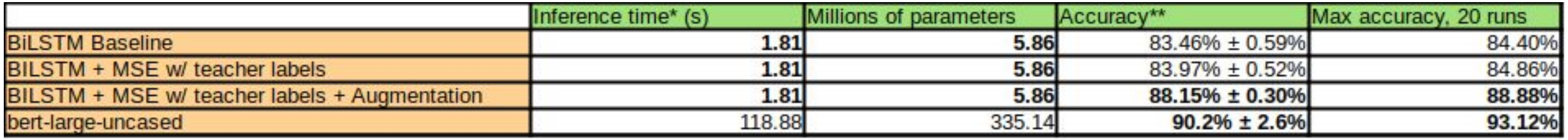
Loop completed


Teacher Model (pretrained)
Teacher training
Teacher Model (fine-tuned)
Training set
Data augmentation
Transfer set + original labels
Inference
α - 1
Student model
Student training
α
Transfer set + teacher labels



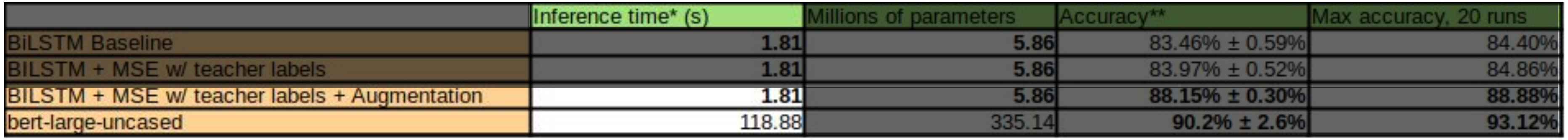
NLP.TOWN

FLOYDHUB



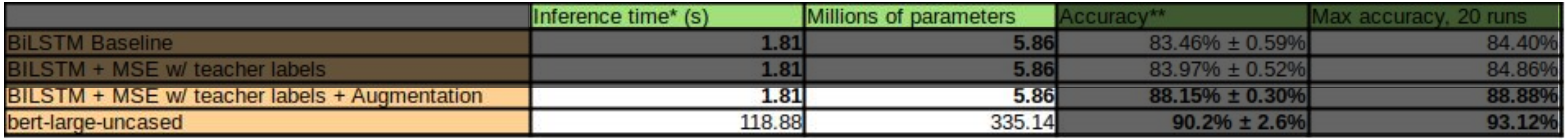
	Inference time* (s)	Millions of parameters	Accuracy**	Max accuracy, 20 runs
BiLSTM Baseline	**1.81**	**5.86**	83.46% ± 0.59%	84.40%
BILSTM + MSE w/ teacher labels	**1.81**	**5.86**	83.97% ± 0.52%	84.86%
BILSTM + MSE w/ teacher labels + Augmentation	**1.81**	**5.86**	**88.15% ± 0.30%**	**88.88%**
bert-large-uncased	118.88	335.14	**90.2% ± 2.6%**	**93.12%**

FLOYDHUB



	Inference time* (s)	Millions of parameters	Accuracy**	Max accuracy, 20 runs
BiLSTM Baseline	**1.81**	**5.86**	83.46% ± 0.59%	84.40%
BILSTM + MSE w/ teacher labels	**1.81**	**5.86**	83.97% ± 0.52%	84.86%
BILSTM + MSE w/ teacher labels + Augmentation	**1.81**	**5.86**	**88.15% ± 0.30%**	**88.88%**
bert-large-uncased	118.88	335.14	**90.2% ± 2.6%**	**93.12%**

FLOYDHUB



	Inference time* (s)	Millions of parameters	Accuracy**	Max accuracy, 20 runs
BiLSTM Baseline	**1.81**	**5.86**	83.46% ± 0.59%	84.40%
BILSTM + MSE w/ teacher labels	**1.81**	**5.86**	83.97% ± 0.52%	84.86%
BILSTM + MSE w/ teacher labels + Augmentation	**1.81**	**5.86**	**88.15% ± 0.30%**	**88.88%**
bert-large-uncased	118.88	335.14	**90.2% ± 2.6%**	**93.12%**

06 RECAP

01

Model Compression
&
Model Change

06 RECAP

01

Model Compression
&
Model Change

02

Soft Labels

06 RECAP

01

Model Compression
&
Model Change

02

Soft Labels

03

Data Augmentation

DISCUSSION POINTS

BIAS

What biases are implied?

CEILING

What ceiling exists?

THANKS

Questions?

hampus.londogard@afry.com
0733 673 179

[November - Presentation] Transfer Learning
How can we transfer knowledge from a base-task into new specific tasks, achieving SotA results with very little data?
(stream)

[December - Workshop] Transformers in Visual Recoginition
How can we use Transformers to find objects in images, even if the Transformer isn't hand-curated for vision?
(stream, notebook, code)

[February - Workshop] Self-Attention & Transformers from scratch
How does the Transformer work under the hood? What makes them so powerful and why do they scale so well?
(stream, blog/notebook)