fast.ai NLP

Introducing state of the art text classification with universal language models

Written: 15 May 2018 by *Jeremy Howard and Sebastian Ruder* • Classification

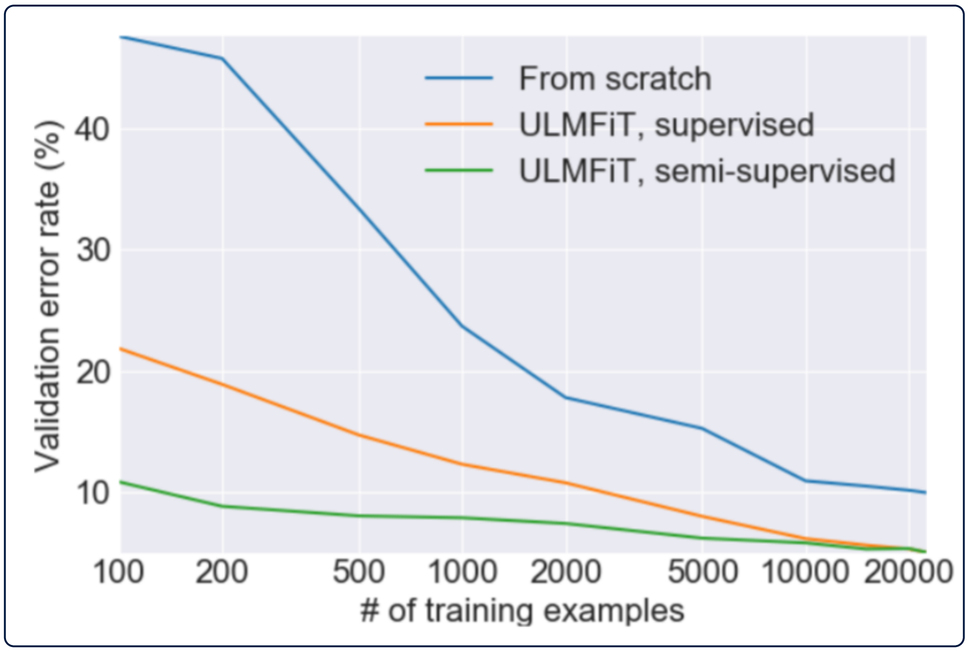
This post is a lay-person's introduction to our new paper, which shows how to classify documents automatically with both higher accuracy and less data requirements than previous approaches. We'll explain in simple terms: natural language processing; text classification; transfer learning; language modeling; and how our approach brings these ideas together. If you're already familar with NLP and deep learning, you'll probably want to jump over to our NLP classification page for technical links.

Introduction

Today we're releasing our paper Universal Language Model Fine-tuning for Text Classification (ULMFiT), pre-trained models, and full source code in the Python programming language. The paper has been peer-reviewed and accepted for presentation at the Annual Meeting of the Association for Computational Linguistics (ACL 2018). For links to videos providing an in-depth walk-through of the approach, all the Python modules used, pre-trained models, and scripts for building your own models, see our NLP classification page.

This method dramatically improves over previous approaches to *text classification*, and the code and pre-trained models allow anyone to leverage this new approach to better solve problems such as:

- Finding documents relevant to a legal case;
- Identifying spam, bots, and offensive comments;
- Classifying positive and negative reviews of a product;
- Grouping articles by political orientation;
- ...and much more.



ULMFiT requires orders of magnitude less data than previous approaches. (Figure 3 from the paper)

So what does this new technique do exactly? Let's first of all take a look at part of the abstract from the paper and see what it says—and then in the rest of this article we'll unpack this and learn exactly what it all means:

> Transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100x more data.

NLP, deep learning, and classification

Natural language processing (NLP) is an area of computer science and artificial intelligence that deals with (as the name suggests) using computers to process natural language. Natural language refers to the normal languages we use to communicate day to day, such as English or Chinese—as opposed to specialized languages like computer code or music notation. NLP is used in a wide variety of applications, such as search, personal assistants, summarization, etc. Overall, NLP is challenging as the strict rules we use when writing computer code are a poor fit for the nuance and flexibility of language. You've likely run into those limitations yourself, with the frustrating experience of trying to communicate with automated phone answering systems, or limited capabilities of early "conversational bots" like Siri.

In the last couple of years we've started to see *deep learning* making significant inroads into areas where computers have previously seen limited success. Rather than requiring a set of fixed rules that are defined by the programmer, deep learning uses neural networks that learn rich non-linear relationships directly from data. Most notable is the success of deep learning in computer vision, as seen for example in the rapid progress in image classification in the Imagenet competition.

