Deep Transfer Learning in NLP

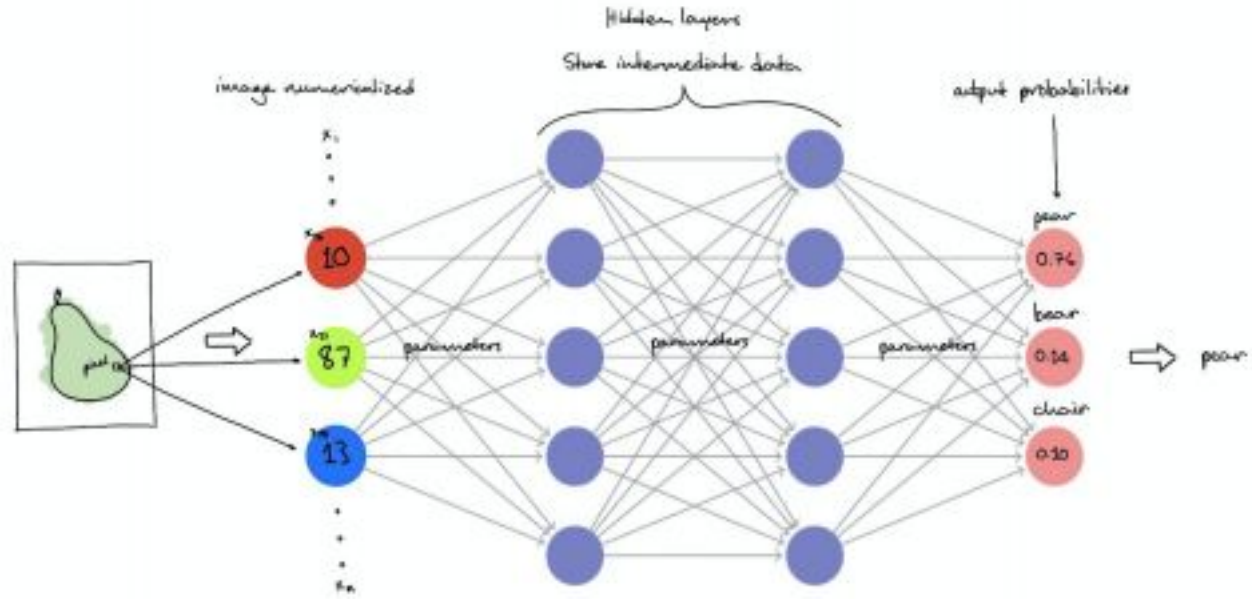
How can we Improve our natural language processing models using transfer learning?

Vance Degen
McDonogh High School
10/2019

What is deep learning?

Deep learning is a subset of artificial intelligence involving specialized machine learning algorithms that process numericalized data through several layers of parameters in order to produce an output. Essentially, the model's output is a function of the input data.

For example, suppose we have a classification model that can differentiate pears, bears, and chairs. When we show the model an image, it converts each pixel into three numbers based on the amount of red, green, and blue it holds. Then, several mathematical transformations convert the hundreds of input numbers into three final probabilities - each corresponding to either "pear", "bear", or "chair". If we show the model an image of a pear, the [probability image is pear] would be greater than P(image is bear) or P(image is chair), and we would be returned the label "pear".



structure of a basic neural network (simplified)

How do models learn?

When it's first created, our pears/bears/chairs model classifies with an average accuracy of 33% - pure guesswork. This is because the parameters haven't been properly initialized to do the correct transformations of the input data, so our model's just a huge random number generator at this point. However, if we put the model into "training mode", it can improve its accuracy by tweaking its own parameters - and all we need to do is show it labeled data! But what's actually happening behind the scenes? As we know, every time we show the model an image, three probabilities will be outputted. In training mode, we employ a "loss function" that measures the variance of the expected output from the actual output each time an image is passed through (expected output will always have a 1.0 in the node with the same name as the image's label, and 0.0 in the other two). Each time a loss is calculated, the model will tweak its parameters accordingly. In our example, when we showed the model an image of a pear, the nodes for pear, bear, and chair respectively lit up with the probabilities 0.76, 0.14, and 0.10. When we show the model another image of a pear after some training, these numbers would hopefully be closer to 1.0, 0, and 0. Essentially, the loss is a function of the parameters, and the goal of training is to find the parameters that minimize the loss. This is accomplished through gradient descent.

