

Transfer Learning



Peerapon Vateekul, Ph.D
(Credit: Ammarin Jetthakun, Can Udomcharoenchaikit)



Outline

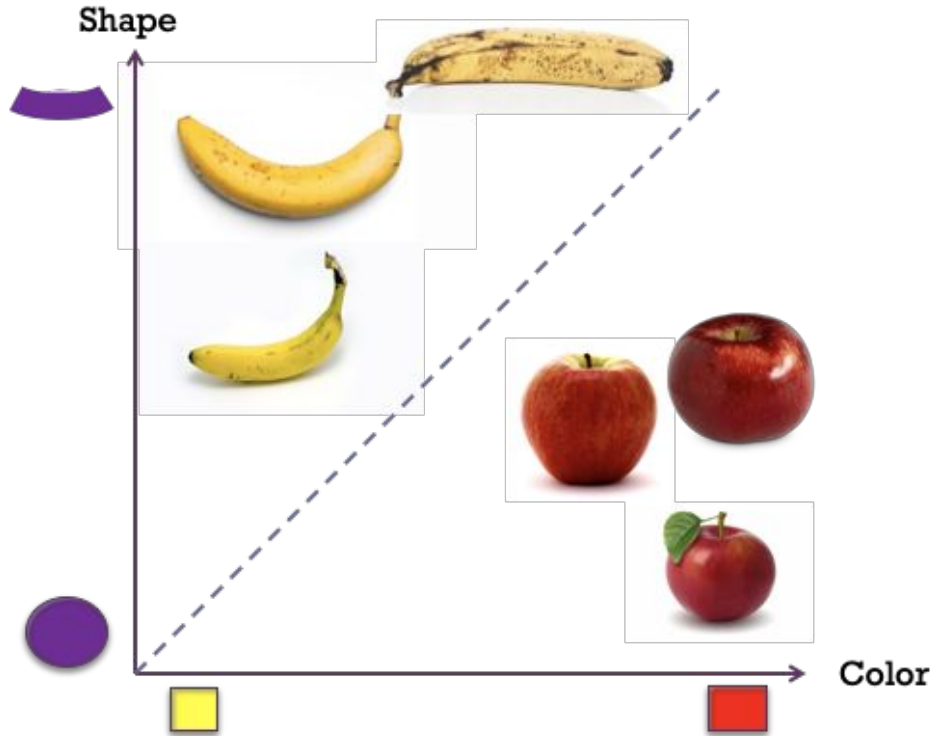
1. Introduction of transfer learning
2. Transfer learning for NLP
3. Conclusion

1

Introduction of Transfer learning



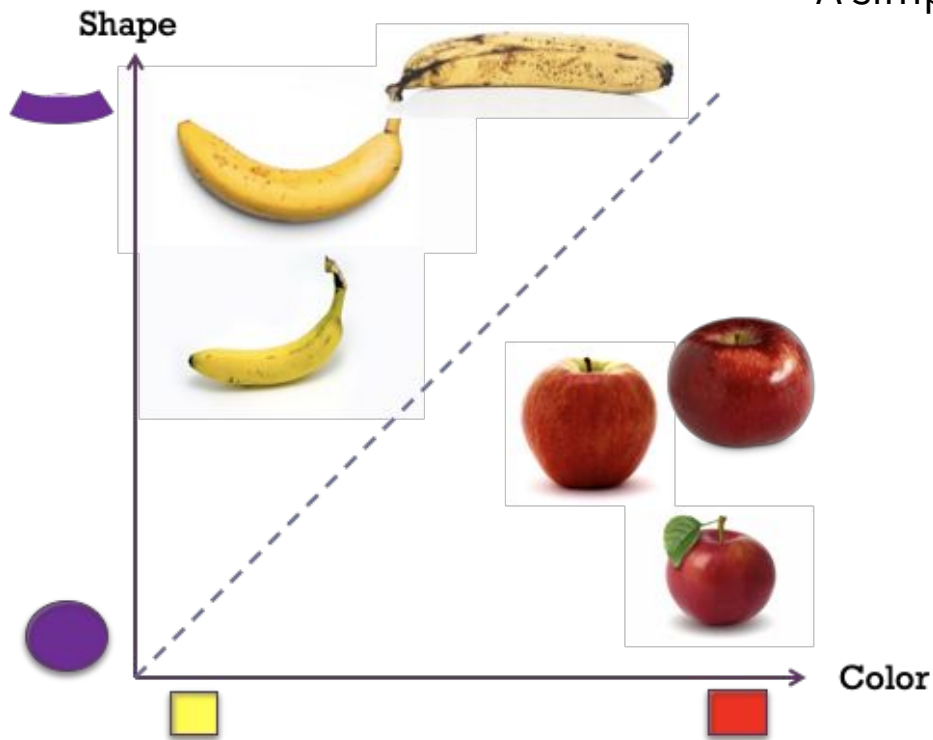
We need **enough** training data
to infer the data pattern and to create model





We need **enough** training data to infer the data pattern and to create model (cont.)

A simple problem requires less training data.



รวมปรากฏการณ์ นายกรัฐมนตรี ปรี๊ดแตก! สื่อต้องกลา...
thairath.co.th



โพลซี "บักตุ" ยังเหมาะนั่งนายกรัฐมนตรีเพราะสถาน...
posttoday.com



ประวัติความเป็นมาของลุงตู่นายกคนปัจจุบัน !!
stu40406site.wordpress.com



Have you ever seen this creature before?
Can you guess whether it is a land animal or a water animal?





Have you ever seen this creature before?
Can you guess whether it is a land animal or a water animal?
You can transfer your knowledge from the past





Transfer learning

Myth: you can't do deep learning unless you have a million labelled examples for your problem.

Reality:

- You can **transfer** learned representations from **a related task**
- You can train on a nearby **surrogate objective** for which it is easy to generate labels



Transfer learning: idea

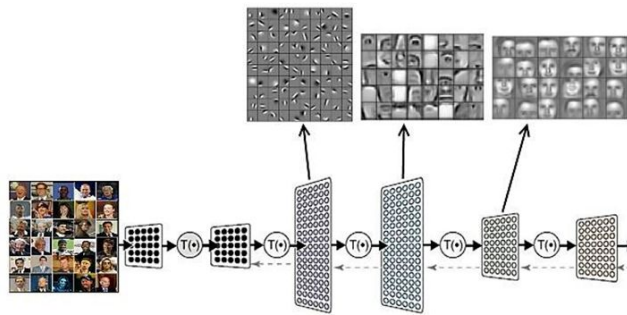
Instead of training a deep network from scratch for your task, you can

- Take a network trained on **a different domain** for a **different source task**
- **Adapt** it for your domain and your **target task**

This lecture will talk about how to do this.

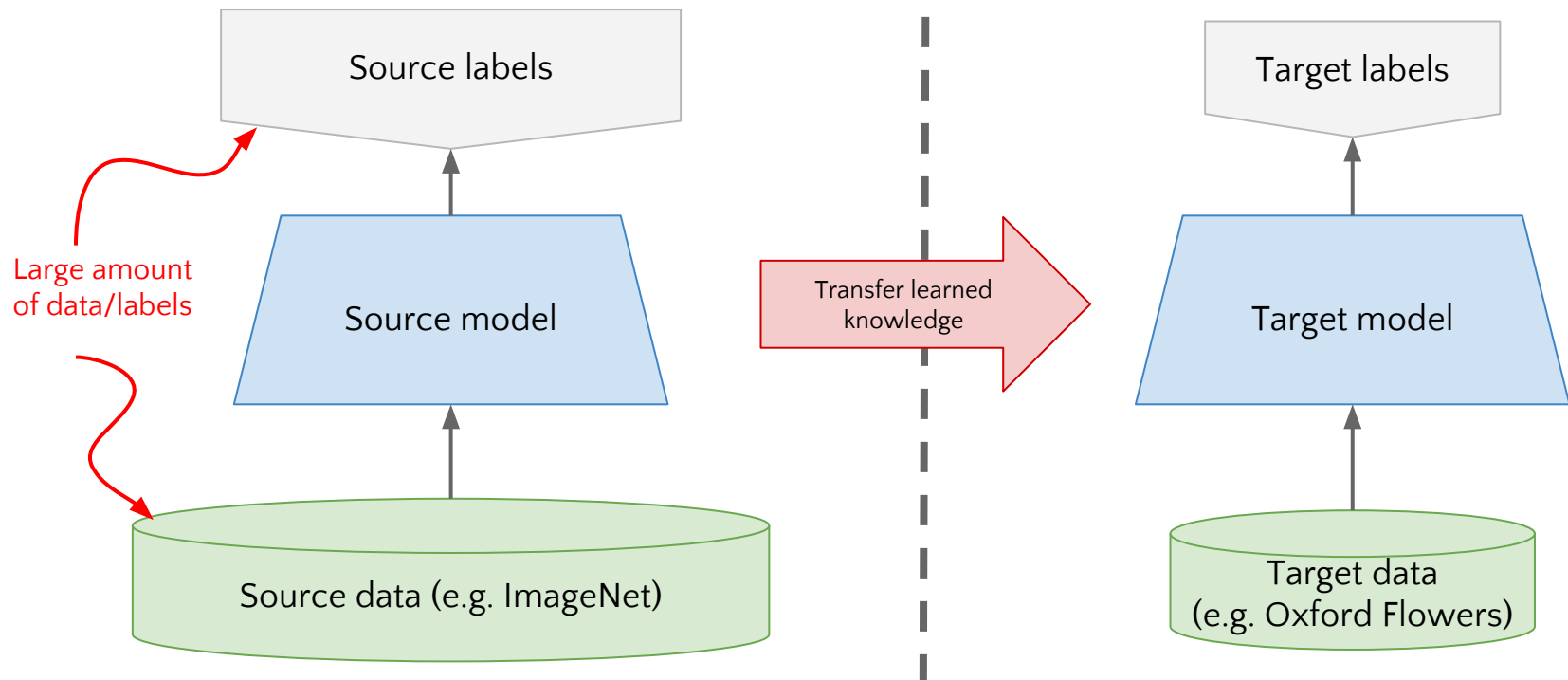
Variations:

- Different domain (data), same task
- Different domain (data), different task





Transfer learning: idea





<http://www.image-net.org/>

IMAGENET

14,197,122 images, 21841 synsets indexed

[Explore](#) [Download](#) [Challenges](#) [Publications](#) [CoolStuff](#) [About](#)

Not logged in. [Login](#) | [Signup](#)

Object recognition: 14M++ images on 20K++ categories

ImageNet is an image database organized according to the **WordNet** hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.

[Click here](#) to learn more about ImageNet, [Click here](#) to join the ImageNet mailing list.

