If Not Notebooks, Then What?

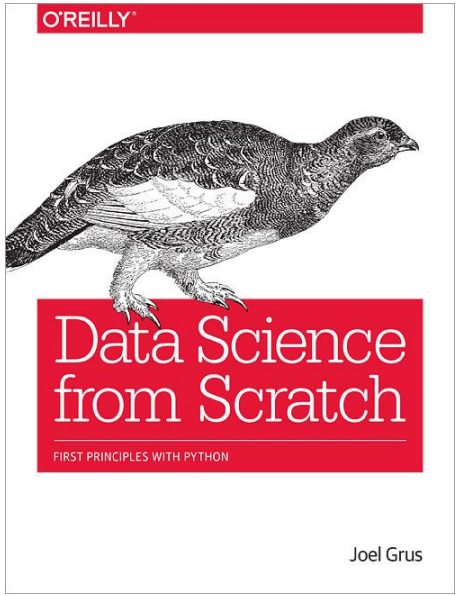
Thoughts on notebooks and reproducibility

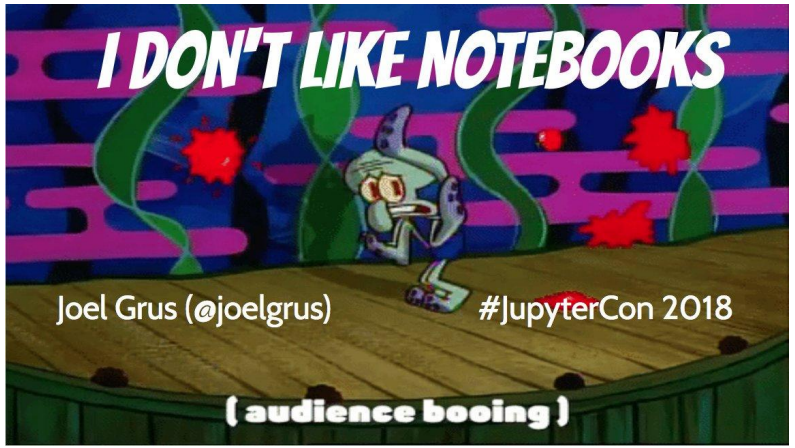
AAAI 2019 Workshop on Reproducible AI

Joel Grus
@joelgrus

About Me

- Research engineer on the AllenNLP team
- Previously SWE@Google, Data Science@VoloMetrix, ...
- Author of *Data Science from Scratch*
- Co-host of "Adversarial Learning" podcast
- The "Fizz Buzz in Tensorflow" guy
- Started "the first notebook war"





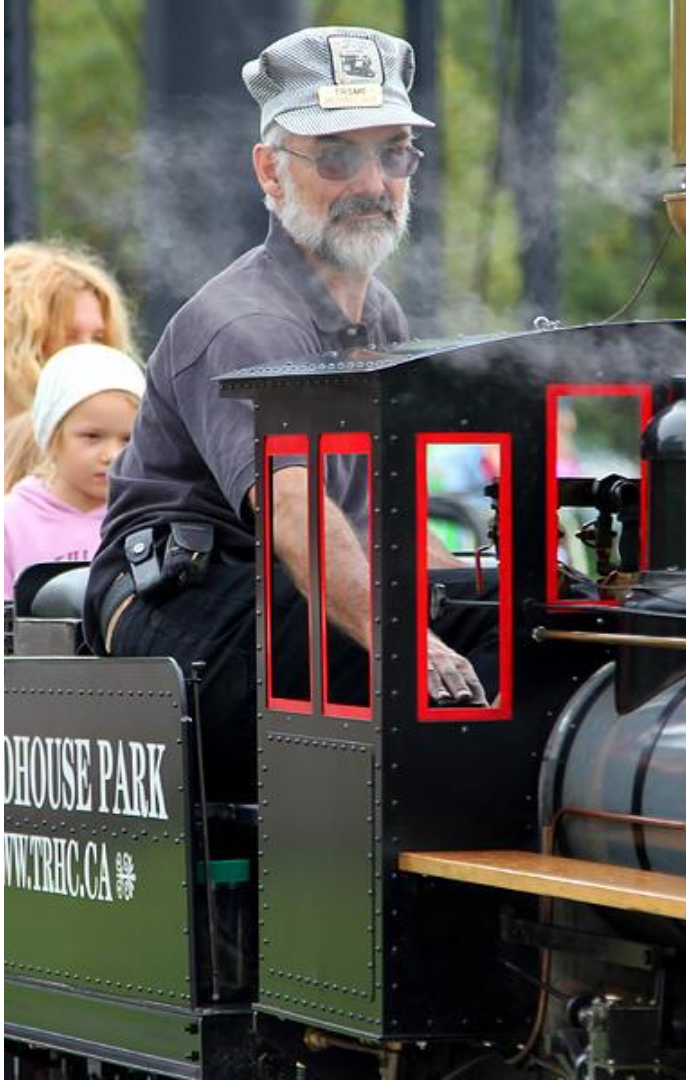
AllenNLP

AI for the Common Good.

Our mission is to contribute to humanity through high-impact AI research and engineering.

What is a Research Engineer?

- I'm not a researcher
  - (although I can pretend to be one)
- I'm an engineer who *cares deeply* about research
- My job is to help researchers be successful
  - by building tools for them
  - by advocating best practices
  - and also by collaborating with them
- And so my job involves thinking a lot about reproducibility



Outline

- Why reproducibility?
- Why not notebooks?
- If not notebooks, then what?
- Reproducibility and AllenNLP
- Reproducibility and Beaker

Why Reproducibility?


I take a somewhat expansive
view of the role of
reproducibility in science

Reproducibility Helps With Correctness

If no one ever runs your code but you, it might be wrong

Reproducibility Protects Against Bad Actors

- Certainly *you* would never try to publish fraudulent research, but what about your bitter rival?
- (Hopefully this is a minor consideration.)

Reproducibility Makes It Easy to Try New Datasets

- Maybe your model works great on a different dataset
- Maybe it works terribly on a different dataset
- Hard datasets help move AI forward

Reproducibility Makes It Easy to Try New Tasks

You haven't thought of (or tried) every way your model could be used

Can you use BERT to generate text?