Deep learning has also seen some success in NLP, for example in automatic translation, as discussed in this extensive NY Times article. A common feature of successful NLP tasks is that large amounts of labeled data are available for training a model. However, until now such applications were limited to those institutions that were able to collect and label huge datasets and had the computational resources to process them on a cluster of computers for a long time.

One particulaar area that is still challenging with deep learning for NLP, curiously enough, is the exact area where it's been most successful in computer vision: *classification*. This refers to any problem where your goal is to categorize things (such as images, or documents) into groups (such as images of cats vs dogs, or reviews that are positive vs negative, and so forth). A huge number of important real-world problems turn out to largely be about classification, which is why, for example, the success of deep learning on Imagenet (which is a classification problem) has led to a great many commercial applications. In NLP, current approaches are good at identifying, for instance, when a movie review is positive or negative, a problem known as *sentiment analysis*. Models struggle, however, as soon as things get more ambiguous, as often there is not enough labeled data to learn from.

Transfer learning

Our goal was to address these two problems: a) deal with NLP problems where we don't have masses of data and computational resources, and b) make NLP classification easier. As it turned out, we (Jeremy and Sebastian) had both been working on the exact field that would solve this: transfer learning. Transfer learning refers to the use of a model that has been trained to solve one problem (such as classifying images from Imagenet) as the basis to solve some other somewhat similar problem. One common way to do this is by *fine-tuning* the original model (such as classifying CT scans into cancerous or not—an application of transfer learning that Jeremy developed when he founded Enlitic). Because the fine-tuned model doesn't have to learn from scratch, it can generally reach higher accuracy with much less data and computation time than models that don't use transfer learning.

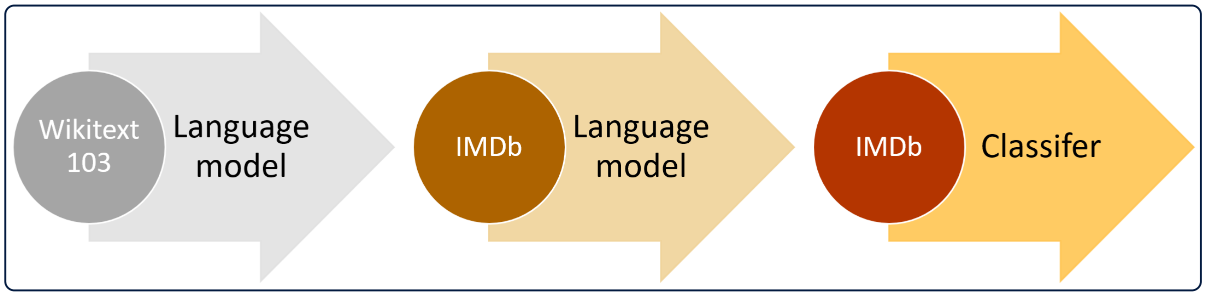
Very simple transfer learning using just a single layer of weights (known as *embeddings*) has been extremely popular for some years, such as the *word2vec* embeddings from Google. However, full neural networks in practice contain many layers, so only using transfer learning for a single layer was clearly just scratching the surface of what's possible.

The question, then, was what could we transfer from, in order to solve NLP problems? The answer to this question fell into Jeremy's lap, when his friend Stephen Merity announced he had developed the AWD LSTM language model, which was a dramatic improvement over previous approaches to *language modeling*. A language model is an NLP model which learns to predict the next word in a sentence. For instance, if your mobile phone keyboard guesses what word you are going to want to type next, then it's using a language model. The reason this is important is because for a language model to be really good at guessing what you'll say next, it needs a lot of *world knowledge* (e.g. "I ate a hot" → "dog", "It is very hot" → "weather"), and a deep understanding of grammar, semantics, and other elements of natural language. This is exactly the kind of knowledge that we leverage implicitly when we read and classify a document.

We found that in practice this approach to transfer learning has the features that allow it to be a *universal* approach to NLP transfer learning:

1. It works across tasks varying in document size, number, and label type
2. It uses a single architecture and training process
3. It requires no custom feature engineering or preprocessing
4. It does not require additional in-domain documents or labels.

Making it work



High level ULMFiT approach (IMDb example)

This idea has been tried before, but required millions of documents for adequate performance. We found that we could do a lot better by being smarter about how we fine-tune our language model. In particular, we found that if we carefully control how fast our model learns and update the pre-trained model so that it does not forget what it has previously learned, the model can adapt a lot better to a new dataset. One thing that we were particularly excited to find is that the model can learn well even from a limited number of examples. On one text classification dataset with two classes, we found that training our approach with only 100 labeled examples (and giving it access to about 50,000 unlabeled examples), we were able to achieve the same performance as training a model from scratch with 10,000 labeled examples.