Knowing all this, it's intuitive that the more data we use for training, the more

chances we have to tweak the network parameters, and the higher our model's accuracy will be. But does that always have to be the case?

What is transfer learning?

"Transfer learning" is a relatively new technique in the field of deep learning by which one model that's trained to do a particular task is reused and fine-tuned to do another related task. Instead of beginning with an empty architecture, we start with a model with almost-correctly initialized parameters that only need minor tweaking.

Why use it?

Compared to empty architectures, pre-trained models require less data to train, have faster compute times, and generally lead to more robust final models. These differences become more and more consequential as we move into the realm of big data. From a more existential standpoint, some experts say that transfer learning is the key to creating a general artificial intelligence - That is, the next major advancement in civilization foretold by many a sci-fi novel.

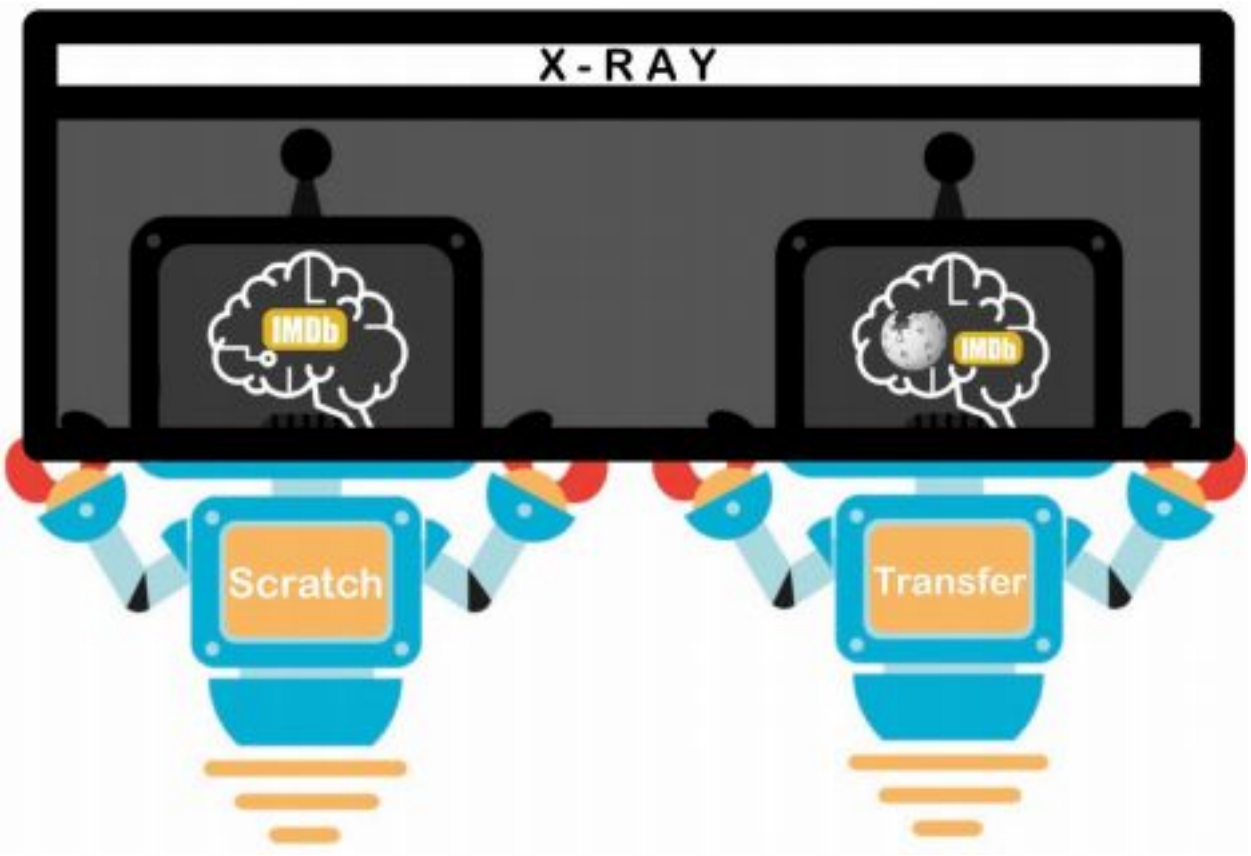
This project

While transfer learning has been long accepted as superior to training from scratch for image recognition models, like our pears/bears/chairs classifier, the technique has only recently gained traction in deep natural language processing. NLP has to do with how