16 Jan 2019

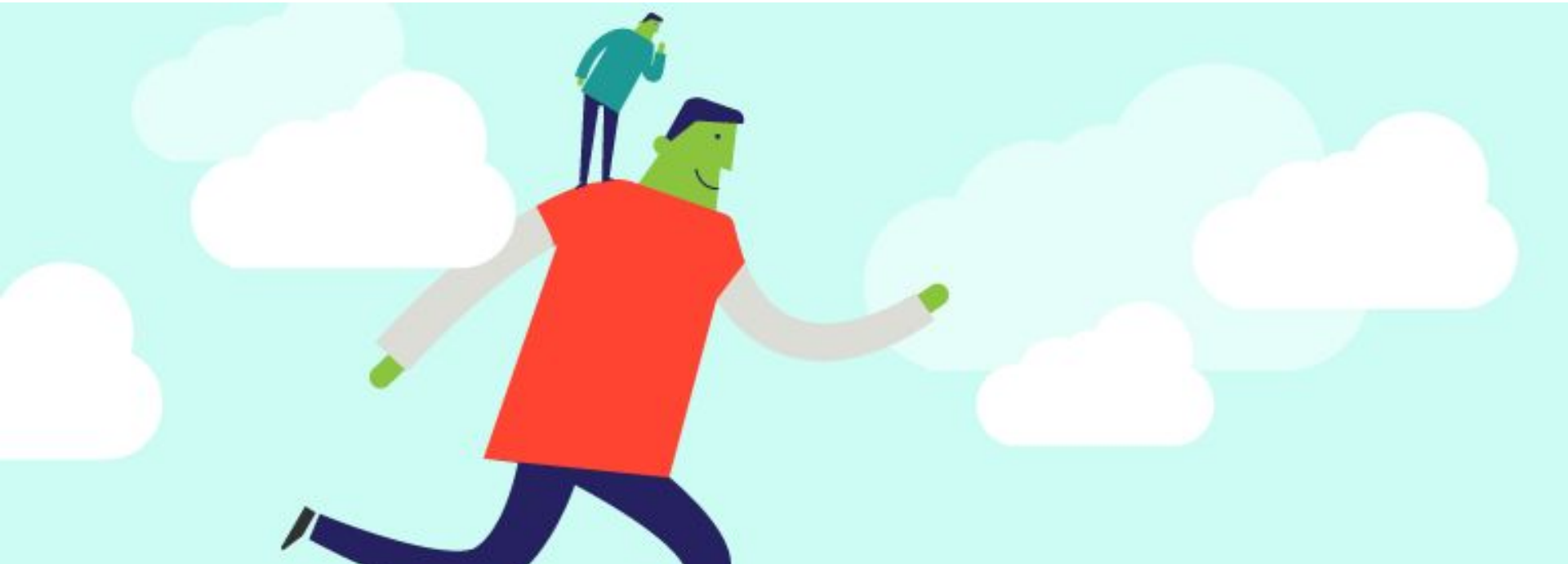
Just quickly wondering if you can use BERT to generate text. I'm using huggingface's pytorch pretrained BERT model (thanks!). I know BERT isn't designed to generate text, just wondering if it's possible. It's trained to predict a masked word, so maybe if I make a partial sentence, and add a fake mask to the end, it will predict the next word. As a first pass on this, I'll give it a sentence that has a dead giveaway last token, and see what happens.

Reproducibility Enables Strong Baselines

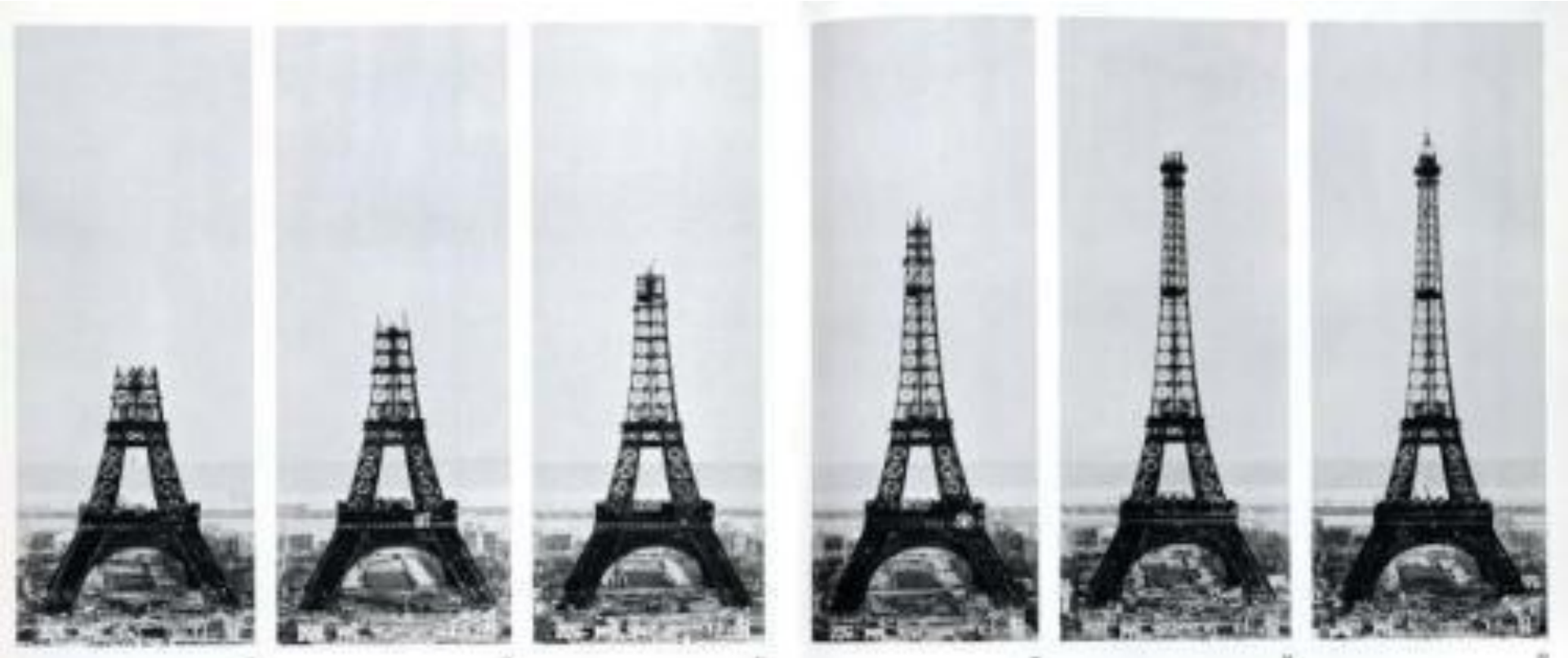
Wouldn't you like your model to be the standard by which new models are judged?

