STANFORD
DAWN

Home Vision Slides People Members Projects Papers

Seminar Blog

Massive Multi-Task Learning with Snorkel MeTaL: Bringing More Supervision to Bear

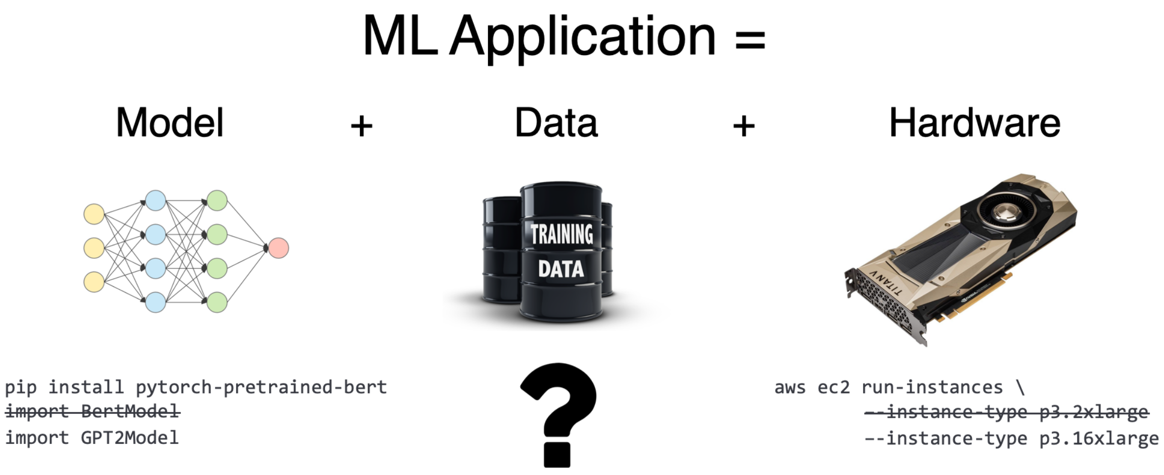
by Braden Hancock, Clara McCreery, Ines Chami, Vincent Chen, Sen Wu, Jared Dunnmon, Paroma Varma, Max Lam, and Chris Ré

22 Mar 2019

> ***TL;DR***: *We use Snorkel MeTaL*[1] *to construct a simple model (pretrained BERT + linear task heads) and incorporate a variety of supervision signals (traditional supervision, transfer learning, multi-task learning, weak supervision, and ensembling) in a Massive Multi-Task Learning (MMTL) setting, achieving a new state-of-the-art score on the GLUE Benchmark and four of its nine component tasks (CoLA, SST-2, MRPC, STS-B). Research is ongoing, with a code release of the MMTL package coming in Snorkel MeTaL v0.5 in April 2019.*

Designing for Flexible Supervision

There are three components you need to tackle a supervised learning problem: a model, hardware, and training data. Thanks to flourishing research and open source communities, state-of-the-art models are typically only one `pip install` away (looking at you, Google, Hugging Face, OpenAI)! Thanks to cloud computing, state-of-the-art hardware is growing equally accessible: a virtual machine with 8 of the latest and greatest GPUs can be spun up on demand in minutes (thanks, AWS and Google Cloud)! Gathering enough labeled data to train these open-source models on this hardware, however, is rarely so simple. In fact, this obstacle has become the primary bottleneck for most machine learning applications.[2] For this reason, **practitioners are increasingly turning to more indirect ways of injecting supervision signal into their models.**[3]



`pip install pytorch-pretrained-bert`
~~`import BertModel`~~
`import GPT2Model`

`aws ec2 run-instances \`
~~`--instance-type p3.2xlarge`~~
`--instance-type p3.16xlarge`

State-of-the-art model architectures and hardware are becoming increasingly commoditized, accessible with just a few lines of code. Obtaining state-of-the-art-caliber training data, however, still requires a fair bit of flexibility and creativity.