ImageNet

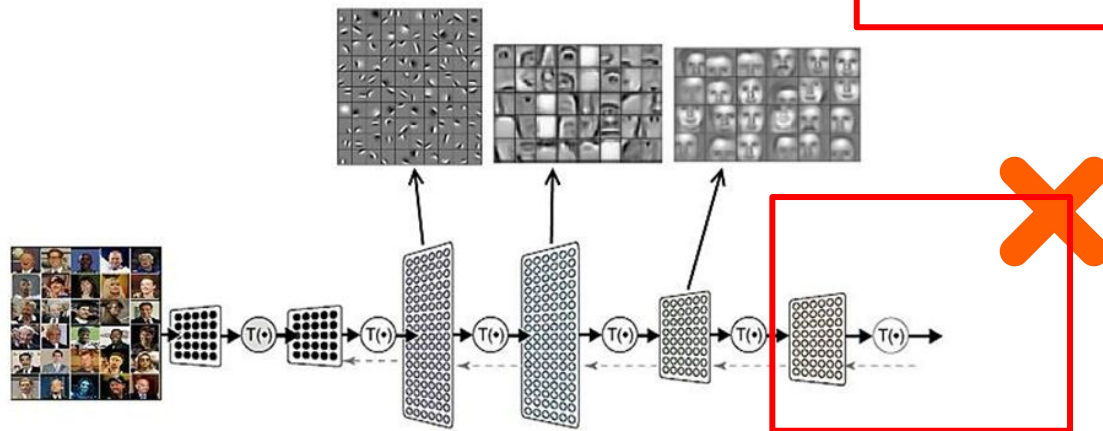
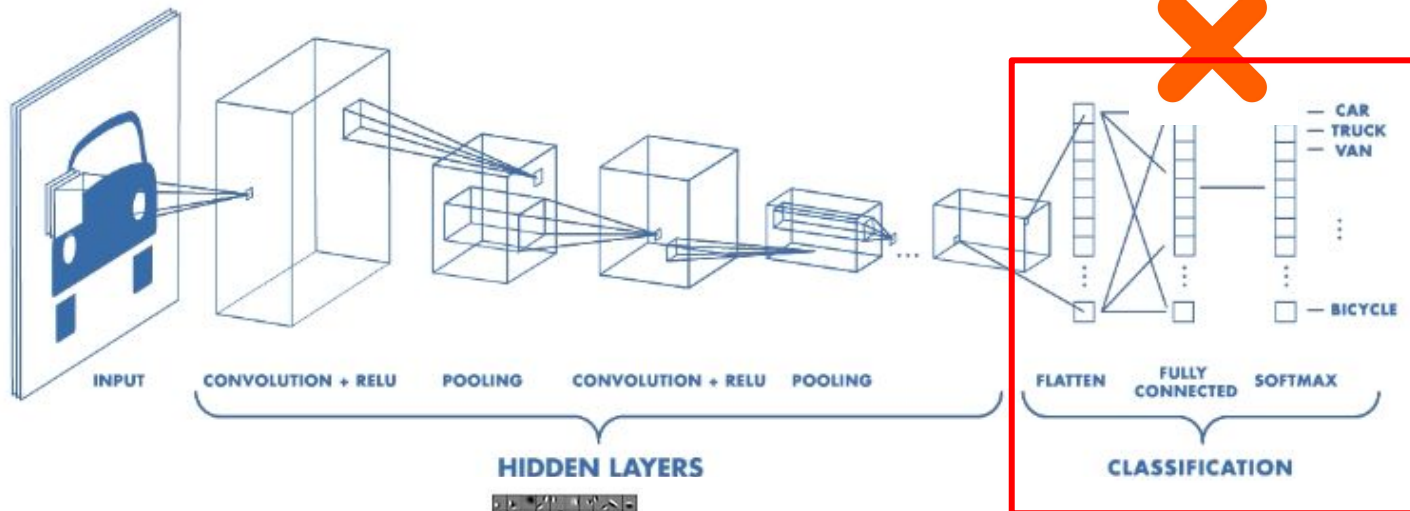
From Wikipedia, the free encyclopedia

The **ImageNet** project is a large visual [database](#) designed for use in [visual object recognition software](#) research. More than 14 million^{[1][2]} images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided.^[3] ImageNet contains more than 20,000 categories^[2] with a typical category, such as "balloon" or "strawberry", consisting of several hundred images.^[4] The database of annotations of third-party





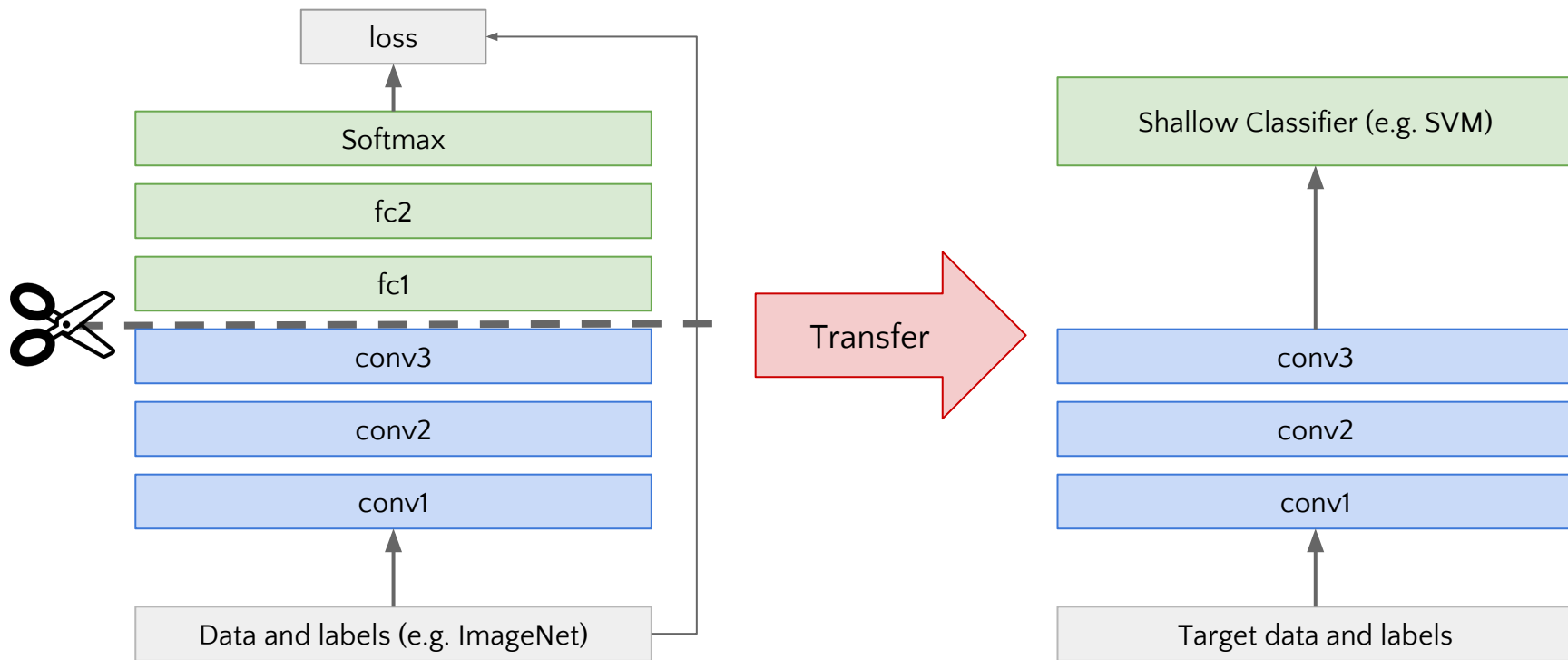
“Off-the-shelf (Feature Extractor)”





“Off-the-shelf (Feature Extractor)” (cont.)

Idea: use outputs of one or more layers of a network trained on a different task as **generic feature detectors**.
Train a new shallow model on these features.





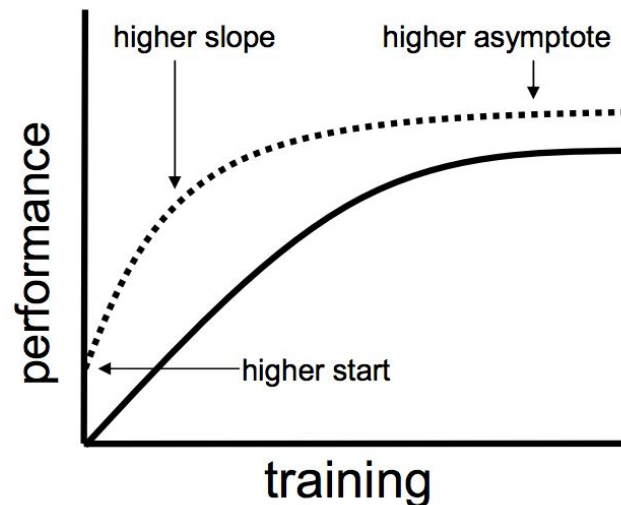
Transfer learning: 3 benefits



Lisa Torrey and Jude Shavlik in their chapter on transfer learning describe **three possible benefits** to look for when using transfer learning:

1. **Higher start.** The initial skill (before refining the model) on the source model is higher than it otherwise would be.
2. **Higher slope.** The rate of improvement of skill during training of the source model is steeper than it otherwise would be.
3. **Higher asymptote.** The converged skill of the trained model is better than it otherwise would be.

..... with transfer
— without transfer



Can we do better than off the shelf features ?



“



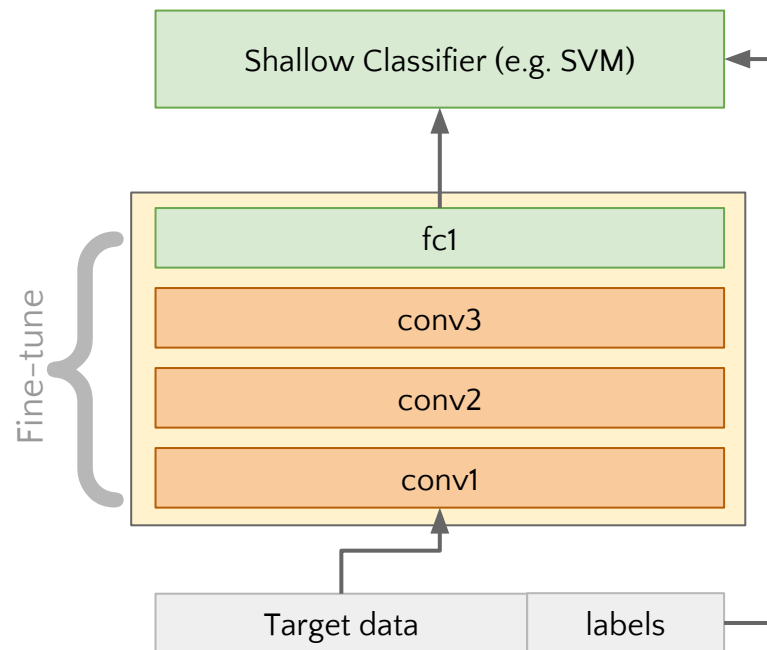
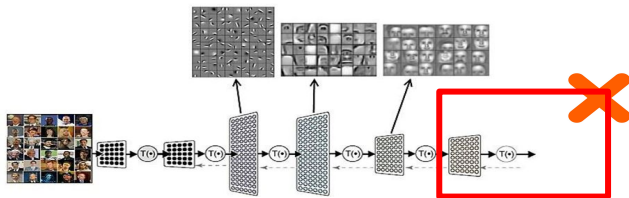
Fine-tuning: supervised task adaptation

Train deep net on **nearby task** for which it is **easy to get labels** using standard backprop.

- E.g. ImageNet classification
- Pseudo classes from augmented data
- Slow feature learning, ego-motion

Cut off top layer(s) of network and **replace with supervised objective for target domain**.

Fine-tune network using backprop with labels for target domain until validation loss starts to increase.





How transferable are features

Lower layers: more general features.

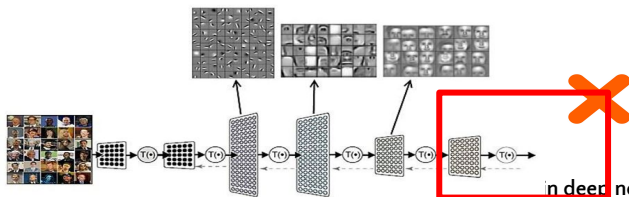
Transfer very well to other tasks.

Higher layers: more task specific.

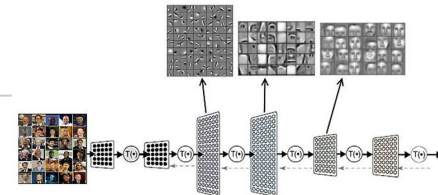
Fine-tuning improves generalization when sufficient examples are available.

Transfer learning and fine tuning often lead to better performance than training from scratch on the target dataset.

