CS 410: Text Information Systems

Tech Review
Transfer Learning for Text Classification
November 5, 2020

Mattias Rumpf
mrumpf2@illinois.eu

1. What is transfer learning?

Transfer learning in simple terms means acquiring knowledge from a source task and applying this knowledge to a different (target) task (see Figure 1).

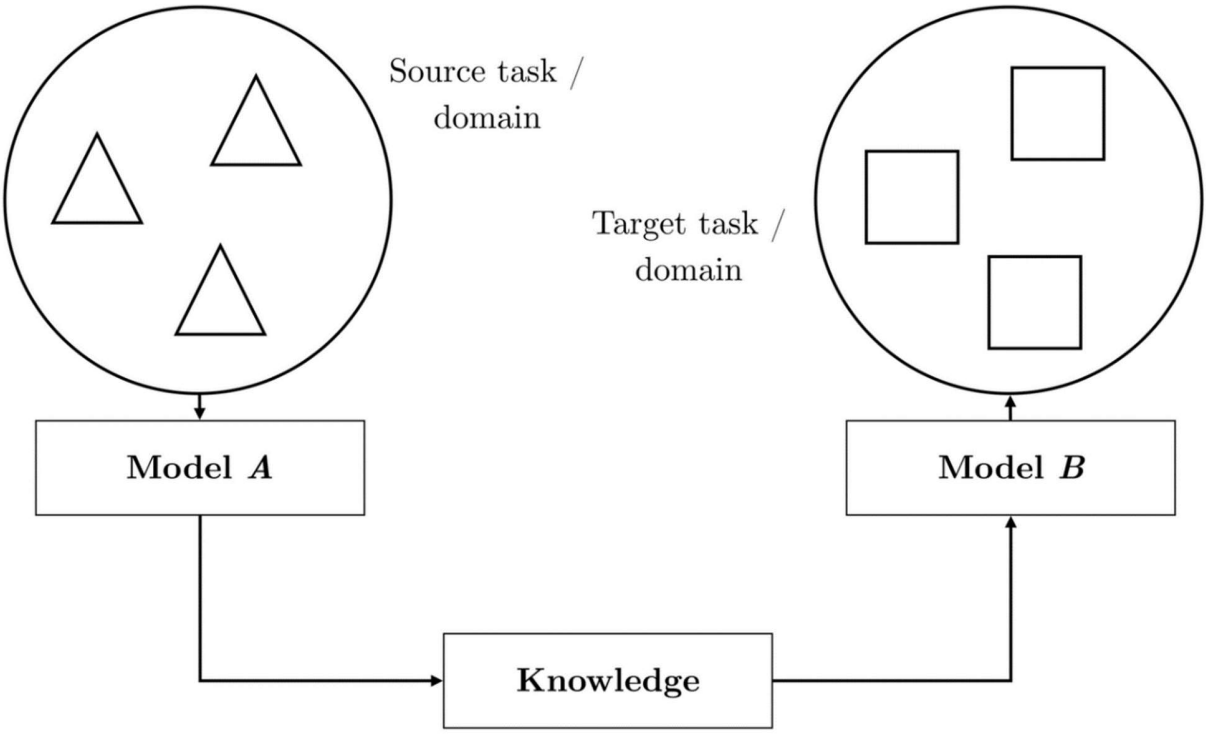

Figure 1: The process of transfer learning [1]

2. How does it generally work for text classification?

Sebastian Ruder [2] categorizes transfer learning for NLP over three dimensions, a) whether the source and target task are the same or different, b) whether learning transfers knowledge between different domains or different languages, or c) whether different tasks are learned simultaneously or sequentially (see Figure 2).

General idea

In this tech review, I discuss the most common branch for text classification: Inductive and sequential transfer learning. We could use sequential transfer learning for text classification by pre-training a language model *unsupervised* (or rather *self-supervised,* using pre-training tasks like next word prediction) on a large corpus of text data and fine-tuning the model *supervised learning* on labeled text.[1]