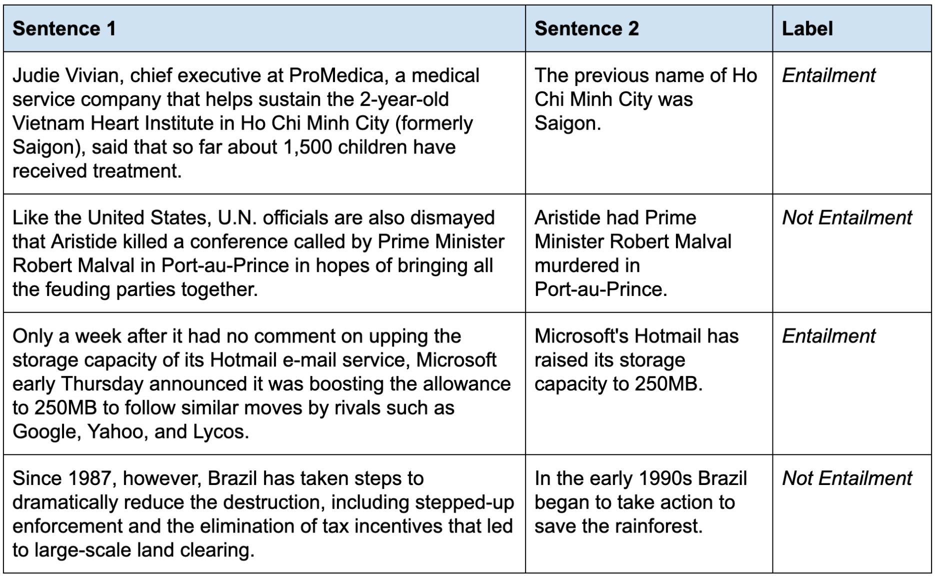
Observing this trend, we set out to build a framework where supervision is a first-class citizen. Our goal was to make it as easy as possible to support the many potential sources of supervision signal commonly used today, including traditional supervision, transfer learning, multi-task learning, and weak supervision. We call this setting—where we have large numbers of tasks and labels of varying types, granularities, and label accuracies—*Massive Multi-Task Learning* (MMTL). To guide our development process, we used the recently introduced GLUE Benchmark as our playground.[4]

The GLUE Benchmark consists of nine natural language understanding tasks (e.g., natural language inference, sentence similarity, etc.). Each comes with its own unique set of examples and labels, ranging in size from 635 training examples (WNLI) to 393k (MNLI). To keep our focus and contributions on the supervision aspect of ML, we used commodity hardware (AWS p3.8xlarge instances) and a very simple model architecture (a shared BERT module plus single-layer linear task heads). Thus, each improvement that we saw came from taking advantage of a new source of signal or more intelligently mixing the supervision we already had.

In the following sections, we'll walk through how our score improved on one of these tasks, RTE (Recognizing Textual Entailment), by adding increasing amounts of supervision signal.[5] This work is just beginning, though; our hope is that when we release our open source framework in April, others will find new and creative ways to bring even more signal into the fold, and push the state of the art even further!

Signal 1: Traditional Supervision

The RTE dataset comes with a labeled training set of 2.5k examples. The goal of the task is to indicate whether the second sentence is implied by the first; this is sometimes called a *textual entailment* task or *natural language inference* (NLI).



Sentence 1	Sentence 2	Label
Judie Vivian, chief executive at ProMedica, a medical service company that helps sustain the 2-year-old Vietnam Heart Institute in Ho Chi Minh City (formerly Saigon), said that so far about 1,500 children have received treatment.	The previous name of Ho Chi Minh City was Saigon.	*Entailment*
Like the United States, U.N. officials are also dismayed that Aristide killed a conference called by Prime Minister Robert Malval in Port-au-Prince in hopes of bringing all the feuding parties together.	Aristide had Prime Minister Robert Malval murdered in Port-au-Prince.	*Not Entailment*
Only a week after it had no comment on upping the storage capacity of its Hotmail e-mail service, Microsoft early Thursday announced it was boosting the allowance to 250MB to follow similar moves by rivals such as Google, Yahoo, and Lycos.	Microsoft's Hotmail has raised its storage capacity to 250MB.	*Entailment*
Since 1987, however, Brazil has taken steps to dramatically reduce the destruction, including stepped-up enforcement and the elimination of tax incentives that led to large-scale land clearing.	In the early 1990s Brazil began to take action to save the rainforest.	*Not Entailment*

Example sentence pairs from the RTE dataset. The label indicates whether Sentence 2 must be true if Sentence 1 is true.