Even features transferred from **distant tasks** are often better than random initial weights!



in deep neural networks, NIPS 2014 <https://arxiv.org/abs/1411.1792>

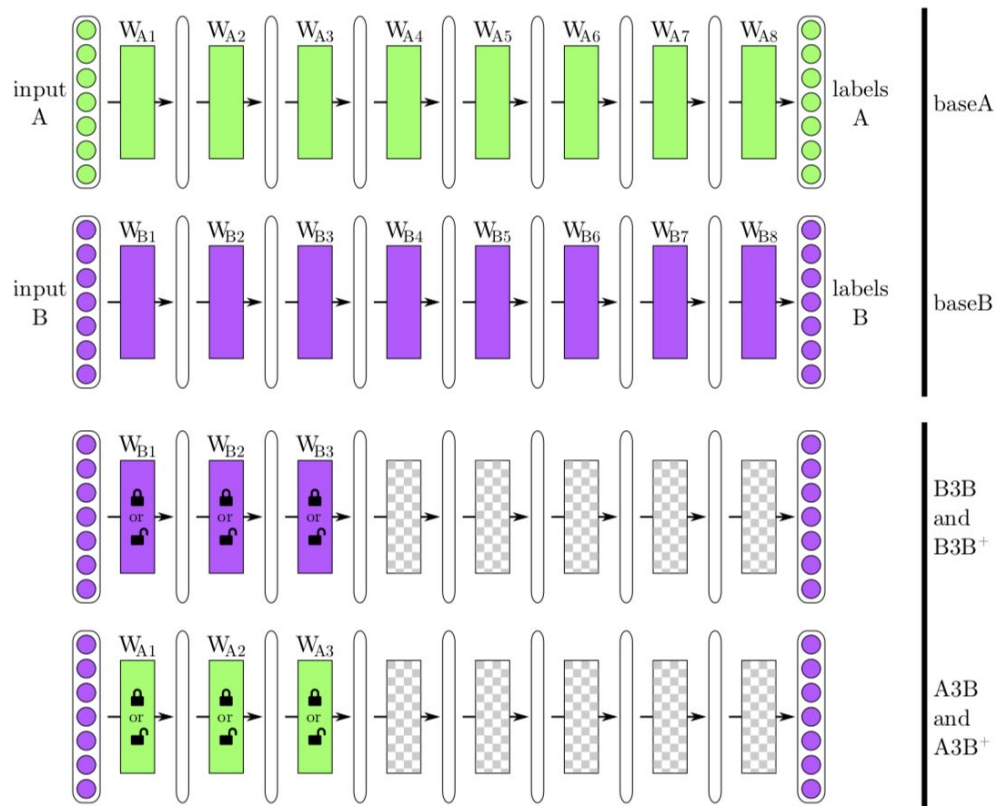


More specific

More generic



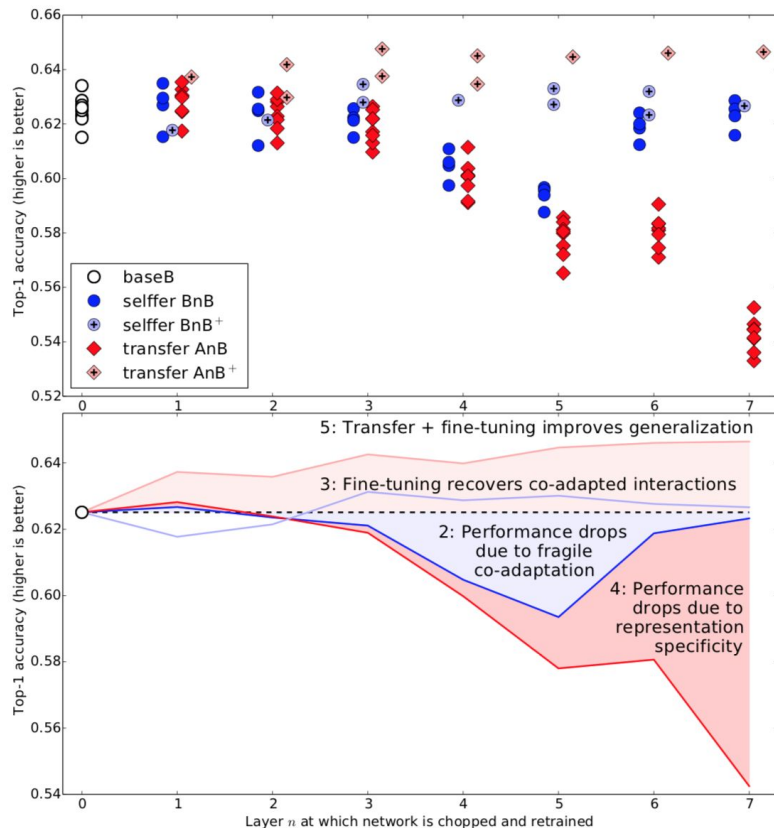
How transferable are features



- **A selfer network B3B:** the first 3 layers are copied from baseB and frozen. The five higher layers (4–8) are initialized randomly and trained on dataset B. This network is a control for the next transfer network.
- **A transfer network A3B:** the first 3 layers are copied from base A and frozen. The five higher layers (4–8) are initialized randomly and trained toward dataset B. Intuitively, here we copy the first 3 layers from a network trained on dataset A and then learn higher layer features on top of them to classify a new target dataset B. If A3B performs as well as baseB, there is evidence that the third-layer features are general, at least with respect to B. If performance suffers, there is evidence that the third-layer features are specific to A.
- **B3B⁺ and A3B⁺** are just like B3B and A3B, but where all transferred layers are *fine-tuned*.



How transferable are features



Co-adaptation issue is a problem that layers interact with each other.

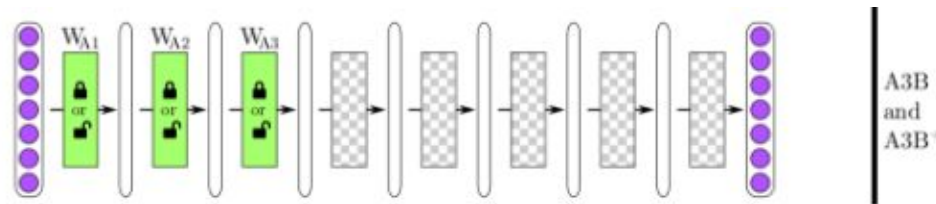


Figure 2: The results from this paper’s main experiment. *Top*: Each marker in the figure represents the average accuracy over the validation set for a trained network. The white circles above $n = 0$ represent the accuracy of baseB. There are eight points, because we tested on four separate random A/B splits. Each dark blue dot represents a BnB network. Light blue points represent BnB⁺ networks, or fine-tuned versions of BnB. Dark red diamonds are AnB networks, and light red diamonds are the fine-tuned AnB⁺ versions. Points are shifted slightly left or right for visual clarity. *Bottom*: Lines connecting the means of each treatment. Numbered descriptions above each line refer to which interpretation from Section 4.1 applies.

2

Transfer learning for NLP



Transfer learning for NLP vs CV

NLP is more difficult than image domain

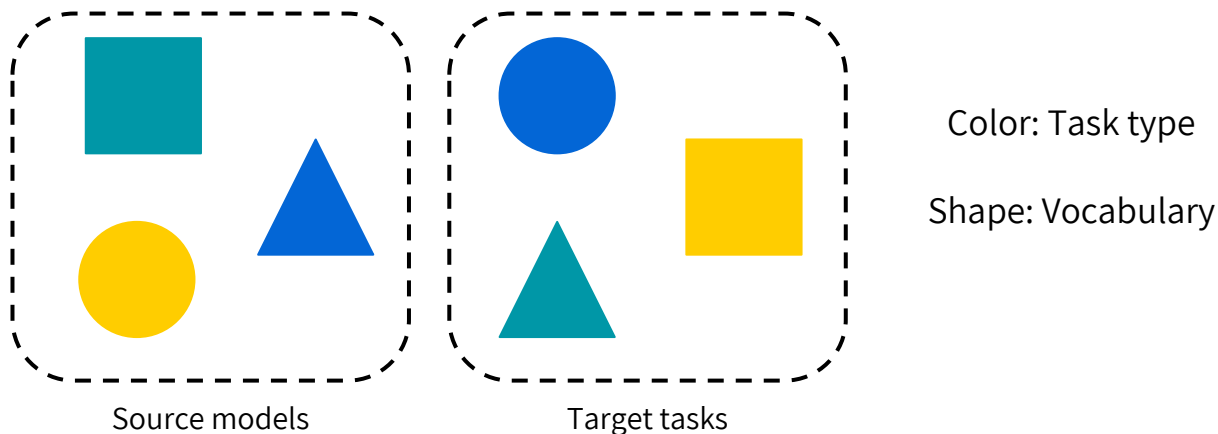
Differences between transfer learning for NLP and for Computer vision fields:

- More variety in types of **target tasks**.
 - E.g. entity extraction, classification, sequence labeling.
- More variety in **the input data** (e.g. source language, field-specific terminology).
- No clear “ImageNet” equivalent (lack of large, generic, and labeled **corpora**).
- Lack of consensus on **what source tasks (e.g., image classification)** produce good representations.



Model **Alignment** for transfer learning

- Source model is the single most important variable.
- Keep source model and target model **well-aligned (close to each other)** when possible.
- Source vocabulary should be aligned with target vocabulary (**similar domain**).
- Source task should be aligned with target task (**similar task**).
- For example:
 - **Good**: product review sentiment → product review categorization
 - **Good**: hotel rating → restaurant rating
 - **Less good**: product review sentiment → biology paper classification





1) What source tasks?

We expect the source tasks to be able to produce the good and general representations

- Natural language inference: are two sentences in agreement, disagreement, or neither?
- Machine translation: from one language to another language.
- Multi-task learning: learning to solve many supervised problems at once.
- **Language modeling:**
 - learning to model the distribution of natural language.
 - predicting the next word in a sequence given context.
 - No need for labeled data (unsupervised data)

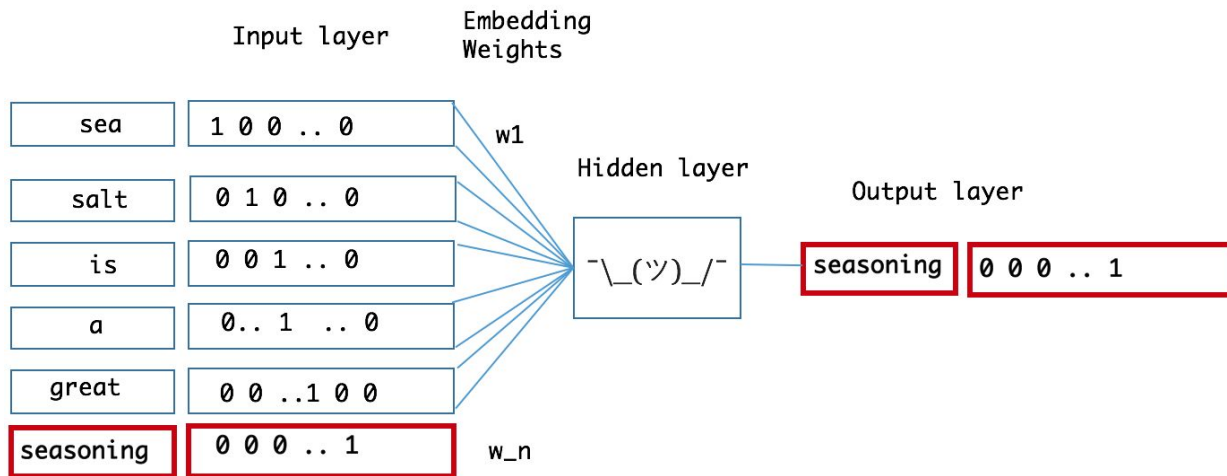


1) Language modeling: (cont.)

predicting the next word

Language modeling is a good objective for source model because:

- It is able to capture long-term dependencies in language (LSTM)
- Annotation is **not** required.
- It effectively help the model learn **syntactic (grammar)** and **semantic (meaning)** features





2) Pre-trained **corpus** for NLP

In NLP, for a couple of years now, the trend is to pre-train any model on the huge text corpus:

- BooksCorpus¹ (980M words)
- English Wikipedia² (2,300M words)
- WMT News crawl³ (3,600M words, only from 2008-2012)

All of these corpora are unlabeled. Since it focuses on LM (predicting next word), it doesn't need any labeling effort.