Reproducibility Is Necessary For Extensibility

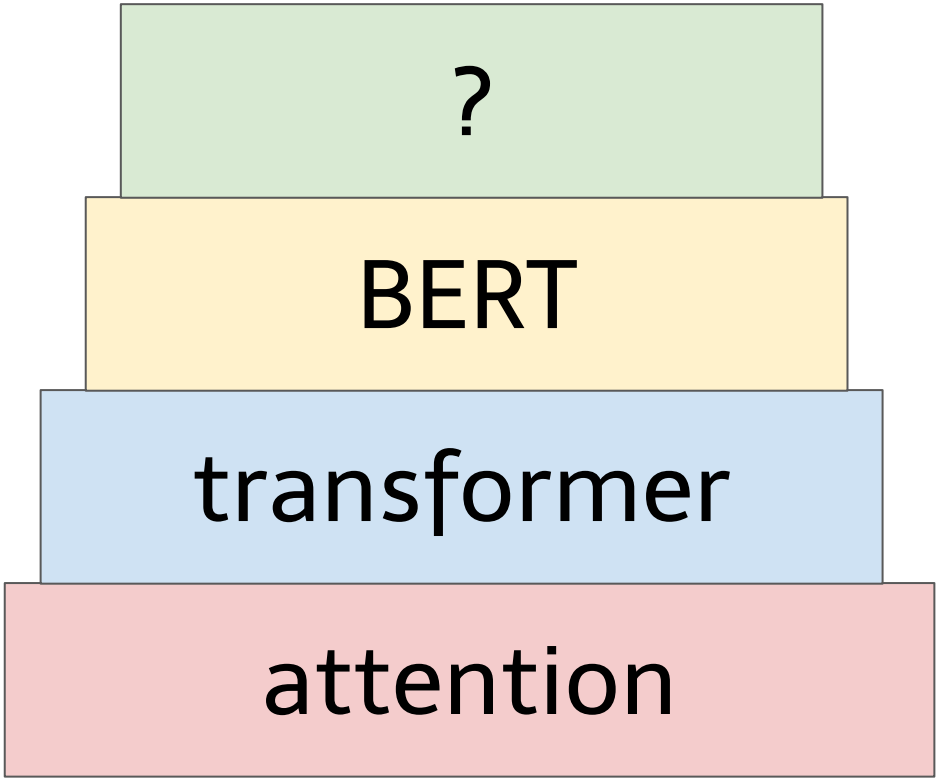
"you can't stand on the shoulders of giants if they keep their shoulders private"



Extensibility Leads to Progress



Extensibility Leads to Progress



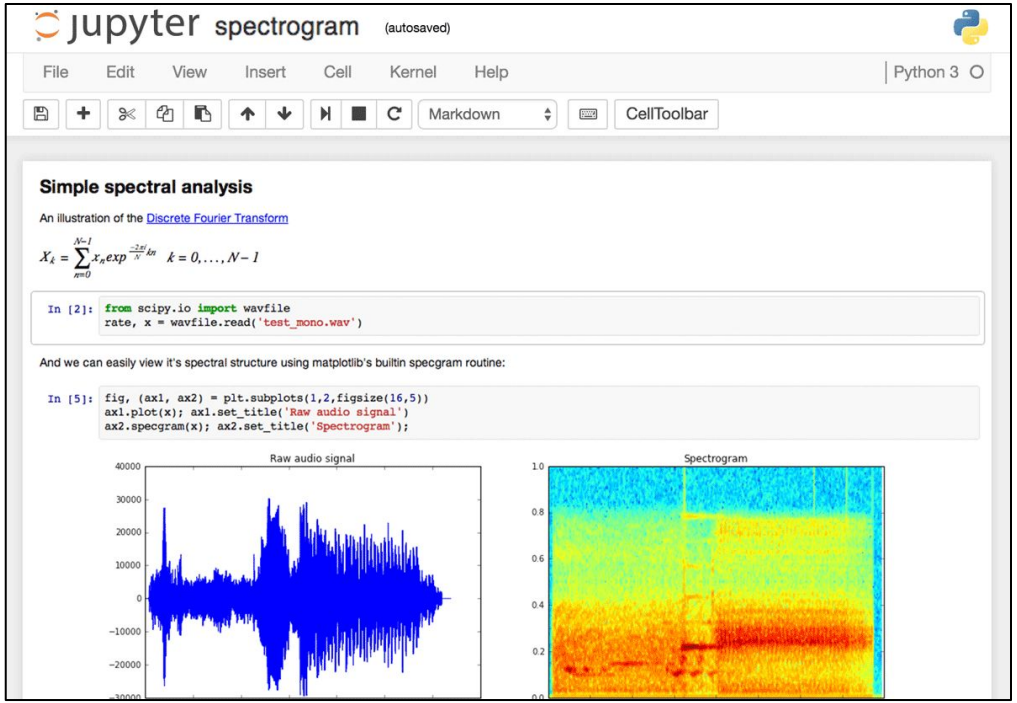
Fundamental Premise:

The tools you choose and the processes you adopt can make reproducibility either a lot harder or a lot easier.

Why Not Notebooks?

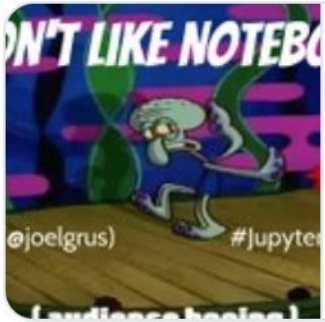
What are Jupyter Notebooks?

- Computational environment that uses the lab notebook as a metaphor
- Easy to mix marked-up text, executable code, results, visualizations, interactive widgets, and so on
- Very popular with data scientists, less so (?) with researchers
- Often promoted as beneficial for "reproducibility"



Joel Grus
@joelgrus

slides for my "I Don't Like Notebooks" #JupyterCon talk:



I Don't Like Notebooks - Joel Grus - #JupyterCon ...
I Don't Like Notebooks hi, I'm Joel, and I don't like notebooks Joel Grus (@joelgrus) #JupyterCon 2018
docs.google.com

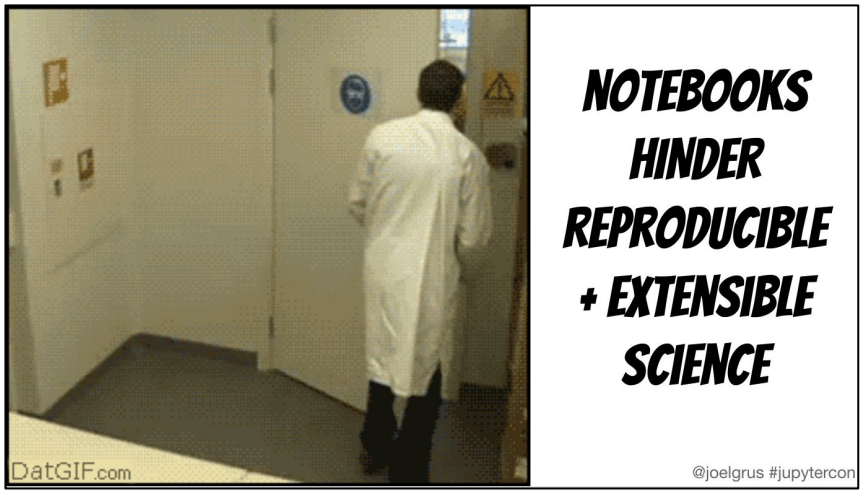
9:55 AM - 24 Aug 2018 from Manhattan, NY