We start by considering the baseline of using well-known NLP architectures. Training a standard biLSTM on this dataset yields an accuracy score of **57.4**. Adding ELMo embeddings and an attention layer on top bumps it up a little bit to **58.9**.[6] Unfortunately, no matter how fancy we make our model architecture, there's only so much our model can learn from 2.5k examples.

We need more signal. 💪

Signal 2: Transfer Learning

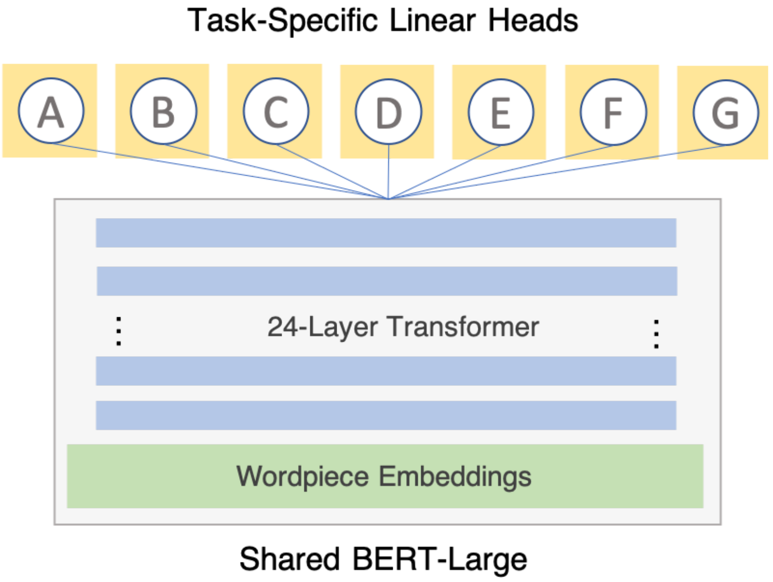
The year 2018 has been called "NLP's ImageNet moment" by some. In other words, it is the year that transfer learning really took off, bringing impressive boosts to a wide variety of NLP tasks.[7] The most well-known victories in this category came with the introduction of ULMFit, GPT, and perhaps loudest of all, BERT—a massive 24-layer transformer network with over 340M parameters trained on a corpus of 3.3 billion words with 256 TPUs over 4 days![8] Each of these models were trained on some sort of language modeling task (loosely speaking, predicting a word from context), which has proven to be a fairly robust choice of task for pre-training NLP models. It turns out that in order to predict a word or sentence from its context, it helps to have an understanding of syntax, grammar, sentiment, coreference resolution, etc.; consequently, the resulting representation is generally rich and useful for a wide variety of tasks.

By fine-tuning a linear layer on top of a pre-trained BERT module, we saw the validation score skyrocket 17.6 points up to **76.5**. The RTE dataset is still small, but by first pre-training on much larger corpora, the network begins the fine-tuning process having already developed many useful intermediate representations which the RTE task head can then take advantage of. Intuitively, this shifts much of the burden of representation learning to the pre-training stage and allows the task head to specialize more on how to combine those intermediate representations for its particular task. Still, we'd like to do better.

We need more signal. 💪

Signal 3: Multi-Task Learning

Language modeling can teach a model a lot of things, but won't teach it everything it needs to know for optimal performance on a more complex task such as natural language inference (NLI), which arguably requires an even deeper level of natural language understanding. Thus, to improve performance on RTE further, we perform multi-task learning with other tasks more closely related to RTE. For example, it just so happens that three of the other tasks in the GLUE benchmark are also NLI tasks.



Our MTL architecture is very simple: the shared portion (over 99.99% of all network parameters) is a single PyTorch module (BERT-Large), with each task having a task-specific linear layer for a task head.