¹ Zhu et al., **Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books**, Proceedings of the IEEE international conference on computer vision, 2015, <https://arxiv.org/abs/1506.06724>

² <https://linguatools.org/tools/corpora/wikipedia-monolingual-corpora/>

³ <http://statmt.org/wmt18/translation-task.html>



Pre-trained Word2Vec from Lecture5 Word Representation

■ Two main Word2Vec models:

- Skip-gram
- CBOW

■ GloVe (Stanford)

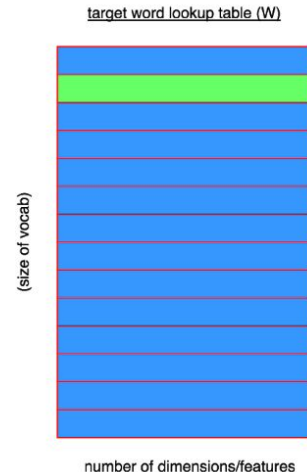
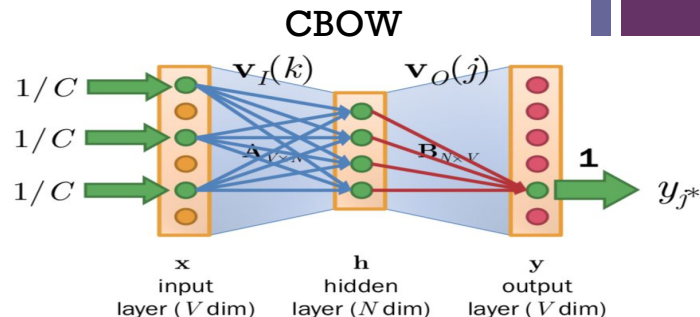
- <https://nlp.stanford.edu/projects/glove/>

■ fastText [Available in Thai language] (Facebook)

- <https://github.com/facebookresearch/fastText>

■ These word vectors do not depend on context.

- Stick? Bar (n) or Adhere (v)





Pre-trained language modeling

Pre-training allows a model to capture and learn a variety of linguistic phenomena, such as *long-term dependencies* and *negation*, from a **large-scale unlabeled corpus**.

Then this knowledge is used (transferred) to initialize and then train another model to perform well on a specific NLP tasks.

- ULMFiT (Universal Language Model Fine-tuning) ¹
- ELMo (Embeddings from Language Models) ²
- BERT (Bidirectional Encoder Representations from Transformers) ³
- Generative Pre-Training

The code of the models and their weights that already pre-trained on massive datasets are available for download.

¹ Jeremy Howard and Sebastian Ruder, **Universal Language Model Fine-tuning for Text Classification**, ACL 2018, <https://arxiv.org/pdf/1801.06146.pdf>

² Peter et al., **Deep contextualized word representations**, NAACL 2018, <https://arxiv.org/abs/1802.05365.pdf>

³ Devlin et al., **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**, arXiv 2018, <https://arxiv.org/pdf/1810.04805.pdf>



1) ULMFiT [Howard, et al, 2018]

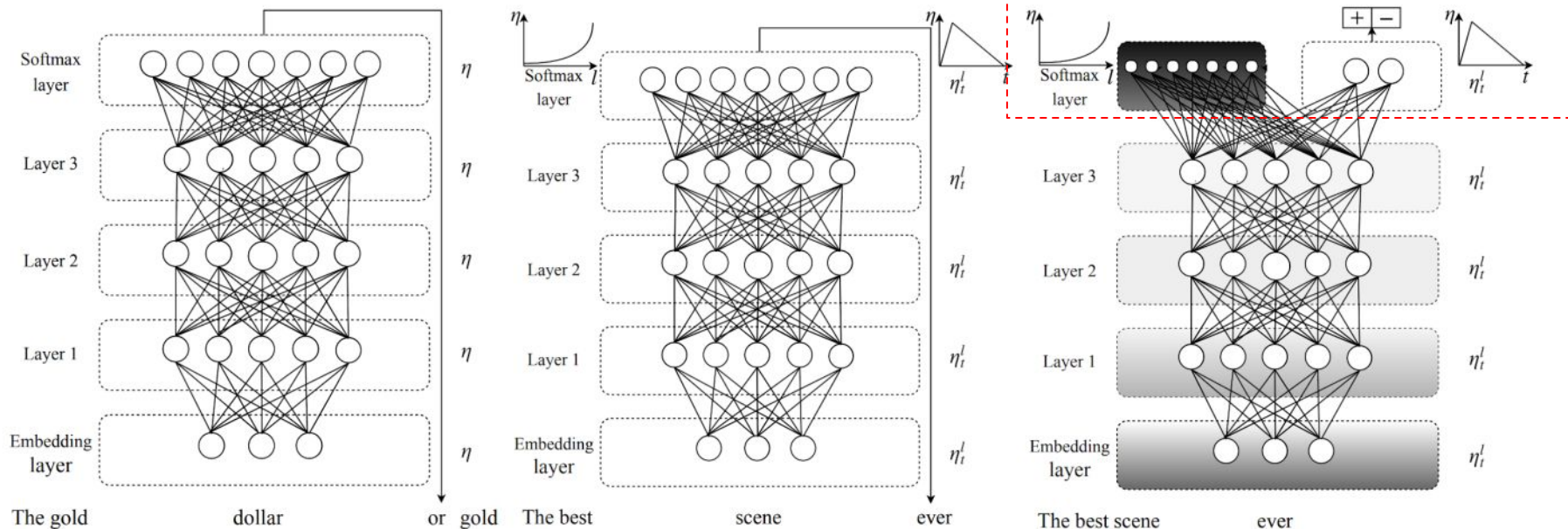
Universal Language Model Fine-tuning for Text Classification

- ULM-FiT introduced methods to effectively utilize a lot of what the model learns during pre-training.
- More than just a single embedding layer, ULMFiT introduced a language model which contains **many layers** and a process to effectively fine-tune that language model for various tasks
- Unlike BERT, this language model is **unidirectional LSTM** which means that each word is only contextualized using the words to its left.



ULMFiT (cont.)

The pretrained model from Step1 is available for download.



(a) Train LM using
large unlabeled corpus
(wiki corpus)

(b) Fine-tune LM using
target unlabeled corpus
(target data without label)

(c) Train classifier using
target corpus
(target data with label)



ULMFiT (cont.)

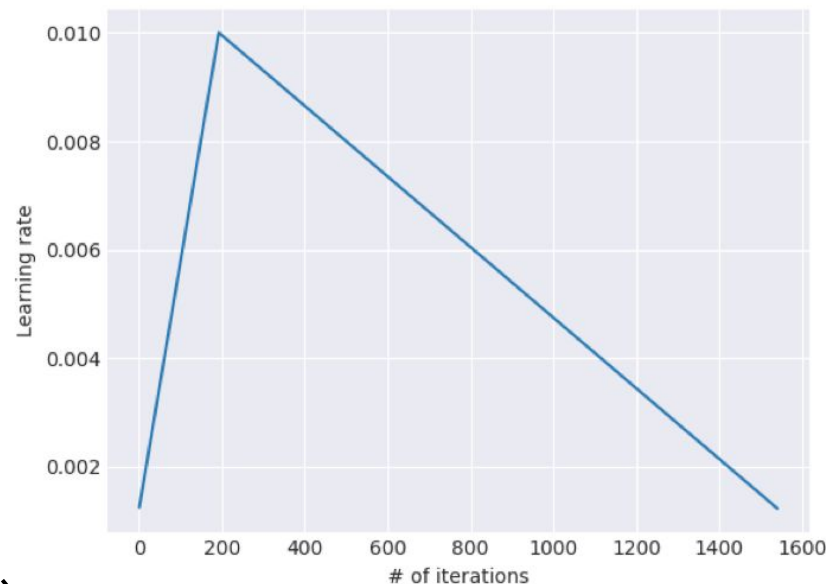
Slanted triangular learning rates (STLR), which first linearly increases the learning rate and then linearly decays it according to the following update schedule.

$$\begin{aligned} cut &= \lfloor T \cdot cut_frac \rfloor \\ p &= \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t-cut}{cut \cdot (1/cut_frac - 1)}, & \text{otherwise} \end{cases} \\ \eta_t &= \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio} \end{aligned}$$

Discriminative fine-tuning

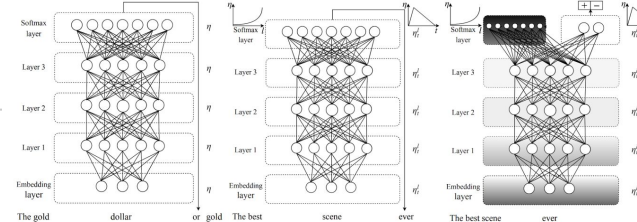
Instead of using the same learning rate for all layers of the model, discriminative fine-tuning allows us to tune **each layer with different learning rates**.

- Layers that are closer to inputs → large learning rate
- Layers that are closer to outputs → small learning rate

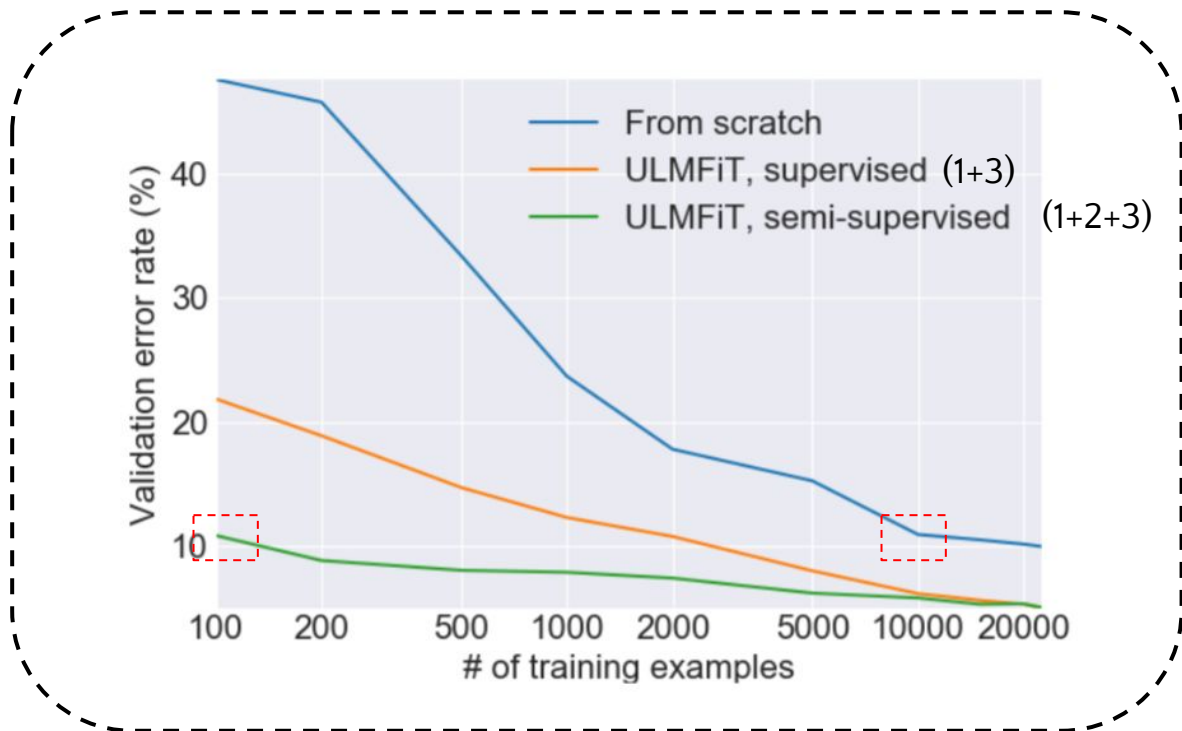




ULMFiT (cont.)