Another important insight was that we could use any reasonably general and large language corpus to create a universal language model—something that we could fine-tune for any NLP target corpus. We decided to use Stephen Merity’s Wikitext 103 dataset, which contains a pre-processed large subset of English Wikipedia.

Research in NLP has mostly focused on English and training a model on a non-English language comes with its own set of challenges. Generally, the number of public datasets for non-English languages is small; if you want to train a text classification model for a language such as Thai, you invariably have to collect your own data. Collecting data in a non-English language often means that you need to annotate the data or find annotators yourself, as crowd-sourcing services such as Amazon Mechanical Turk mostly employ English-speaking annotators.

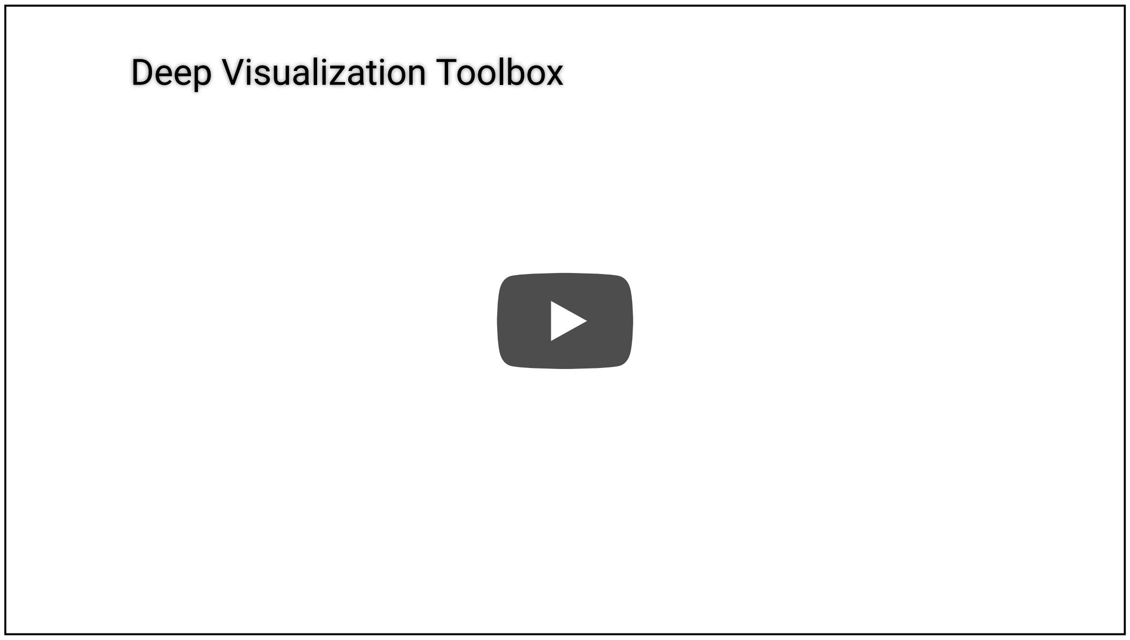
With ULMFiT, we can make training text classification models for languages other than English a lot easier as all we need is access to a Wikipedia, which is currently available for 301 languages, a small number of documents that can easily be annotated by hand, and optionally additional unlabeled documents. To make this even easier, we will soon launch a model zoo with pre-trained language models for many languages.

The future of ULMFiT

We have found that the approach works well on different tasks with the same settings. Besides text classification, there are many other important NLP problems, such as sequence tagging or natural language generation, that we hope ULMFiT will make easier to tackle in the future. We will be updating this site as we complete our experiments and build models in these areas.

In computer vision the success of transfer learning and availability of pre-trained Imagenet models has transformed the field. Many people including entrepreneurs, scientists, and engineers are now using fine-tuned Imagenet models to solve important problems involving computer vision—everything from improving crop yields in Africa to building robots that sort lego bricks. Now that the same tools are available for processing natural language, we hope to see the same explosion of applications in this field too.

Whilst we already have shown state of the art results for text classification, there’s still a lot of work to be done to really get the most out of NLP transfer learning. In the computer vision world there have been a number of important and insightful papers that have analyzed transfer learning in that field in depth. In particular, Yosinski et al. tried to answer the question “how transferable are features in deep neural networks”, and Huh et al. studied “what makes ImageNet good for transfer learning”. Yosinski even created a rich visualization toolkit to help practitioners better understand the features in their computer vision models (shown in the video below).



Deep Visualization Toolbox

If you try out ULMFiT on a new problem or dataset, we'd love to hear about it! Drop by the deep learning forums and tell us how it goes (and do let us know if you have any questions along the way).