Multi-Task Learning (MTL) is the technique of training a single model to predict multiple tasks using a shared representation (see Sebastian Ruder's MTL survey for an excellent overview). Because we believe that it is supervision, not architectures, that really move the needle in ML applications, we keep our architecture very simple: each task that we add to the network simply adds a linear head on top of the BERT module. These linear heads map from the output dimension of the pretrained BERT model (1024-dim) to the cardinality of the classification task (2-dim for RTE).

Our training schedule is simple as well: break up all participating tasks into batches, shuffle them randomly, and feed them through the network one at a time such that each training example from each task is seen exactly once per epoch. After 10 epochs of MTL training, we take the best checkpoint and fine-tune on the individual tasks for an additional 5 epochs, allowing them to take advantage of shared information in the former stage and reduce destructive interference between tasks in the latter stage.[9] From the MTL training alone, RTE gets another significant boost (5.8 points) to an overall validation accuracy of **82.3**. The additional task-specific fine-tuning bumps that up to **83.4**.

Expanding to *Massive* MTL

But there's no reason to limit MTL to only include other complete datasets. The Snorkel MeTaL MMTL package which we use was built specifically to facilitate multi-task learning over large numbers of diverse and varying granularities of supervision; for this reason, it supports **arbitrary network modules, data types, and label types.** We can, for example, mix labels at the sentence label with labels at the token level.

As a thought experiment, suppose error analysis suggests that many mistakes made by the model stem from a lack of understanding of grammatical structure—we may consequently decide to add the auxiliary task of predicting the part of speech tag produced by an off-the-shelf parser for each token in our existing datasets. Or if mistakes related to coreference resolution seem prevalent, we might use an off-the-shelf coreference resolution system to produce labels for the MTL model to learn to predict. These token-level labels will certainly not be perfect, since they are the output of another model that has its own biases, error modes, and blind spots. However, the other models that produce these auxiliary task labels are typically trained over very large labeled datasets, which carry useful signal we would like to leverage. While our leaderboard submission only includes traditional multi-task learning so far, we plan to explore more in this MMTL space in future work.

We need more signal. 💪

Signal 4: Dataset Slicing

As we examined the mistakes made by our model, we noticed the model was consistently underperforming on particular slices of the data (i.e., subsets of the dataset with some property in common). For example, whereas our model achieved an accuracy of 83.4 overall on the validation set, it scored only 76.7 on examples with rare punctuation (e.g., dashes or semicolons) in them, and only 58.3 on examples with multiple pronouns. It's possible that these examples are simply more difficult on average—our model may never perform as well on them as it does on the rest of the dataset—but that doesn't mean that the model can't still make significant improvements on these slices if they're given a little extra attention.

Using heuristics such as the two described above, we programmatically identified the examples in our training set belonging to each slice of interest. (This can be seen as another form of weak supervision, where noisy, higher-level inputs such as heuristics or outputs from another model are used to provide some kind of supervision signal for another model). We then add another linear task head on top of our model for each of these slices, and train them only on those examples belonging to their respective slices. This allows a small portion of the network to focus on a learning a representation that will help it to improve performance on these examples where we are performing poorly. And because most of our network parameters are shared among tasks, these slice-inspired tweaks to the representation can then also be utilized by the primary head for that task (via hard parameter sharing)[10], allowing for improvements on the overall task accuracy. Adding the two slice heads mentioned above bumped the scores on our slices from 76.7 to 79.3 and from 58.3 to 75.0, respectively, giving us an overall RTE validation score of **84.1**.

We need more signal. 💪

Signal 5: Ensembling

Finally, we observed that the uncased BERT model (which operates over inputs all lowercased text) performed better on some of our tasks, while the cased BERT model performed better on others. Conventional wisdom is that an uncased model can reduce overfitting, since more raw tokens will map to the same embedding (e.g., both “Cat” and “cat”), resulting in more frequent co-occurrence with other tokens and therefore a less sparse training signal. On the other hand, cased models potentially have access to more information, with capitalization often distinguishing acronyms or proper nouns (e.g., ‘zip’ to fasten vs. ZIP codes, or Google the company versus google the verb). Furthermore, because of the way the pretrained BERT tokenizes (with a fixed vocabulary of 30k wordpieces), each model uses not only different weights, but also potentially different tokenizations for the same sentence. The result is two models with fairly similar overall performance, but slightly different error buckets.