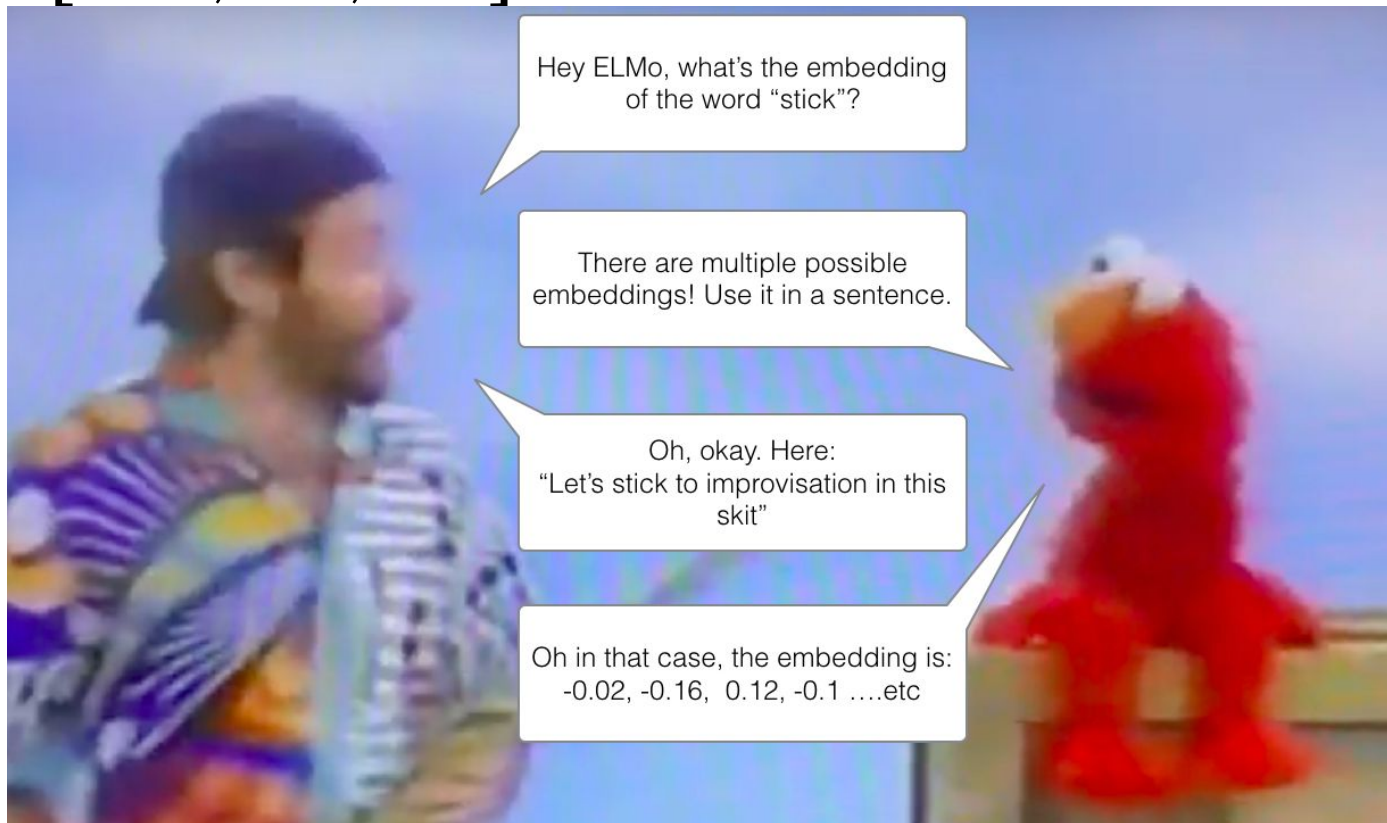


- ULMFiT requires orders of magnitude **less data** than previous approaches.





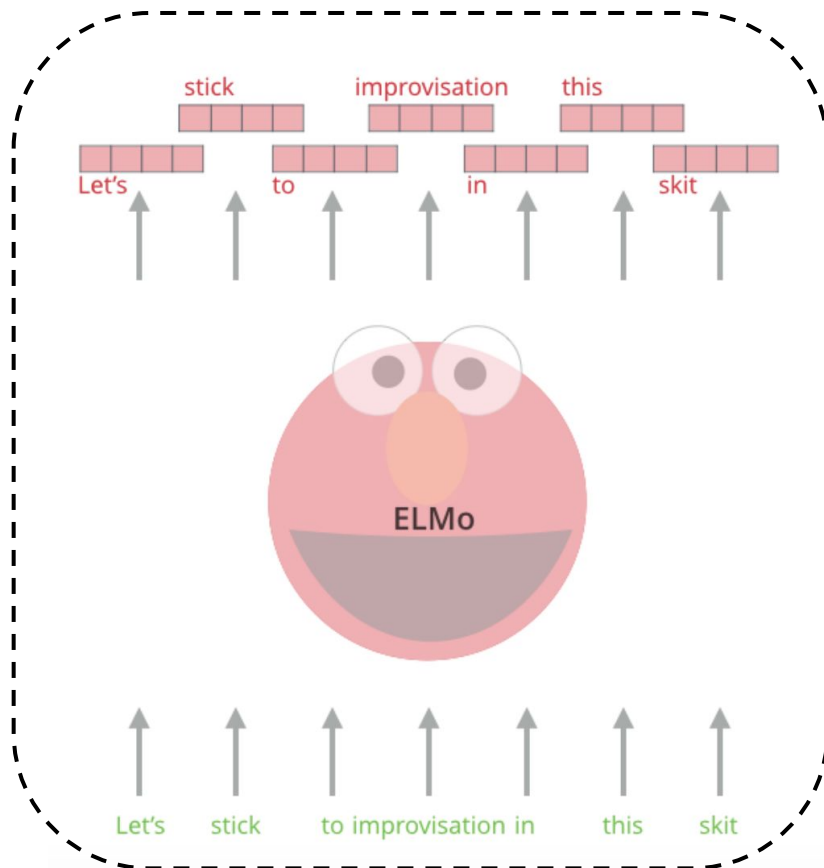
2) ELMo (Embeddings from Language Models) [Peter, et al, 2018]





ELMo (cont.)

- Instead of using a **fixed embedding** for each word, ELMo looks at **the entire sentence** before assigning each word in it an embedding.
- It uses a **LSTM-based bidirectional language model (biLM)** trained on a LM task to be able to create those embeddings.
- ELMo representations are purely **character based**, allowing the network to use morphological clues to form robust representations for out-of-vocabulary (OOV) tokens unseen in training.

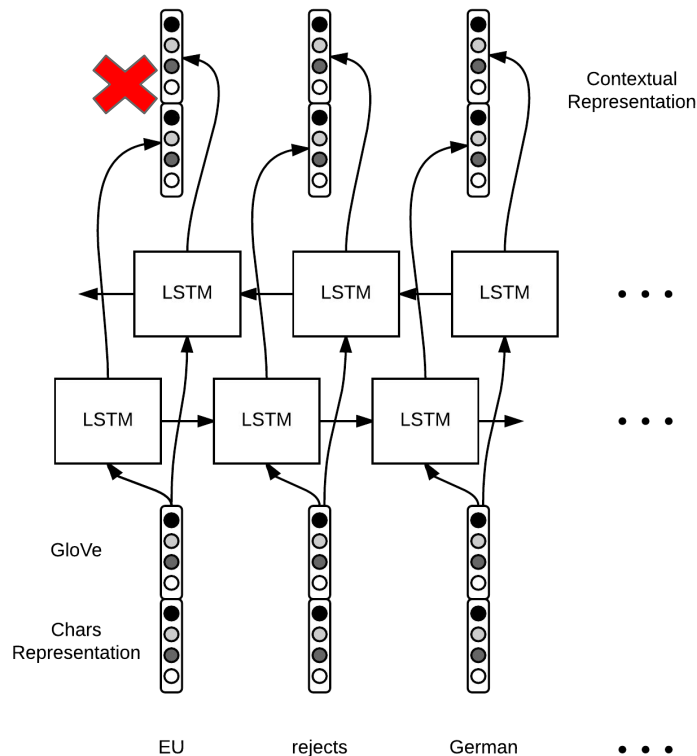




ELMo (cont.)

- Instead of using a **fixed embedding** for each word, ELMo looks at **the entire sentence** before assigning each word in it an embedding.
- It uses **2 separated LSTM-based bidirectional language models (biLMs)** trained on a LM task to be able to create those embeddings.
- ELMo representations are purely **character based**, allowing the network to use morphological clues to form robust representations for out-of-vocabulary (OOV) tokens unseen in training.

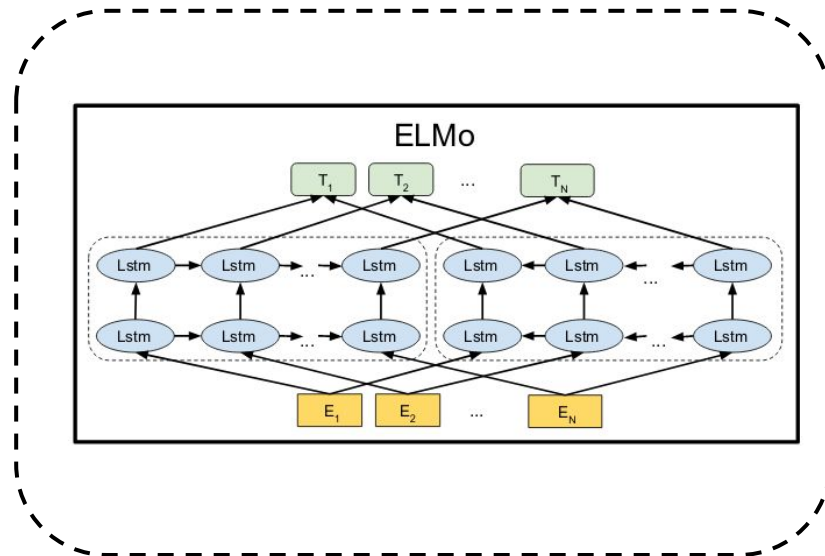
Embedded vector from Bidirectional LSTM is not suitable for LM task since it contains future information. While in the real scenario, we can't see future word, so we cannot embed it.





ELMo (cont.)

- Instead of using a **fixed embedding** for each word, ELMo looks at **the entire sentence** before assigning each word in it an embedding.
- It uses **2 separated LSTM-based bidirectional language models (biLMs)** trained on a LM task to be able to create those embeddings.
- ELMo representations are purely **character based**, allowing the network to use morphological clues to form robust representations for out-of-vocabulary (OOV) tokens unseen in training.





ELMo (cont.)

- Instead of using a **fixed embedding** for each word, ELMo looks at **the entire sentence** before assigning each word in it an embedding.
- It uses **2 separated LSTM-based bidirectional language models (biLMs)** trained on a LM task to be able to create those embeddings.
 - Not concat vector
 - Combine losses from 2 unidirectional LMs for backpropagation
- ELMo representations are purely **character based**, allowing the network to use morphological clues to form robust representations for out-of-vocabulary (OOV) tokens unseen in training.

General Bidirectional Language Model

Forward LM

$$p(t_1, t_2, \dots, t_N) = \prod_{k=1}^N p(t_k | t_1, t_2, \dots, t_{k-1})$$

Backward LM

$$p(t_1, t_2, \dots, t_N) = \prod_{k=1}^N p(t_k | t_{k+1}, t_{k+2}, \dots, t_N)$$

Joint Maximum Likelihood

$$\sum_{k=1}^N (\log p(t_k | t_1, \dots, t_{k-1}; \Theta_x, \vec{\Theta}_{LSTM}, \Theta_s) \\ + \log p(t_k | t_{k+1}, \dots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s))$$



ELMo (cont.)

- Instead of using a **fixed embedding** for each word, ELMo looks at **the entire sentence** before assigning each word in it an embedding.
- It uses a **LSTM-based bidirectional language model (biLM)** trained on a LM task to be able to create those embeddings.
- ELMo representations are purely **character based**, allowing the network to use morphological clues to form robust representations for out-of-vocabulary (OOV) tokens unseen in training.

ELMo Representation

For each token t_k , a L -layer biLM computes a set of $2L+1$ representations. Where \mathbf{x}_k is a context-independent token representation layer and $\mathbf{h}_{k,j}$ is the concatenation forward and backward context-dependent representation for each biLSTM layer.