677 Retweets **1,985** Likes

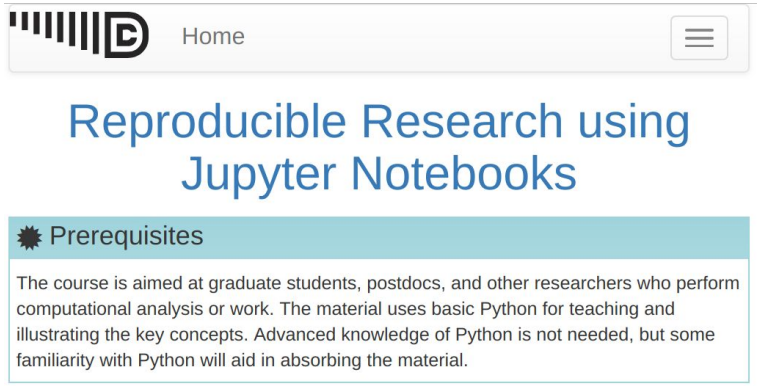
136 677 2.0K

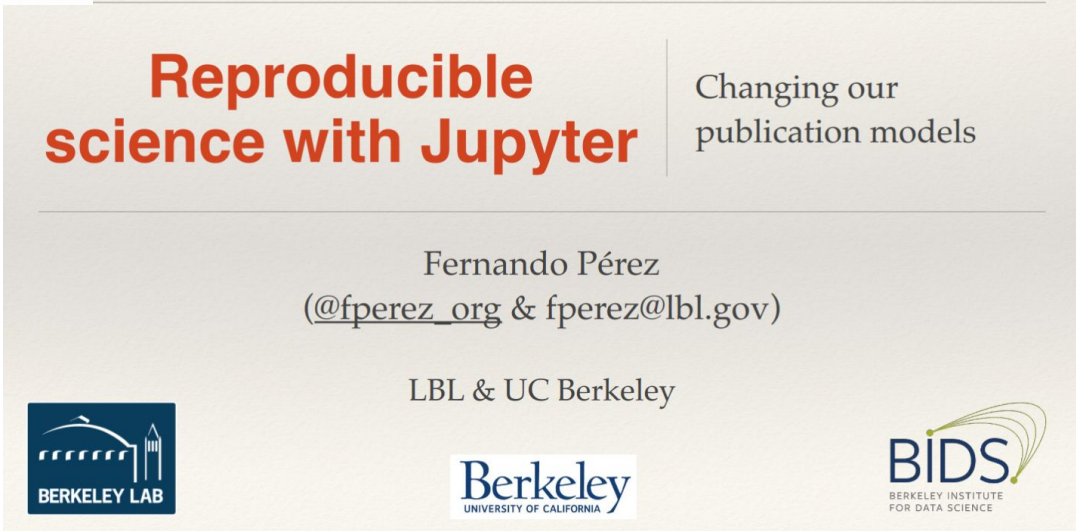
How Could Someone Not Like Notebooks?

- I had a lot of complaints (but also some positives and suggestions)
- It's a very comprehensive talk, you should check it out
- A lot of the complaints were around user-unfriendliness + confusingness
- One dislike was particularly relevant to this workshop:



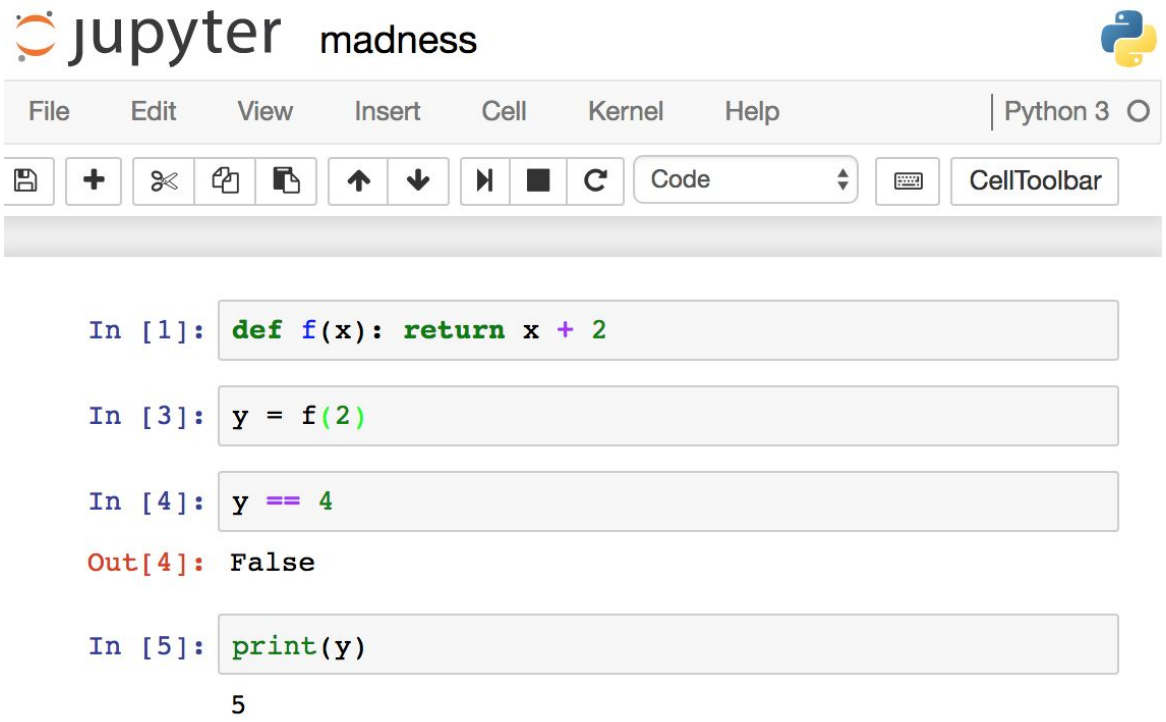
Notebooks for Reproducibility?





Code That's in Notebooks Is Hard to Re-Run

Notebooks Allow Out-of-Order Execution



unless you are very disciplined, *you* may not even be able to reproduce your work

Notebooks Hard-Code Their Parameters

```
        logits_flatten = binary_logits.view(-1, 2)

        return logits_flatten, hidden
```

In [4]:
```
rnn = RNN(input_size=88, hidden_size=512, num_classes=88)
rnn = rnn.cuda()

criterion = nn.CrossEntropyLoss().cuda()

criterion_val = nn.CrossEntropyLoss(size_average=False).cuda()

learning_rate = 0.001
optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
```

In [6]:
```
%matplotlib notebook

import sys, os
sys.path.append("/home/[redacted]/repos/pytorch-segmentation-detection/")
sys.path.insert(0, '/home/[redacted]/repos/pytorch-segmentation-detection/vision/')
```