We also observed overfitting to our validation set, likely due to its small size (276 examples), so we ensembled models trained with different training/validation set splits. By averaging the predicted probabilities from six such models (cased and uncased on varied dataset splits), we were able to increase our RTE test score by an additional point of accuracy.

We need more signal. 💪

Signal 6: TBD

Snorkel is a system for rapidly creating, modeling, and managing training data. Snorkel MeTaL is our multi-task version of Snorkel for exploring multi-task supervision and multi-task learning.

Next month we will be releasing Snorkel MeTaL v0.5, which will include the MMTL package we used to achieve our state-of-the-art results. We hope it will serve as a foundation for others to build on, allowing for new and creative ways to bring additional supervision signal into the fold and mix available signal in intelligent ways. In the coming months we will also be writing a paper with more in-depth analysis and ablation studies. You can always find our latest work (blog posts, collaborations, papers, etc.) exploring supervision on the snorkel landing page: snorkel.stanford.edu. And as always, we'd love to hear your thoughts and feedback in the comments below, on Twitter, or on Github.

Footnotes

[1] Snorkel MeTaL was recently presented at AAAI in (Ratner, et al., 2019), "Training Complex Models with Multi-Task Weak Supervision." It is the multi-task version of Snorkel (Ratner, et al. 2017), "Snorkel: Rapid Training Data Creation with Weak Supervision, which won "Best of VLDB" in 2017. The Snorkel MeTaL repository can be found at https://github.com/HazyResearch/metal. ↩

[2] See our previous blog post "Weak Supervision: The New Programming Paradigm for Machine Learning" for more of our thoughts on this topic. ↩

[3] See Abraham Starosta's "Building NLP Classifiers Cheaply with Transfer Learning and Weak Supervision" for an excellent example of this in action. ↩

[4] While the GLUE benchmark is an NLP task, the Snorkel MeTaL framework is modality agnostic; it can be used with text, images, structured data, etc. or combinations of these for multi-modal applications. ↩

[5] Note that RTE was one of the tasks most amenable to improvement from auxiliary supervision; the magnitude of these improvements varies by task, but in general we find that these trends hold across tasks. The two exceptions are WNLI, for which no submission has improved over guessing the majority class, and CoLA, which is the least similar to the other tasks and empirically worked best when trained in a single-task manner. ↩

[6] These baselines are test numbers pulled from (Bowman, et al. 2019), "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". ↩

[7] Note that transfer learning can be seen as a special case of multi-task learning, as discussed in our previous blog post "Emerging Topics in Multi-Task Learning Systems". ↩

[8] See the two-part series "Dissescting BERT" for an excellent overview of its internal workings. ↩

[9] This type of two-stage fine-tuning was also employed by MT-DNN and BERT on STILTs, two other top submissions on the GLUE leaderboard. ↩

[10] Sebastian Ruder's MTL survey gives a nice overview of the distinction between hard and soft parameter sharing in MTL. ↩

Related posts

DeepFreak: Learning Crystallography Diffraction Patterns with Automated Machine Learning 23 Apr 2019

Model Assertions as a Tool for Quality Assurance and Improving ML Models 11 Mar 2019

DAWN PI Delivers NeurIPS Keynote 05 Dec 2018

3 Comments **dawn-blog** 1 **Login**

♡ **Recommend** 1 Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Jerry Yao • a month ago

When do you release Snorkel MeTaL v0.5 ? It's almost May!

• Reply • Share ›

Abraham Starosta • 2 months ago

Awesome!

• Reply • Share ›

Jerry Yao • 2 months ago

awsome

• Reply • Share ›

Subscribe **Add Disqus to your siteAdd DisqusAdd** **Disqus' Privacy PolicyPrivacy PolicyPrivacy**