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \dots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \dots, L\}, \end{aligned}$$

Note: \mathbf{x}_k can be either token representation or a CNN over characters. ELMo uses **a character-based CNN representations** (Peters et al., 2017).



ELMo (cont.)

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \vec{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \dots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \dots, L\}, \end{aligned}$$

A downstream model collapses all layers in R into a single vector: **ELMo**_{*k*}

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^L s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

\mathbf{s}^{task} are softmax-normalized weights (layer-wise weight), a summation of all layers' weight is one.

γ^{task} is a scalar parameters.

Both \mathbf{s}^{task} and γ^{task} are learnable parameters.



ELMo (cont.): Downstream task.

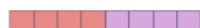
Embedding of “stick” in “Let’s stick to” (word-level task)

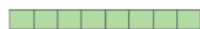
1- Concatenate hidden layers



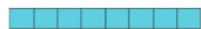
2- Multiply each vector by
a weight based on the task

 $\times S_2$

 $\times S_1$

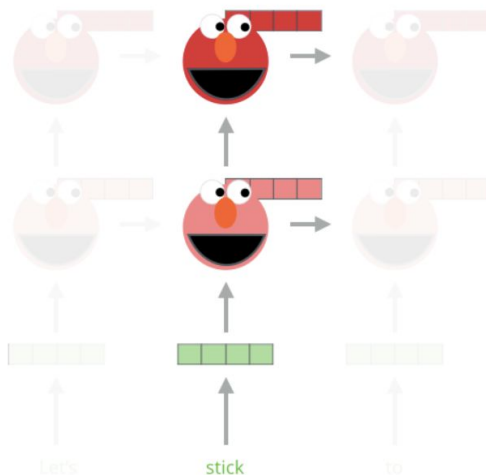
 $\times S_0$

3- Sum the (now weighted)
vectors

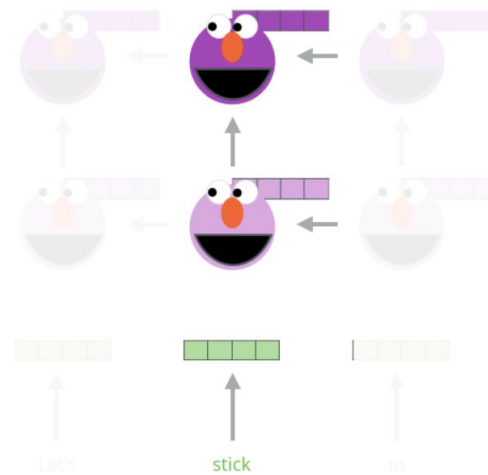


ELMo embedding of “stick” for this task in this context

Forward Language Model



Backward Language Model



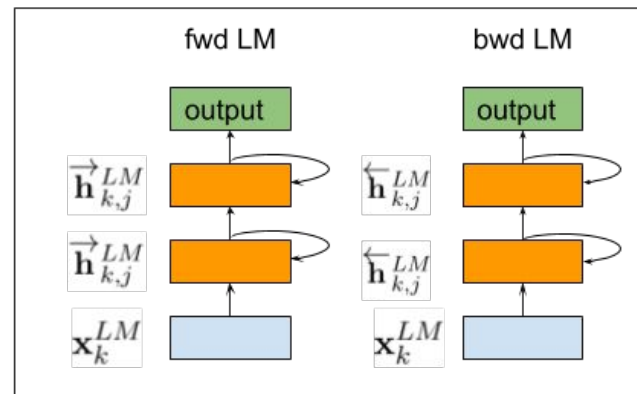


ELMo (cont.)

$$\text{ELMo}_k^{\text{task}} = E(R_k; \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{j=0}^L s_j^{\text{task}} \mathbf{h}_{k,j}^{\text{LM}}.$$

$$\text{ELMo}_k^{\text{task}} = \gamma^{\text{task}} \times \sum \left\{ \begin{array}{l} s_j^{\text{task}} \times \text{[orange box]} \times [\vec{\mathbf{h}}_{k,j}^{\text{LM}}; \overleftarrow{\mathbf{h}}_{k,j}^{\text{LM}}] \\ s_j^{\text{task}} \times \text{[orange box]} \times [\vec{\mathbf{h}}_{k,j}^{\text{LM}}; \overleftarrow{\mathbf{h}}_{k,j}^{\text{LM}}] \\ s_j^{\text{task}} \times \text{[blue box]} \times [\mathbf{x}_k; \mathbf{x}_k] \end{array} \right. \leftarrow$$

biLMs

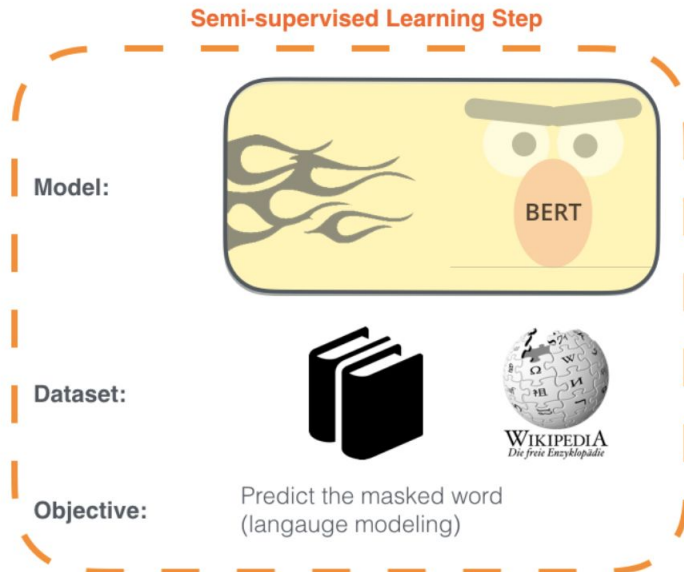




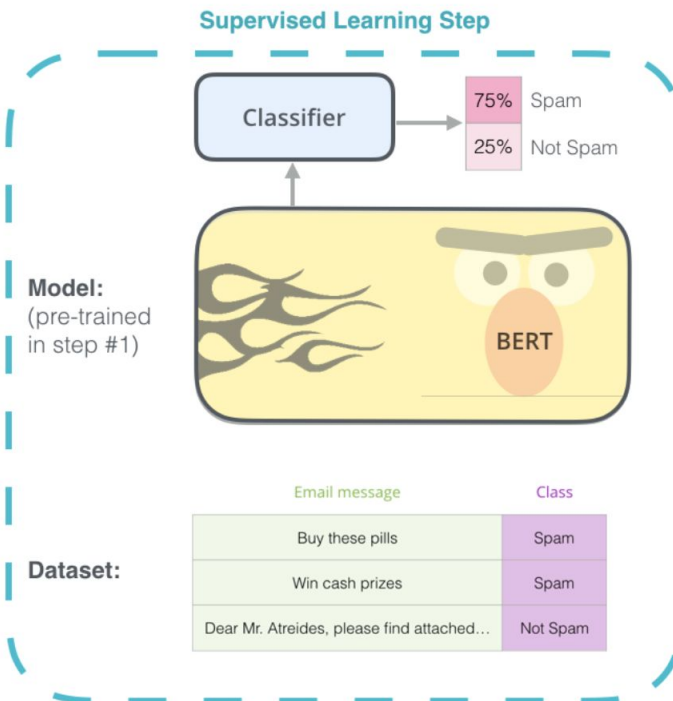
3) BERT [Devlin, et al, 2018]: Bidirectional Encoder Representation from Transformers

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.





BERT (cont.)

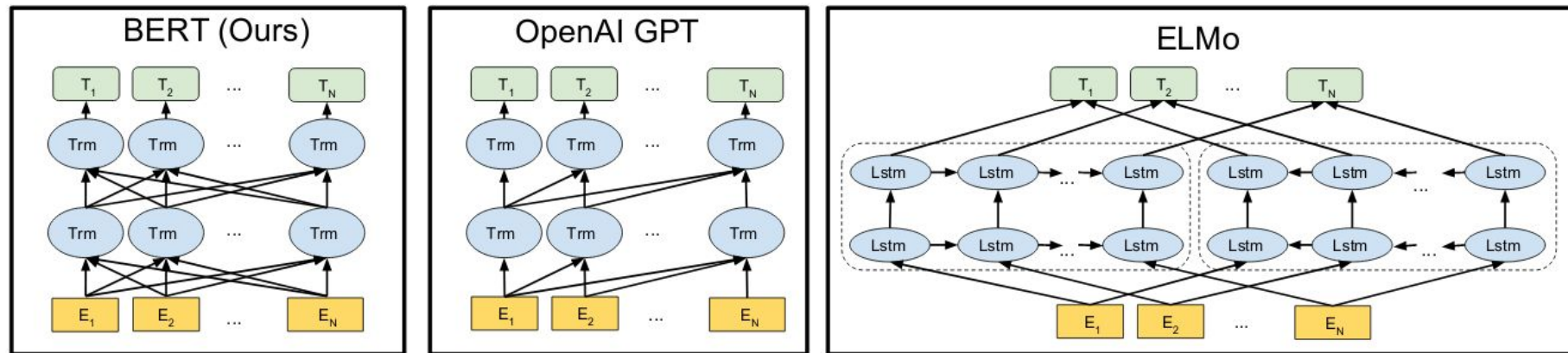


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.



BERT (cont.)

Semi-supervised training, using 2 prediction tasks:

1. Mask language modeling

- represents the word using both its left and right context, so called **deeply bidirectional**.
- Mask out 15% of the words in the input, run the entire sequence through a deep bidirectional encoder, and then predict only the masked words.

```
Input: the man went to the [MASK1] . he bought a [MASK2] of milk.  
Labels: [MASK1] = store; [MASK2] = gallon
```

2. Next sentence prediction

- to learn relationships between sentences.

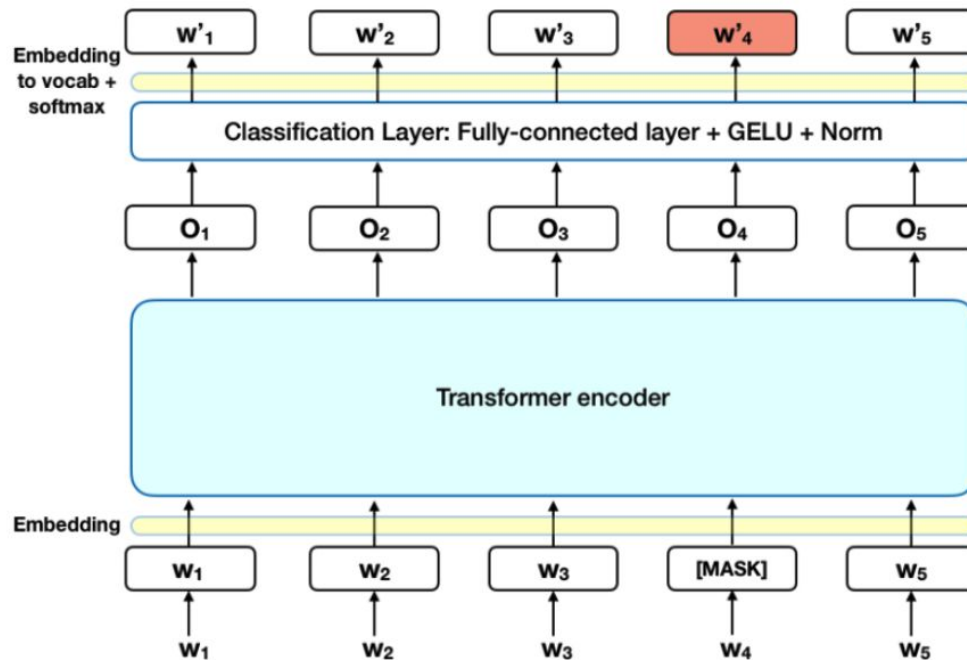
```
Sentence A: the man went to the store .  
Sentence B: he bought a gallon of milk .  
Label: IsNextSentence
```

```
Sentence A: the man went to the store .  
Sentence B: penguins are flightless .  
Label: NotNextSentence
```



BERT (cont.)