Notebooks Don't Help You Enforce Dependencies

Branch: master ▾ Find file Copy path

[redacted] / notebooks / [redacted]n.ipynb

[redacted] added the link to download midi library 11417b5 on Jan 27

1 contributor

1246 lines (1245 sloc) | 104 KB Raw Blame History

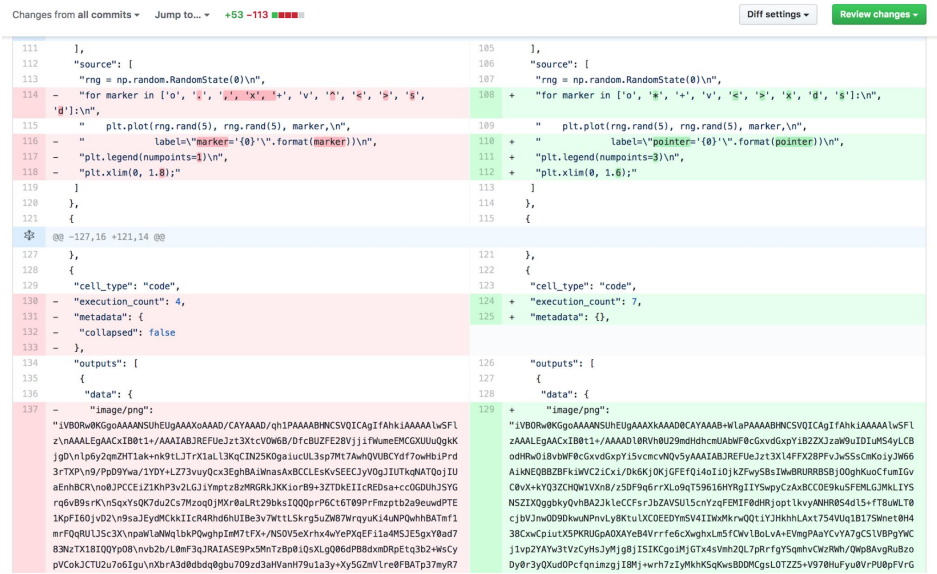
In [1]:
```
import pretty_midi
import numpy as np
import torch
import torch.nn as nn
from torch.autograd import Variable
import torch.utils.data as data
import os
import random
```

Notebooks Complicate Collaboration

Collaboration is a Forcing Function for Reproducibility

Notebooks Don't Play Well With Source Control

distributed source control is the de facto way to collaborate on coding projects



Notebooks Lock You Into Using Notebooks

Notebook Loader

Here we have our Notebook Loader. It's actually quite simple - once we figure out the filename of the module, all it does is:

1. load the notebook document into memory
2. create an empty Module
3. execute every cell in the Module namespace

Since IPython cells can have extended syntax, the IPython transform is applied to turn each of these cells into their pure-Python counterparts before executing them. If all of your notebook cells are pure-Python, this step is unnecessary.

[ ]:
```
class NotebookLoader(object):
    """Module Loader for Jupyter Notebooks"""
    def __init__(self, path=None):
        self.shell = InteractiveShell.instance()
        self.path = path

    def load_module(self, fullname):
        """import a notebook as a module"""
        path = find_notebook(fullname, self.path)

        print ("importing Jupyter notebook from %s" % path)

        # load the notebook object
        with io.open(path, 'r', encoding='utf-8') as f:
            nb = read(f, 4)
```

The Module Finder

The finder is a simple object that tells you whether a name can be imported, and returns the appropriate loader. All this one does is check, when you do:

```
import mynotebook
```

it checks whether `mynotebook.ipynb` exists. If a notebook is found, then it returns a NotebookLoader.

otebooks as

ant to import code fro

e not plain Python files

Register the hook

Now we register the `NotebookFinder` with `sys.meta_path`

[ ]:
```
sys.meta_path.append(NotebookFinder())
```

Notebooks Conflate "Library" Code and "Experiment" Code

In [3]:
```
class RNN(nn.Module):

    def __init__(self, input_size, hidden_size, num_classes, n_layers=2):

        super(RNN, self).__init__()

        self.input_size = input_size
        self.hidden_size = hidden_size
        self.num_classes = num_classes
        self.n_layers = n_layers

        self.notes_encoder = nn.Linear(in_features=input_size, out_features=hidden_size)

        self.lstm = nn.LSTM(hidden_size, hidden_size, n_layers)

        self.logits_fc = nn.Linear(hidden_size, num_classes)


    def forward(self, input_sequences, input_sequences_lengths, hidden=None):

        batch_size = input_sequences.shape[1]

        notes_encoded = self.notes_encoder(input_sequences)

        # Here we run rnns only on non-padded regions of the batch
        packed = torch.nn.utils.rnn.pack_padded_sequence(notes_encoded, input_sequences_lengths)
        outputs, hidden = self.lstm(packed, hidden)
        outputs, output_lengths = torch.nn.utils.rnn.pad_packed_sequence(outputs) # unpack (back
 to padded)

        logits = self.logits_fc(outputs)

        logits = logits.transpose(0, 1).contiguous()

        neg_logits = (1 - logits)

        # Since the BCE loss doesn't support masking, we use the crossentropy
        binary_logits = torch.stack((logits, neg_logits), dim=3).contiguous()

        logits_flatten = binary_logits.view(-1, 2)

        return logits_flatten, hidden
```

In [4]:
```
rnn = RNN(input_size=88, hidden_size=512, num_classes=88)
rnn = rnn.cuda()

criterion = nn.CrossEntropyLoss().cuda()

criterion_val = nn.CrossEntropyLoss(size_average=False).cuda()

learning_rate = 0.001
optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
```

In [8]:
```
clip = 1.0
epochs_number = 12000000
sample_history = []
best_val_loss = float("inf")

for epoch_number in xrange(epochs_number):

    for batch in trainset_loader:

        post_processed_batch_tuple = post_process_sequence_batch(batch)

        input_sequences_batch, output_sequences_batch, sequences_lengths = post_processed_batch_
tuple

        output_sequences_batch_var =  Variable( output_sequences_batch.contiguous().view(-1).cud
a() )

        input_sequences_batch_var = Variable( input_sequences_batch.cuda() )

        optimizer.zero_grad()

        logits, _ = rnn(input_sequences_batch_var, sequences_lengths)

        loss = criterion(logits, output_sequences_batch_var)
        loss_list.append( loss.data[0] )
        loss.backward()

        torch.nn.utils.clip_grad_norm(rnn.parameters(), clip)

        optimizer.step()


    current_val_loss = validate()
    val_list.append(current_val_loss)

    if current_val_loss < best_val_loss:

        torch.save(rnn.state_dict(), 'music_rnn.pth')
        best_val_loss = current_val_loss
```

Notebooks are a recipe for poorly factored code

François Chollet
@fchollet

Following

Buggy code is bad science. Poorly tuned benchmarks are bad science. Poorly factored code is bad science (hinders reproducibility, increases chances of a mistake). If your field is all about empirical validation, then your code *is* a large part of your scientific output.