artificial intelligence analyzes linguistic data in a way that imitates a human-like understanding of language. This project aims to explore how transfer learning compares to training from scratch when applied to NLP models. Specifically, the models I compare are ones that perform "sentiment analysis" - a branch of NLP concerned with

determining the attitude a piece of text takes toward its subject. Their task is to correctly classify thousands of IMDb movie reviews as either positive or negative. In order for a model to be able to understand English semantics, we must train it on text files. In this project, all models are trained on the IMDb movie review target corpus;

the only difference, in terms of transfer learning, is that half of the models start as copies of ULMFiT - the groundbreaking open-sourced base model trained on WikiText-103. The other half start as completely empty architectures. Hypothetically, the pre-trained models already have a good understanding of English from extensive training on such a large corpus. The IMDb training dataset serves to fine-tune these models to recognize special vocabulary, such as slang or names of actors. On the other hand, the empty models must learn English from the ground-up solely from the IMDb dataset.



professional artistic rendering of models from ULMFiT and scratch

For both pre-trained and empty models, I will plot accuracy against dataset size in order to observe how each type performs in situations where data is limited.

The models

Besides the amount of pretraining, another important variable that affects a model's accuracy is the size of its training dataset. Lacking proper data is often a problem when training models. By creating training datasets of different sizes, I can simulate either an abundance or lack of data, and compare the performances of pretrained and

empty models in each situation. The dataset sizes are as follows: 500, 1k, 2.5k, 5k, 10k, and 20k texts. In the end, I have twelve models, trained by various methods, whose accuracies I can compare.