Mask language modeling:

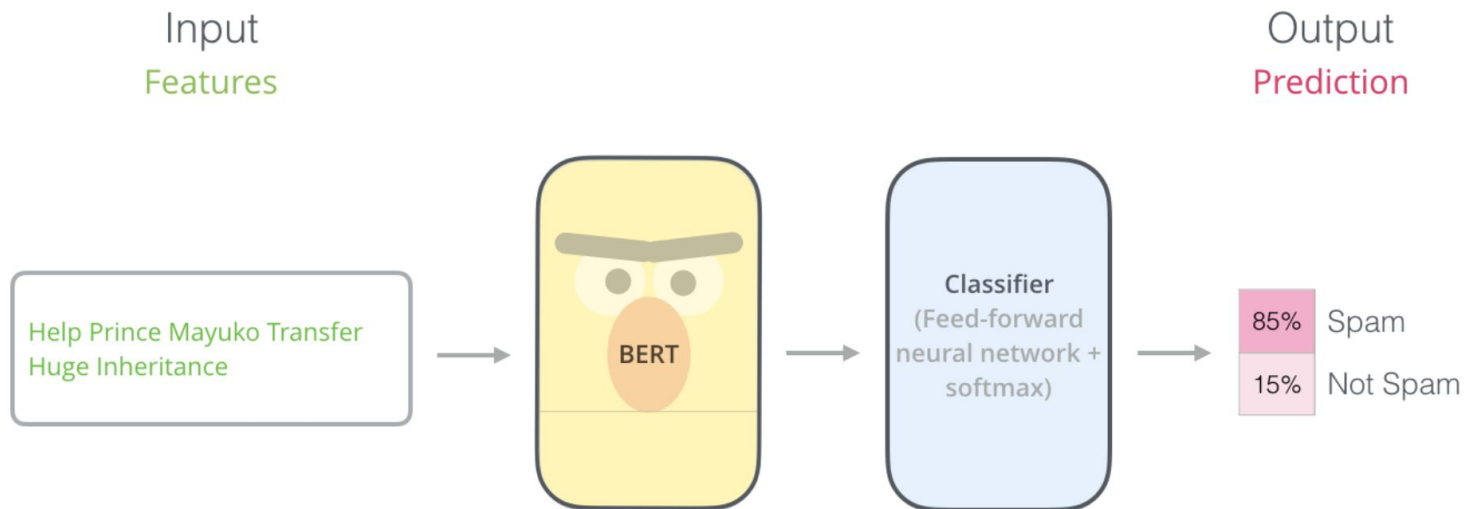




BERT (cont.)

Supervised training (e.g. Email sentence classification)

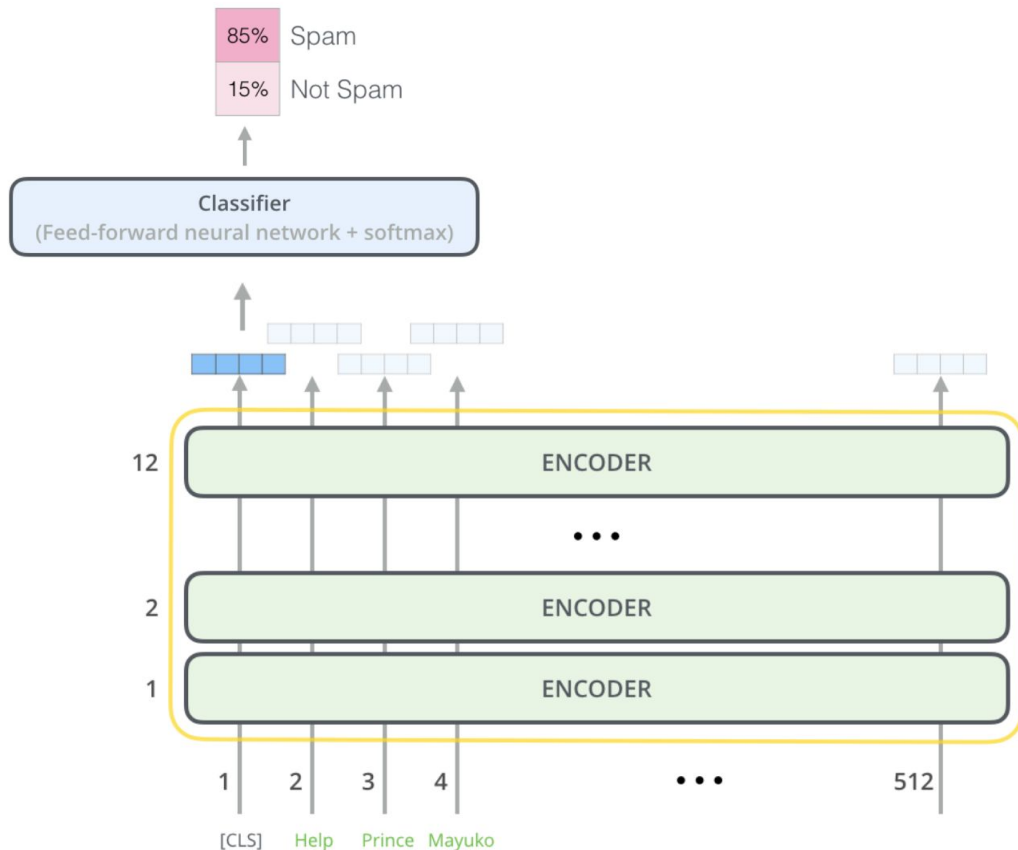
- you mainly have to train the classifier, with minimal changes happening to the pre-trained model during the training phase ([fine-tuning approach](#)).





BERT (cont.)

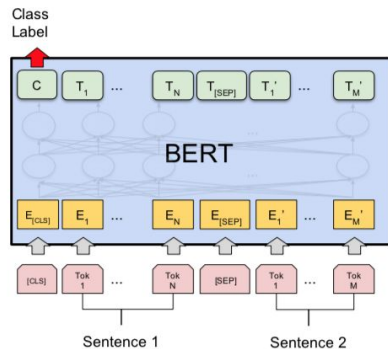
- BERT is basically a trained Transformer Encoder¹ stack.
- The first input token is supplied with a special [CLS] token for classification task to represent the entire sentence.
- The output is generated from only this special [CLS] token.



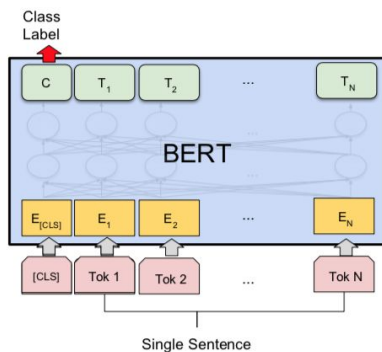
¹ Vaswani et al., **Attention Is All You Need**, NIPS 2017, <https://arxiv.org/pdf/1706.03762.pdf>



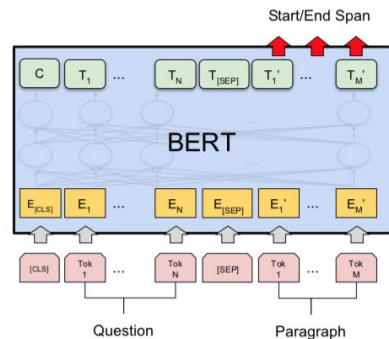
BERT (cont.)



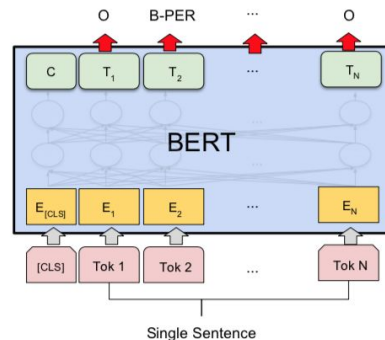
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER



OpenAI GPT (Generative Pre-Training) [Radford, 2018]

GPT training procedure consists of 2 steps:

1. Unsupervised pre-training:

- a. Given an unsupervised corpus of tokens $U = \{u_1, \dots, u_n\}$, use a standard language modeling objective to maximize the following likelihood:

$$L_1(U) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ .

2. Supervised fine-tuning:

- a. We assume a labeled dataset C , where each instance consists of a sequence of input tokens, x_1, \dots, x_m , along with a label y . This gives us the following objective to maximize:

$$L_2(C) = \sum_{(x,y)} \log P(y | x^1, \dots, x^m).$$

- b. Include language modeling as an auxiliary objective to the fine-tuning:

$$L_3(C) = L_2(C) + \lambda * L_1(C)$$

Auxiliary task (LM)

Multitask; add this loss



OpenAI GPT (Generative Pre-Training)

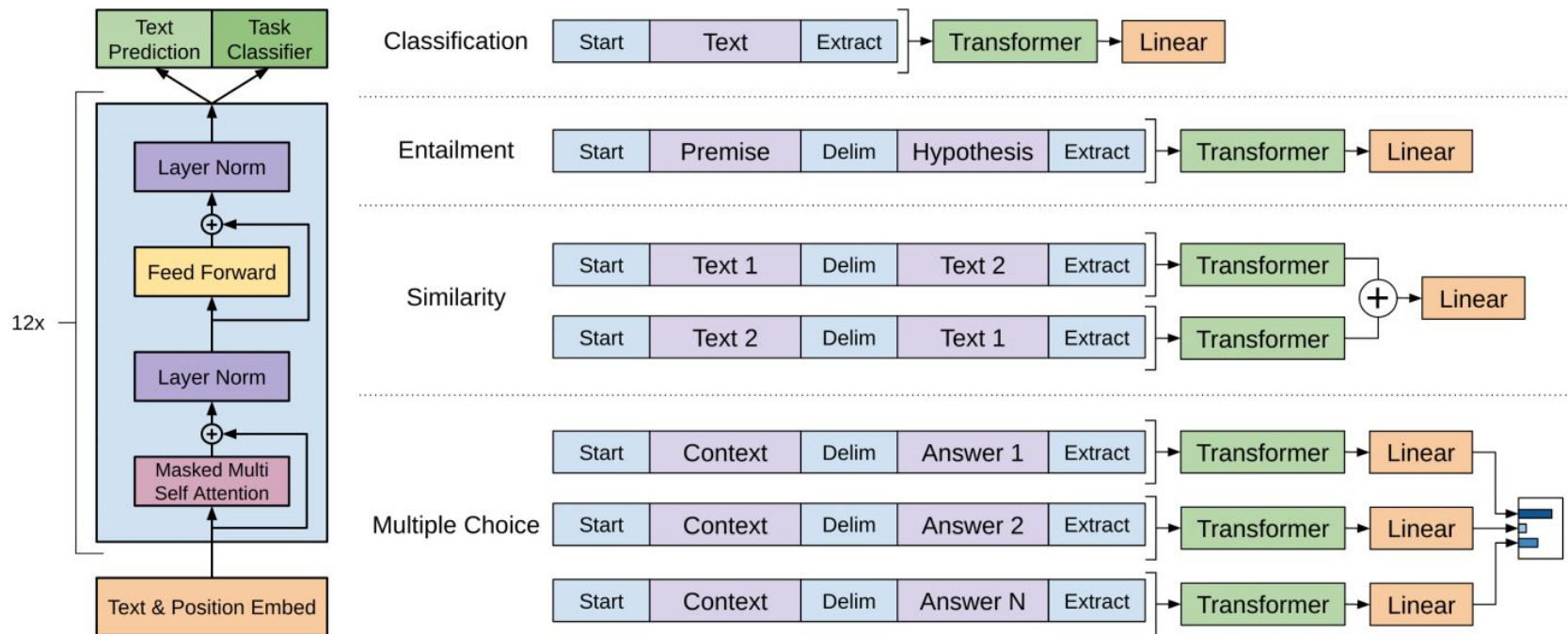
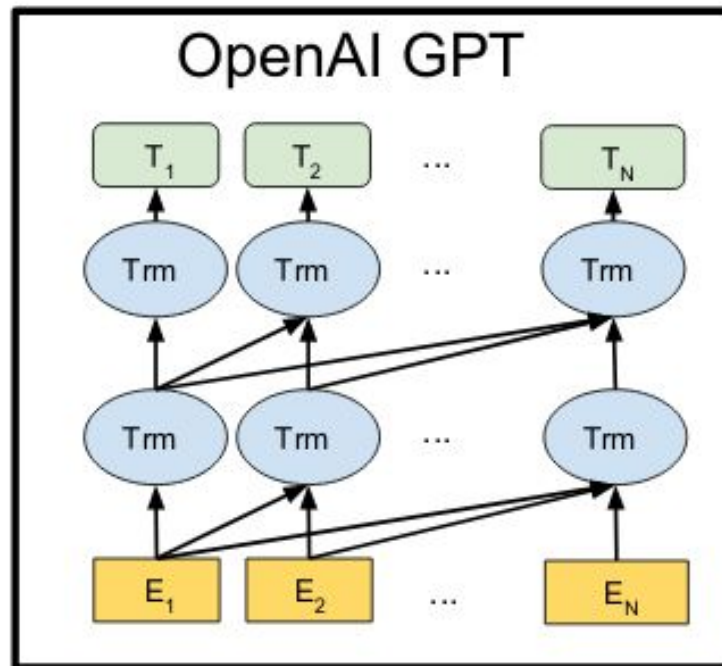
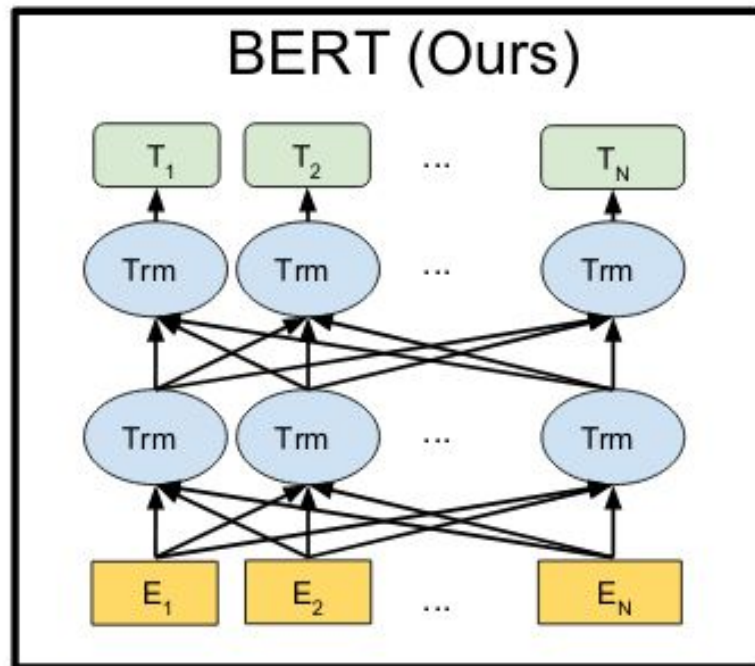