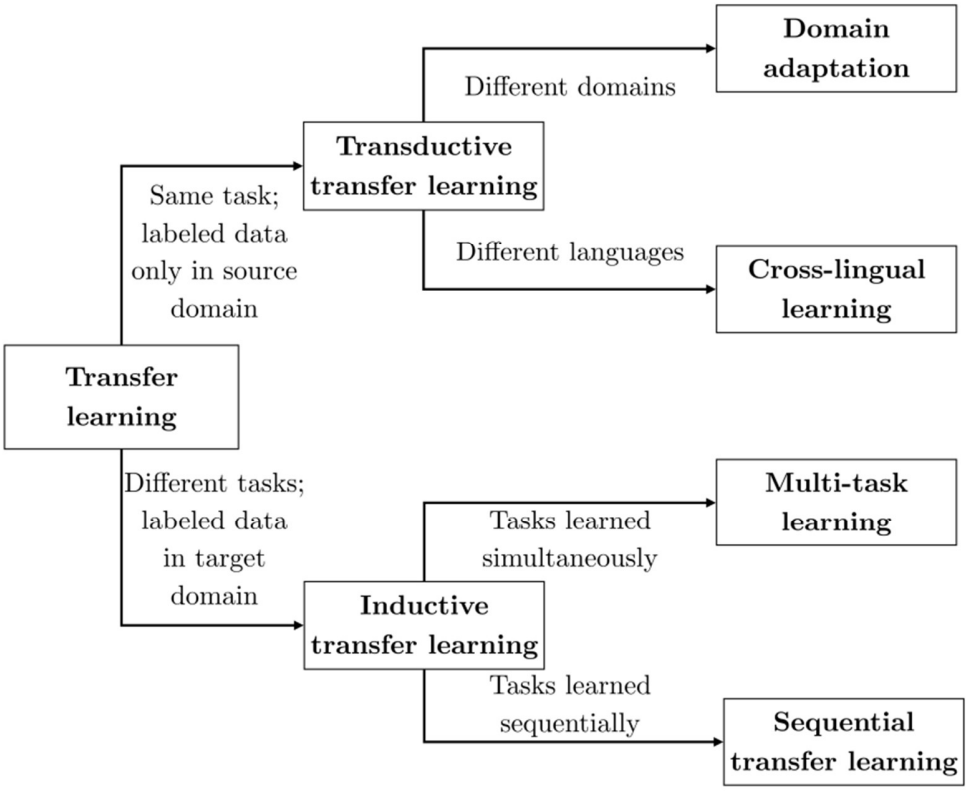

Figure 2: A taxonomy of transfer learning [2]

[1] Chuong Do and Andrew Ng [3] define *transductive transfer learning* on the classification algorithm as follows. They argue that many text classification algorithms such as TFIDF variants result from a tedious, manual try and error approach in proposing and testing different functional forms and corresponding parameter values for a parameter function over text statistics (such as term and document frequency). They propose automatically learning the functional form and parameter values from related classification (source) tasks and applying the learned parameter function to the target task.

What is a language model?

A language model is a probability distribution over sequences of words [4]. This distribution can be very simple, treating each word as independent from all other words in the vocabulary and especially independent from co-occurring neighboring words. In this case, called the *Unigram model*, the language model could simply represent the individual frequencies of all the words appearing in the corpus of text data on which we train the model. The probability of a sequence of words is hence simply the product of the probabilities of the words that form the sequence.

We can extend the complexity of the language model in many ways. For example, by considering not only single words, but by forming probability distributions over terms consisting of 2, 3, or n words, a representation called *n-gram model*. We could introduce further complexity by considering the impact of a word's context on its probability. *Bidirectional models* consider the probability of a word conditional on both the pre- and post-context. Even larger models consider context at the level of sentences.

Usually language models are thus representations in which words or phrases are mapped into vectors of real numbers [5], so called *word embeddings*. In case of a unigram model, we simply have a vector with a length equal to the number of unique words in the vocabulary and with the vector's elements representing each word's probability, i.e. relative frequency. Obviously, a simple word embedding for n-grams plus context is only feasible as long as the window for context and the number of terms in the n-grams remain small. At some point, calculating probabilities for any possible combination of words and context simply is too costly.

Neural language models, i.e. language models based on neural networks, take raw word vectors or embeddings as inputs and are trained on practical tasks for language modeling to gain knowledge about the semantic properties of the source text. For example, tasks used for training BERT (see below) are the prediction of randomly masked words (Masked Language Model (MLM)) and predicting if two pairs of sentences are following each other in the original text (Next Sentence Prediction (NSP)) [6]. The resulting language model hence consists of the network architecture, i.e. its layers, neurons, interactions, activation functions and the corresponding trained weights optimized and learned in a *self-supervised* manner on the training task(s), most commonly next word prediction.

How does fine-tuning work for neural language models?

Three common ways to fine-tune a neural language model: First, we could adjust weights on all neurons and adapt the entire architecture for example by back-propagation. Second, we could freeze some layers and only allow certain layers to be adjusted during fine-tuning. Third, we could freeze the entire architecture and add new layers to the existing network which are trained on the target data [6]. Off course a combination of these three approaches is possible, and different models and frameworks employ various algorithms to achieve proper adjustment of the weights in a neural network.