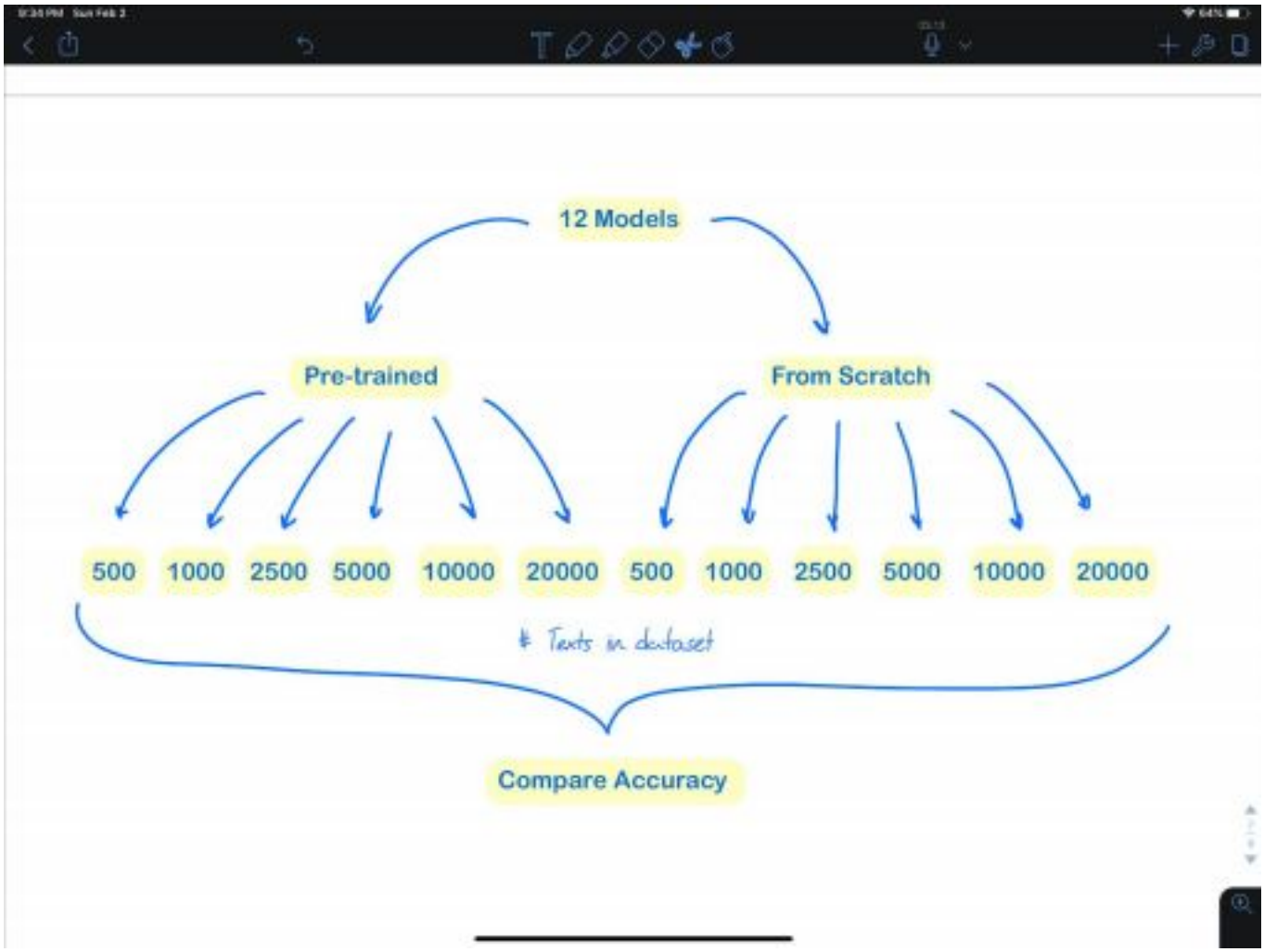
Variables

Explanatory variables:

- Whether the model uses transfer learning
- Dataset size
    - Levels: 500, 1000, 2500, 5000, 10000, or 20000 texts

Response variable:

- Model accuracy



Data Collection Process

Below are screenshots of two cells from my Google Collab notebook. In each, I call a function that takes a dataset and a base model as parameters, and uses them to train a classifier model. These cells display the analytics for the scratch model trained on

500 texts, and the transfer model trained on 1000 texts respectively. It should be noted that I don't necessarily record the highest accuracy I see, as it may have been achieved from model overfitting. This is a way for a model to be weak, but still display high accuracies.

```
[ ] trainClassifier('scratch_enc500',data_clas500, 1, 1, 1, 2)
```

epoch	train_loss	valid_loss	accuracy	time
0	0.755694	0.688892	0.517787	00:04
epoch	train_loss	valid_loss	accuracy	time
0	0.656314	0.689873	0.541502	00:04
epoch	train_loss	valid_loss	accuracy	time
0	0.507225	0.693270	0.549407	00:06
epoch	train_loss	valid_loss	accuracy	time
0	0.597928	0.677284	0.577075	00:07
1	0.504441	0.643215	0.608696	00:07

```
[ ] trainClassifier('enc1000_transfer',data_clas1000,1,1,1,2)
```

epoch	train_loss	valid_loss	accuracy	time
0	0.622859	0.630730	0.650000	00:08
epoch	train_loss	valid_loss	accuracy	time
0	0.588805	0.527663	0.752083	00:09
epoch	train_loss	valid_loss	accuracy	time
0	0.446606	0.452724	0.795833	00:12
epoch	train_loss	valid_loss	accuracy	time
0	0.543174	0.665939	0.564583	00:14
1	0.386157	0.390115	0.825000	00:16