12:26 AM - 15 Jul 2018

12 Retweets **66** Likes

3 12 66

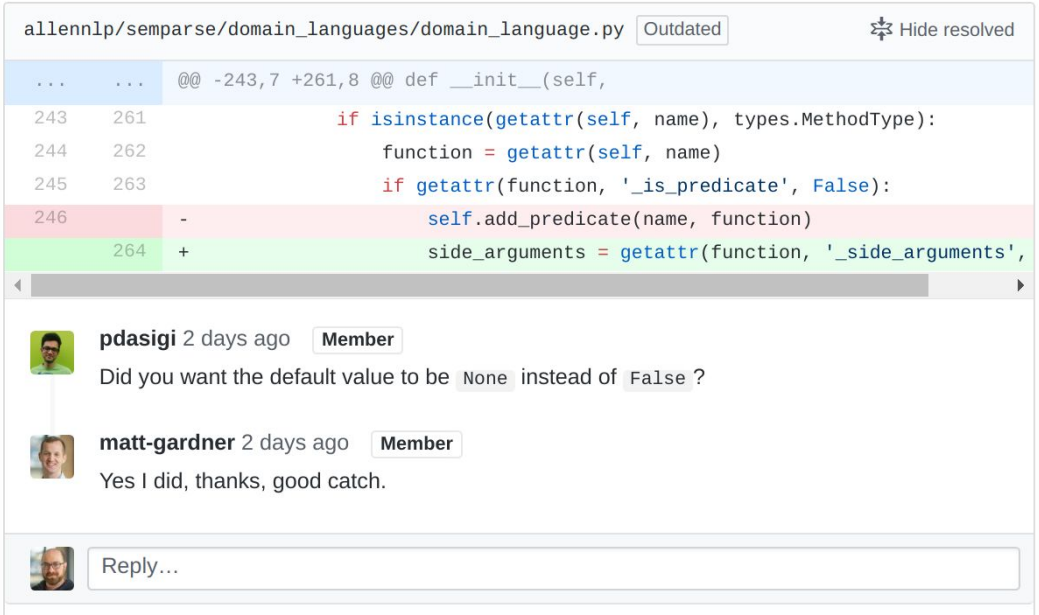
Notebooks Frustrate Software-Engineering Best Practices

Software Engineering?

- You may not think of your AI experiments as software engineering
- But I do
- And you should!
- I sometimes give talks on "why data scientists should care about software engineering best practices"
- This is sort of my "why AI researchers should care about software engineering best practices" variant

Notebooks Complicate Code Reviews

- Code reviews are an early bulwark against incorrect code (and hence incorrect science)
- Code reviews help you grow as a coder and as a scientist



Notebooks Make it Hard to Unit Test

- Unit tests assure you that small pieces of your logic are correct
- Unit tests make it easy to iterate by running your model end-to-end on small datasets
- Unit tests make it safe to refactor your code
- Unit tests are small working examples
- Yes, you can put `asserts` in your notebook, but **you want to be able to run your tests without running your experiment**

ai.ipynb

```
# load data
# maybe assert something

# train model
# maybe assert something

# get results
# maybe assert something
# save results
```

Yeah, but what do unit tests for AI experiments even look like?

Unit Tests for AI Experiments

```
def test_forward_pass_runs_correctly(self):
    training_tensors = self.dataset.as_tensor_dict()
    output_dict = self.model(**training_tensors)
    tags = output_dict['tags']
    assert len(tags) == 2
    assert len(tags[0]) == 7
    assert len(tags[1]) == 7
    for example_tags in tags:
        for tag_id in example_tags:
            tag = self.model.vocab.get_token_from_index(tag_id, namespace="labels")
            assert tag in {'O', 'I-ORG', 'I-PER', 'I-LOC'}
```

tiny known dataset

check that model runs

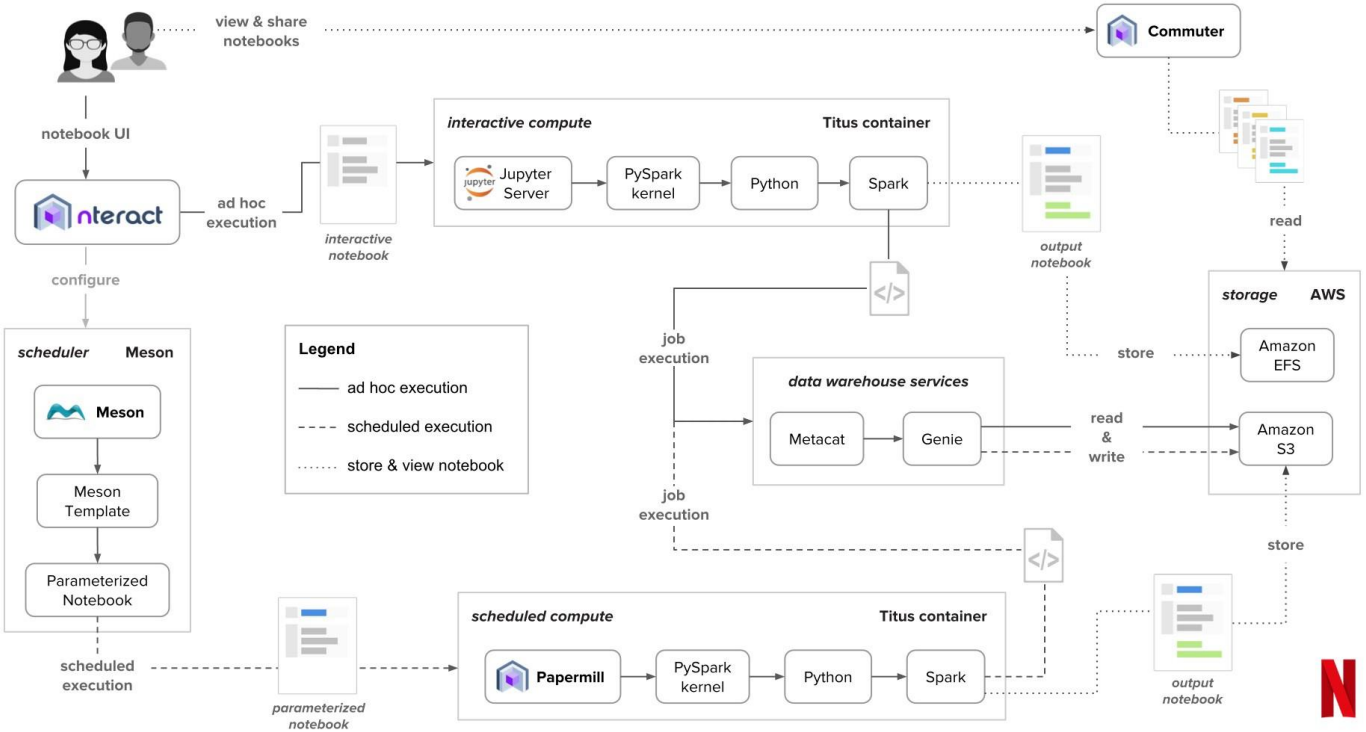
check that output has the right fields

check that output has the right shape

check that output has reasonable values

Notebooks Make Science Harder

Notebooks Are Hard to Parametrize



Notebooks Lump Together Code and Data

- Often you'll run many experiments with the same code but different parameters
- For analysis it can be good to have your results mixed in
- For science you want to aggregate results across experiments
- For science you want your results to have a life of their own

Notebook Code Can't Easily Be Built On

In Short

- "Notebooks as a source of reproducibility" presupposes a *static* view of AI as a science:
  - "I did some science, now here is an artifact containing the code and data"
- I'm advocating for a *dynamic* view:
  - "I did some science, now you do some more science *on top of it*"
  - Which is the kind of reproducibility that moves AI forward

If Not Notebooks, Then What?