3. What are the state-of-the-art transfer learning models/frameworks? How do they differ?[2]

State-of-the-art transfer learning models for NLP vary over several dimensions: The neural network architecture, the training tasks and the data used. And finally, in application, these models also differ in terms of fine-tuning to the target task. I will very briefly mention some of the most seminal models for text classification and their most important features. Unfortunately, the scope of this tech review does not allow to go into the machine room to investigate all the nuts and bolts of these models. Malte and Ratadiya [8] provide a very useful overview to put the models in the context of recent developments in NLP transfer learning, see Figure 3. And I will provide further readings and tutorials for interested readers at the end of this review.

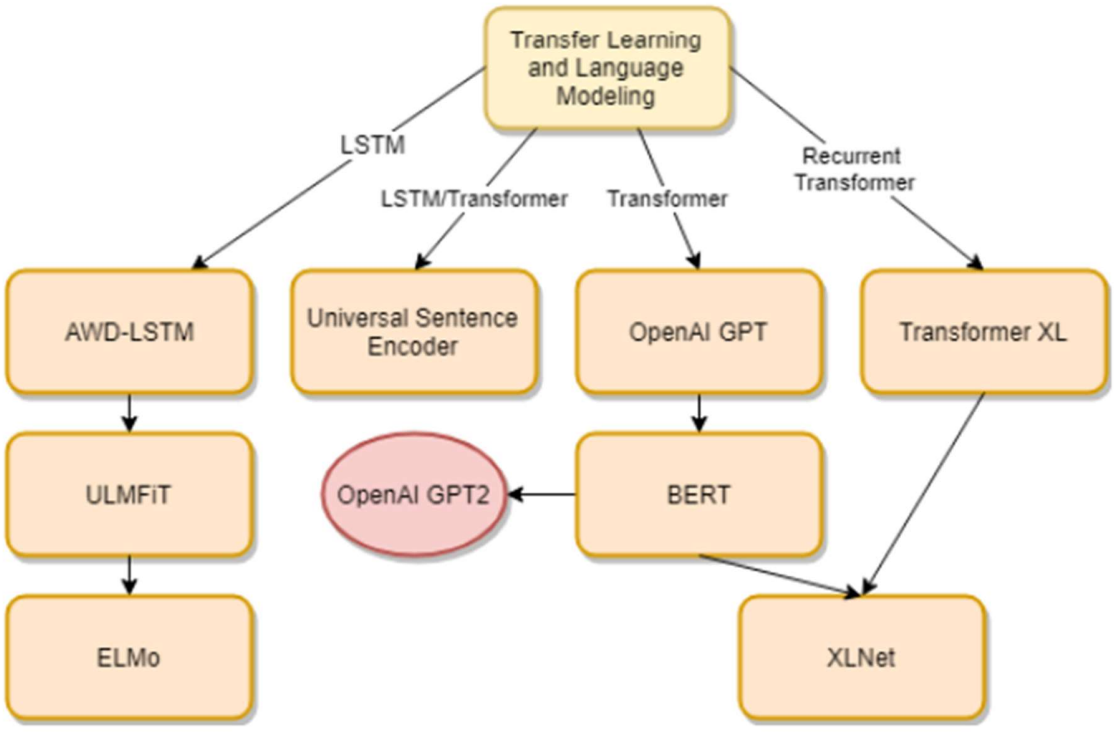

Figure 3: Developments in Transfer Learning and Language Modeling

[2] See also https://medium.com/@gauravghati/comparison-between-bert-gpt-2-and-elmo-9ad140cd1cda

ULMFit[3]

ULMFit is not necessarily state of the art anymore but it introduced several fine-tuning algorithms that retain knowledge and prevent "catastrophic forgetting" in the pre-trained model. The authors use discriminative fine-tuning to vary fine-tuning over network layers, slanted triangular learning rates, and an adaptive learning rate technique. After fine-tuning the language model on the target task data, the model is concatenated with two more blocks for classifier fine tuning and the entire model is fine-tuned using gradual unfreezing starting from the layer that contains the least general knowledge.

The pre-trained model is an ASGD Weight-Dropped Long Short-Term Memory Model (AWD-LSTM).[4] The original paper on ULMFit does not state special training tasks so I assume that they used the standard task of predicting the next word in a sentence.[5] The model is pretrained on Wikitext-103 consisting of 28,595 preprocessed Wikipedia articles and 103 million words.