The Data

Classifications
ULMFiT

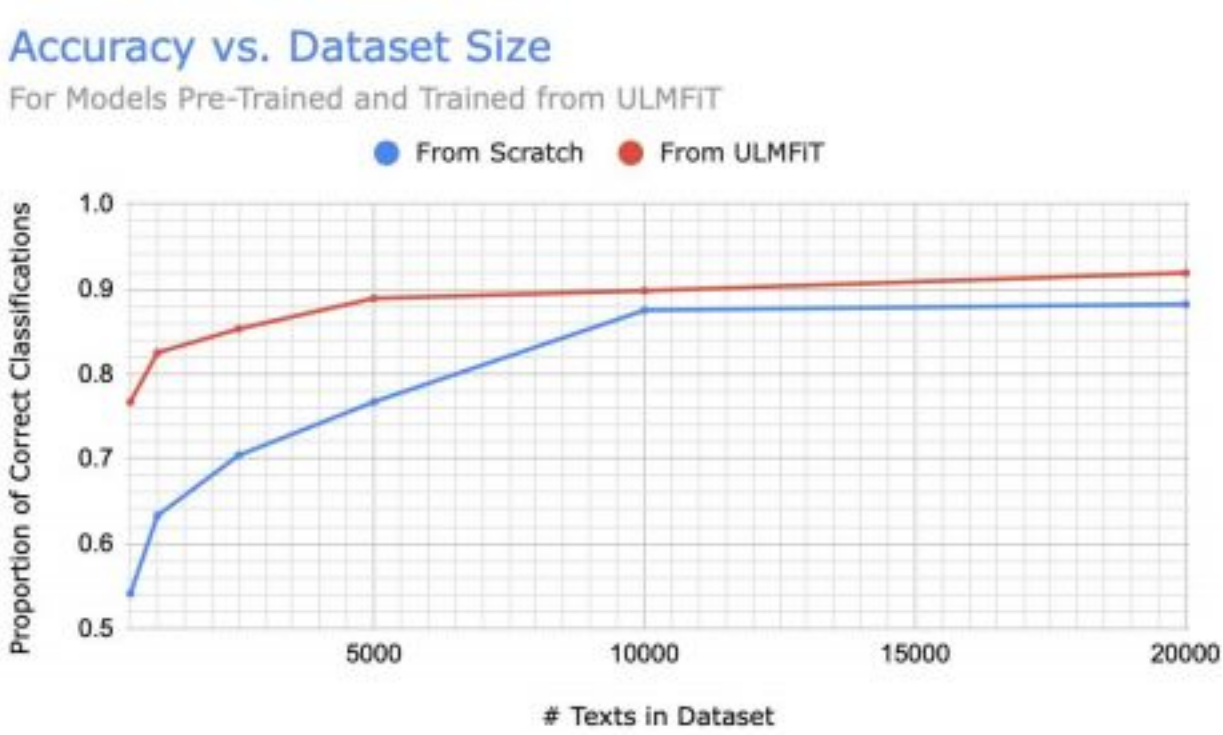
Texts in Dataset Scratch Proportion of Correct Proportion of Correct Classifications

500 0.541 0.767 1000 0.633 0.825 2500 0.704 0.853

5000 0.767 0.889

10000 0.875 0.898

20000 0.882 0.919



Conclusion

As the graph suggests, Transfer learning will consistently yield a higher accuracy than training from scratch. It's interesting to note that the difference between the accuracies starts out very large, at about 22%, but shrinks as more training data becomes available. This behavior can be taken advantage of when you're trying to train a model for a task that's so unprecedented, there's barely any training data available. In that kind of situation, try to find any open-source model whose task is even vaguely related that could serve as a base. This conclusion isn't as intuitive as this paper would
have you think; I can say from firsthand experience that a surprising amount of long-time AI practitioners are only somewhat familiar with the technique, but not aware of its full versatility or effectiveness. It only recently became prolific in NLP because a coder decided to test it on a whim, not predicting that he would propel the field into a Golden Era.

In some ways, this project could also be seen as a mini proof-of-concept for new

techniques to preserve privacy. As it stands, privacy concerns are the biggest, if not the only socially-imposed impediment to technological advancement. While deep learning is the new state-of-the-art, industries are kept from using it to its full potential because of the fact that that would entail using sensitive consumer data. However, new techniques are arising that could potentially solve this issue, with a few of them centered on transfer learning. Take, for example, the idea of edge computing. For this technique, we would download a base model to a mini computer that is completely disconnected from any cloud or WiFi. Then, this privatized model could safely be trained and fine-tuned on more sensitive data without risk of it being hacked into or stolen. In my opinion, it's only a matter of time before the full implications of transfer learning start making waves on the industry level.