Code in Modules

```
# models/crf_tagger.py

class CrfTagger(Model):
    """
    The ``CrfTagger`` encodes a sequence of text with a ``Seq2SeqEncoder``,
    then uses a Conditional Random Field model to predict a tag for each token in the sequence.
    def __init__(self, vocab: Vocabulary,
                 text_field_embedder: TextFieldEmbedder,
                 encoder: Seq2SeqEncoder,
                 label_namespace: str = "labels",
                 feedforward: Optional[FeedForward] = None,
                 label_encoding: Optional[str] = None,
                 include_start_end_transitions: bool = True,
                 constrain_crf_decoding: bool = None,
                 calculate_span_f1: bool = None,
                 dropout: Optional[float] = None,
                 verbose_metrics: bool = False,
                 initializer: InitializerApplicator = InitializerApplicator(),
                 regularizer: Optional[RegularizerApplicator] = None) -> None:
        super().__init__(vocab, regularizer)
        ...
```

Unit Tests

- develop your model on a tiny test set
- iterate
- run the tests frequently

The best time to find mistakes is before you run your experiments!

Be Explicit About Your Dependencies

```
#### ESSENTIAL LIBRARIES FOR MAIN FUNCTIONALITY ####


# This installs Pytorch for CUDA 8 only. If you are using a newer version,
# please visit http://pytorch.org/ and install the relevant version.
# For now AllenNLP works with both PyTorch 1.0 and 0.4.1. Expect that in
# the future only >=1.0 will be supported.
torch>=0.4.1


# Parameter parsing (but not on Windows).
jsonnet==0.10.0 ; sys.platform != 'win32'


# Adds an @overrides decorator for better documentation and error checking when using subclasses.
overrides


# Used by some old code.  We moved away from it because it's too slow, but some old code still
# imports this.
nltk

# Mainly used for the faster tokenizer.
spacy>=2.0,<2.1
```

Make It Easy To Vary Parameters

```
export BERT_BASE_DIR=/path/to/bert/uncased_L-12_H-768_A-12
export GLUE_DIR=/path/to/glue

python run_classifier.py \
  --task_name=MRPC \
  --do_train=true \
  --do_eval=true \
  --data_dir=$GLUE_DIR/MRPC \
  --vocab_file=$BERT_BASE_DIR/vocab.txt \
  --bert_config_file=$BERT_BASE_DIR/bert_config.json \
  --init_checkpoint=$BERT_BASE_DIR/bert_model.ckpt \
  --max_seq_length=128 \
  --train_batch_size=32 \
  --learning_rate=2e-5 \
  --num_train_epochs=3.0 \
  --output_dir=/tmp/mrpc_output/
```

Consider Docker Images

- Create container with OS + environment + code (+ data?)
- Can share and anyone can run (in theory)
- Build up step by step with smart caching when you change it

```
$ docker run -it --entrypoint /bin/bash allennlp/allennlp:v0.8.0
```

```
root@7d1f120a83e9:/stage/allennlp# echo '{"sentence": "Did Uriah honestly think he could beat the
game in under three hours?"}' | allennlp predict
https://s3-us-west-2.amazonaws.com/allennlp/models/srl-model-2018.05.25.tar.gz -
```


```
input:  {"sentence": "Did Uriah honestly think he could beat the game in under three hours?"}
prediction:  {"verbs": [{"verb": "Did", "description": "[V: Did] Uriah honestly think he could beat
the game in under three hours ?", "tags": ["B-V", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O",
"O", "O", "O"]}, {"verb": "think", "description": "Did [ARG0: Uriah] [ARGM-MNR: honestly] [V:
think] [ARG1: he could beat the game in under three hours] ?", "tags": ["O", "B-ARG0",
"B-ARGM-MNR", "B-V", "B-ARG1", "I-ARG1", "I-ARG1", "I-ARG1", "I-ARG1", "I-ARG1", "I-ARG1",
"I-ARG1", "I-ARG1", "O"]}, {"verb": "could", "description": "Did Uriah honestly think he [V: could]
beat the game in under three hours ?", "tags": ["O", "O", "O", "O", "O", "B-V", "O", "O", "O", "O",
"O", "O", "O", "O"]}, {"verb": "beat", "description": "Did Uriah honestly think [ARG0: he]
[ARGM-MOD: could] [V: beat] [ARG1: the game] [ARGM-TMP: in under three hours] ?", "tags": ["O",
"O", "O", "O", "B-ARG0", "B-ARGM-MOD", "B-V", "B-ARG1", "I-ARG1", "B-ARGM-TMP", "I-ARGM-TMP",
"I-ARGM-TMP", "I-ARGM-TMP", "O"]}], "words": ["Did", "Uriah", "honestly", "think", "he", "could",
"beat", "the", "game", "in", "under", "three", "hours", "?"]}
```

Provide Instructions

Installation

This repo was tested on Python 3.5+ and PyTorch 0.4.1/1.0.0

With pip

PyTorch pretrained bert can be installed by pip as follows:

```
pip install pytorch-pretrained-bert
```

From source

Clone the repository and run:

```
pip install [--editable] .
```

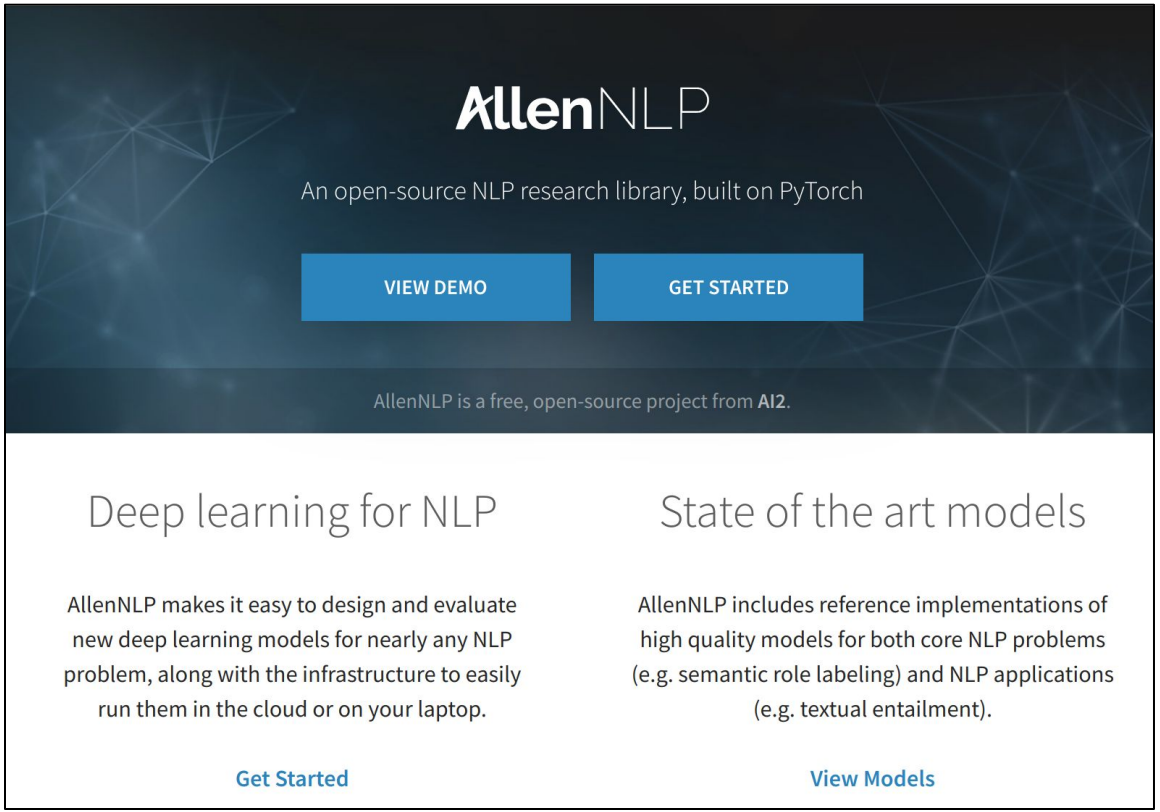
A series of tests is included in the tests folder and can be run using `pytest` (install pytest if needed: `pip install pytest`).