ELMo[6]

I want to mention ELMo because the model is special in the sense that it uses "Embeddings from Language Models", hence the acronym. The embeddings are obtained by training a deep bidirectional LSTM on a language model objective (again next word prediction, I presume). With this approach, ELMo is able to learn syntax and semantics of words and its variation across context. This means that the vector assigned by ELMo to a word or term is a function of the entire sentence. This makes ELMo different from traditional word embeddings such as word2vec. ELMo represents the same word by a different vector if it appeared in a different context, i.e. sentence. [9]

BERT[7]

BERT established itself (at least for some time) as the go-to model for a variety of NLP tasks and applications. The research team behind BERT describes the model as follows [7]: *"BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks."*

Bidirectional means it is designed to learn context from the right and left of a word or term. Furthermore BERT is pre-training is based on a very "large corpus of unlabelled text including the entire Wikipedia (that's 2,500 million words!) and Book Corpus (800 million words)." [7]

Put simply BERT entails a fast body of knowledge on language semantic and syntax, and it is fairly easy to tune it to perform well on the target task (by simply adding one additional output layer, as stated above).

---

[3] See the original paper at https://arxiv.org/pdf/1801.06146.pdf
[4] See the original paper at https://arxiv.org/pdf/1708.02182.pdf
[5] See, for example, https://dimensionless.in/building-a-neural-network-in-python-language-modeling-task/
[6] See the original paper at https://arxiv.org/pdf/1802.05365.pdf
[7] See the original paper at https://arxiv.org/pdf/1810.04805.pdf

Note that today there are many extensions/modifications to the models mentioned above such as GPT-2 and the very restricted API based GPT-3, and extension to BERT such as RoBERTa, DistilBERT.[8] I hope I was able to motivate interested readers to autonomously investigate further where the research frontier on transfer learning in NLP is leading us.

4. Sources/Tutorials for understanding and implementing transfer learning models for NLP

Good introduction and a lot of reading material on neural language models
https://machinelearningmastery.com/statistical-language-modeling-and-neural-language-models/

Understanding LSTM Networks
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

What makes the AWD-LSTM great?
https://yashuseth.blog/2018/09/12/awd-lstm-explanation-understanding-language-model

Text classification with ULMFit
https://www.analyticsvidhya.com/blog/2018/11/tutorial-text-classification-ulmfit-fastai-library/

How to fine-tune and apply BERT
https://www.analyticsvidhya.com/blog/2020/01/learning-path-nlp-2020/?utm_source=blog&utm_medium=fine_tune_BERT

Tansformer models with the fast.ai Library
https://towardsdatascience.com/fastai-with-transformers-bert-roberta-xlnet-xlm-distilbert-4f41ee18ecb2

References

[1] Ruder, Sebastian. (2017) "Transfer Learning - Machine Learning's Next Frontier", http://ruder.io/transfer-learning/

[2] Ruder, Sebastian. (2019) "Neural Transfer Learning for Natural Language Processing", Ph.D. Thesis, National University of Ireland, Galway, http://ruder.io/thesis/

[3] Do, Chuong and Andrew Ng. (2005) “Transfer learning for text classification”, conference paper, https://papers.nips.cc/paper/2843-transfer-learning-for-text-classification.pdf

[8] See the following tutorial on towardsdatascience.com: https://towardsdatascience.com/fastai-with-transformers-bert-roberta-xlnet-xlm-distilbert-4f41ee18ecb2

[4] Wikipedia contributors. (2020, November 1). Language model. In Wikipedia, The Free Encyclopedia. Retrieved 05:57, November 4, 2020, from https://en.wikipedia.org/w/index.php?title=Language_model&oldid=986592354

[5] Wikipedia contributors. (2020, October 24). "Word embedding". In Wikipedia, The Free Encyclopedia. Retrieved 07:10, November 4, 2020, from https://en.wikipedia.org/w/index.php?title=Word_embedding&oldid=985266609

[6] Jeswani, Shrishti. (2020) "A Study of Transfer Learning Methods within Natural Language Processing and Reinforcement Learning", University of California, Berkeley, Technical Report No. UCB/EECS-2020-98, https://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-98.html

[7] Joshi, Prateek. (2020, July, 21) "Transfer Learning for NLP: Fine-Tuning BERT for Text Classification" In Analytics Vidhya. Retrieved 11:12, November 4, 2020, from https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/

[8] Malte, Aditya and Pratik Ratadiya. (2020) "Evolution of Transfer Learning in Natural Language Processing" https://arxiv.org/abs/1910.07370

[9] Joshi, Prateek. (2019, March, 21) "A Step-by-Step NLP Guide to Learn ELMo for Extracting Features from Text" In Analytics Vidhya. Retrieved 11:12, November 4, 2020, from https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/