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.



GPT vs BERT





GPT-2 [Radford, 2019]

- GPT-2 is a large transformer-based language model with 1.5 billion parameters
 - GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data.
 - Trained on 8 million webpages (scraped from outbound links from Reddit with at least 3 karma points)
 - State-of-the-art LM
 - On other language tasks like question answering, reading comprehension, summarization, and translation, GPT-2 learns these tasks from the raw text, using no task-specific training data. (zero-shot setting)

VIEW CODE

READ PAPER

Our model, called GPT-2 (a successor to GPT), was trained simply to predict the next word in 40GB of Internet text. Due to our concerns about malicious applications of the technology, we are not releasing the trained model. As an experiment in responsible disclosure, we are instead releasing a much smaller model for researchers to experiment with, as well as a technical paper.

GPT-2 is a large transformer-based language model with 1.5 billion parameters, trained on a dataset^[1] of 8 million web pages. GPT-2 is trained with a simple objective: predict the next word, given all of the previous words within some text. The diversity of the dataset causes this simple goal to contain naturally occurring demonstrations of many tasks across diverse domains. GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data.

<https://openai.com/blog/better-language-models/>



Ekapol Chuangsuwanich ได้นัชรลิ่งก์

22 มีนาคม เวลา 22:53 น.

มีคนเอา GPT2 ตัวที่ปล่อยออกมาไปเล่น ผลสรุปออกมาก็ไม่ได้ดีขนาดนั้น ประสบปัญหาเดิมๆ คือมีโอกาสติดลูปเหมือนโมเดลกneralเก่าๆ

บทความรวมตัวอย่างฮาๆ

[https://arstechnica.com/.../twenty-minutes-into-the-future-.../...](https://arstechnica.com/.../twenty-minutes-into-the-future-.../)

GREAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT-GOAT



ARSTECHNICA.COM

Twenty minutes into the future with OpenAI's Deep Fake Text AI



GPT-2: Example of Generated Texts

SYSTEM PROMPT
(HUMAN-WRITTEN)

Legolas and Gimli advanced on the orcs, raising their weapons with a harrowing war cry.

MODEL
COMPLETION
(MACHINE-
WRITTEN, FIRST
TRY)

The orcs' response was a deafening onslaught of claws, claws, and claws; even Elrond was forced to retreat. "You are in good hands, dwarf," said Gimli, who had been among the first to charge at the orcs; it took only two words before their opponents were reduced to a blood-soaked quagmire, and the dwarf took his first kill of the night. The battle lasted for hours until two of the largest Orcs attempted to overwhelm Aragorn. When they finally stopped, they lay defeated and lifeless for miles and miles.

"I take nothing," said Aragorn. "But I give my word, at my peril and mine, that I will never forget this day of horror. None of us will forget. Ever!"



https://github.com/cstorm125/thai2fit/blob/master/wongnai_cls/classification.ipynb

1274 lines (1273 sloc) | 71.9 KB

<> [icon] Raw Blame History [icon]

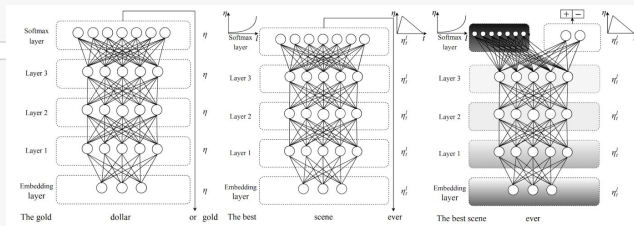
wongnai-corpus Classification Benchmark

We provide two benchmarks for 5-star multi-class classification of [wongnai-corpus: fastText](#) and [ULMFit](#). In both cases, we first finetune the embeddings using all data. The benchmark numbers are based on the test set. Performance metric is the micro-averaged F1 by the test set of [Wongnai Challenge](#).

| model | micro_f1_public | micro_f1_private |
|----------------------|-----------------|------------------|
| ULMFit | 0.59313 | 0.60322 |
| fastText | 0.5145 | 0.5109 |
| LinearSVC | 0.5022 | 0.4976 |
| Kaggle Score | 0.59139 | 0.58139 |
| BERT | 0.56612 | 0.57057 |

```
In [ ]: # #uncomment if you are running from google colab
# !pip install sklearn_crfsuite
# !pip install https://github.com/PyThaiNLP/pythainlp/archive/dev.zip
# !pip install fastai==1.0.45
# !wget https://github.com/wongnai/wongnai-corpus/raw/master/review/review_dataset.zip; unzip review_dataset.zip
# !mkdir wongnai_data; ls
```

```
In [3]: import pandas as pd
import numpy as np
from ast import literal_eval
from tqdm import tqdm_notebook
from collections import Counter
import re
```



Step1) load pretrained model on thai corpus (unsupervised LM)

Step2) Fine-tune on LM task with Wongnai data

Step3) Gradually unfreeze each layer to fine-tune on supervised task (sentiment analysis on Wongnai data)

thai2fit (formerly thai2vec)

ULMFit Language Modeling, Text Feature Extraction and Text Classification in Thai Language. Created as part of [pyThaiNLP](#) with [ULMFit](#) implementation from [fast.ai](#)

Models and word embeddings can also be downloaded via [Dropbox](#).

We pretrained a language model with 60,005 embeddings on [Thai Wikipedia Dump](#) (perplexity of 28.71067) and text classification (micro-averaged F-1 score of 0.60322 on 5-label classification problem. Benchmarked to 0.5109 by [fastText](#) and 0.4976 by LinearSVC on [Wongnai Challenge: Review Rating Prediction](#). The language model can also be used to extract text features for other downstream tasks.

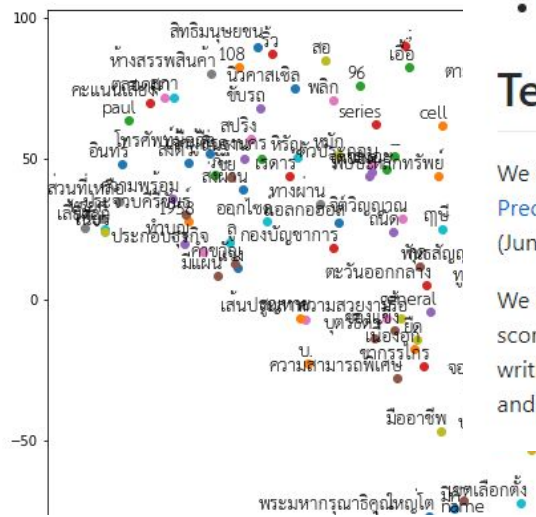
v0.4 (In Progress)

- Replace AWD-LSTM/QRNN with transformers-based models
- Named-entity recognition

Text Classification

We trained the [ULMFit model](#) implemented by [thai2fit](#) for text classification. We use [Wongnai Challenge: Review Rating Prediction](#) as our benchmark as it is the only sizeable and publicly available text classification dataset at the time of writing (June 21, 2018). It has 39,999 reviews for training and validation, and 6,203 reviews for testing.

We achieved validation perplexity at 35.75113 and validation micro F1 score at 0.598 for five-label classification. Micro F1 scores for public and private leaderboards are 0.59313 and 0.60322 respectively, which are state-of-the-art as of the time of writing (February 27, 2019). [FastText](#) benchmark based on their own [pretrained embeddings](#) has the performance of 0.50483 and 0.49366 for public and private leaderboards respectively. See [ulmfit_wongnai.ipynb](#) for more details.



3

Conclusion

Transfer learning conclusion



- Low-resource/small training data is a common issue especially in NLP.
- Possible to train large models on **small labeled data** by using transfer learning and domain adaptation.
- **Off the shelf features** (fixed weights) work very well in various domains and tasks for smaller samples, **but should be adapted to increase performance (fine tune)**.
- **Lower layers** of network contain very generic features, **higher layers** more task specific features. Tasks that are very different are harder to transfer.
- **Supervised domain adaptation via fine tuning almost always improves performance.**

NLP conclusion

- In NLP, LM is a common unsupervised task, and Wiki corpus is a common corpus for unsupervised learning. Analogy, LM = image classification & Wiki = ImageNet.
- There are 4 recent pretrained LM's: ULMFiT, ELMO, BERT, GPT where BERT is the SoTA. There are two main phases: (1) unsupervised on LM task [pretrained and available for download] and (2) supervised on downstream task [transferred model for downstream task (fine-tune & inference)].
- ULMFiT comprises of one embedding layer and three LSTM layers, where each layer will be gradually unfreezed in a top-down fashion during the fine-tuning phrase,.
- ELMO employs bidirectional LMs (separated forward and backward LMs) and a CNN-over-character embedding to create a word embedding vector -- this alleviates OOV issue.
- BERT is based on Transformer (attention) with two unsupervised objectives: (1) mask language model (word-level embedding vector) & (2) next sentence prediction (sentence embedding vector).
- GPT is an unidirectional transformer-based model. LM is also used as an auxiliary objective during supervised training phrase.

References



- K. McGuinness, Transfer Learning (D2L4 Insight@DCU Machine Learning Workshop 2017)
- <https://medium.com/dair-ai/a-light-introduction-to-transfer-learning-for-nlp-3e2cb56b48c8>
- <http://jalammar.github.io/illustrated-bert/>