You can run the tests with the command:

```
python -m pytest -sv tests/
```

Reproducibility and AllenNLP

What is AllenNLP?



Programming to Higher-Level Abstractions

```
# models/crf_tagger.py

class CrfTagger(Model):
    """
    The ``CrfTagger`` encodes a sequence of text with a ``Seq2SeqEncoder``,
    then uses a Conditional Random Field model to predict a tag for each token in the sequence.
    def __init__(self, vocab: Vocabulary,
                 text_field_embedder: TextFieldEmbedder,
                 encoder: Seq2SeqEncoder,
                 label_namespace: str = "labels",
                 feedforward: Optional[FeedForward] = None,
                 label_encoding: Optional[str] = None,
                 include_start_end_transitions: bool = True,
                 constrain_crf_decoding: bool = None,
                 calculate_span_f1: bool = None,
                 dropout: Optional[float] = None,
                 verbose_metrics: bool = False,
                 initializer: InitializerApplicator = InitializerApplicator(),
                 regularizer: Optional[RegularizerApplicator] = None) -> None:
        super().__init__(vocab, regularizer)
        ...
```

Declarative Configuration

```
"model": {
  "type": "crf_tagger",
  "label_encoding": "BIOUL",
  "dropout": 0.5,
  "include_start_end_transitions": false,
  "text_field_embedder": {
    "token_embedders": {
      "tokens": {
          "type": "embedding",
          "embedding_dim": 50,
          "pretrained_file": "/path/to/glove.txt.gz",
          "trainable": true
      },
      "elmo":{
          "type": "elmo_token_embedder",
          "options_file": "/path/to/elmo/options.json",
          "weight_file": "/path/to/elmo/weights.hdf5",
          "do_layer_norm": false,
          "dropout": 0.0
      },
      "token_characters": {
          "type": "character_encoding",
          "embedding": {
              "embedding_dim": 16
          },
          "encoder": {
              "type": "cnn",
              "embedding_dim": 16,
              "num_filters": 128,
              "ngram_filter_sizes": [3],
              "conv_layer_activation": "relu"
          }
      }
    },
    "encoder": {
        "type": "lstm",
        "input_size": 1202,
        "hidden_size": 200,
        "num_layers": 2,
        "dropout": 0.5,
        "bidirectional": true
    },
    "regularizer": [
      [
        "scalar_parameters",
        {
          "type": "l2",
          "alpha": 0.1
        }
      ]
    ]
},
```

Declarative Configuration

JSON blob

some voodoo involving `inspect.getfullarg` spec and type annotations

`CrfTagger.__init__`

```
CrfTagger(
  (text_field_embedder): BasicTextFieldEmbedder(
    (token_embedder_token_characters):
TokenCharactersEncoder(
      (_embedding): TimeDistributed(
        (_module): Embedding()
      )
      (_encoder): TimeDistributed(
        (_module): PytorchSeq2VecWrapper(
          (_module): GRU(25, 80, num_layers=2,
batch_first=True, dropout=0.25, bidirectional=True)
        )
      )
    )
    (token_embedder_tokens): Embedding()
  )
  (encoder): PytorchSeq2SeqWrapper(
    (_module): GRU(210, 300, num_layers=2,
batch_first=True, dropout=0.5, bidirectional=True)
  )
  (tag_projection_layer): TimeDistributed(
    (_module): Linear(in_features=600, out_features=4,
bias=True)
  )
  (crf): ConditionalRandomField()
)
```

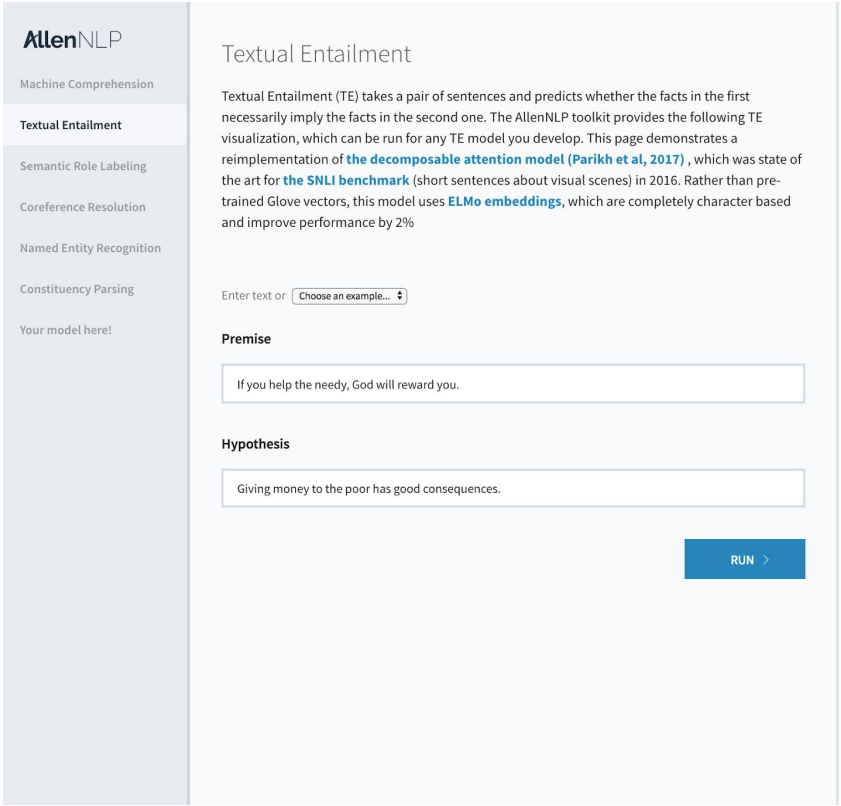
Command-Line Tools

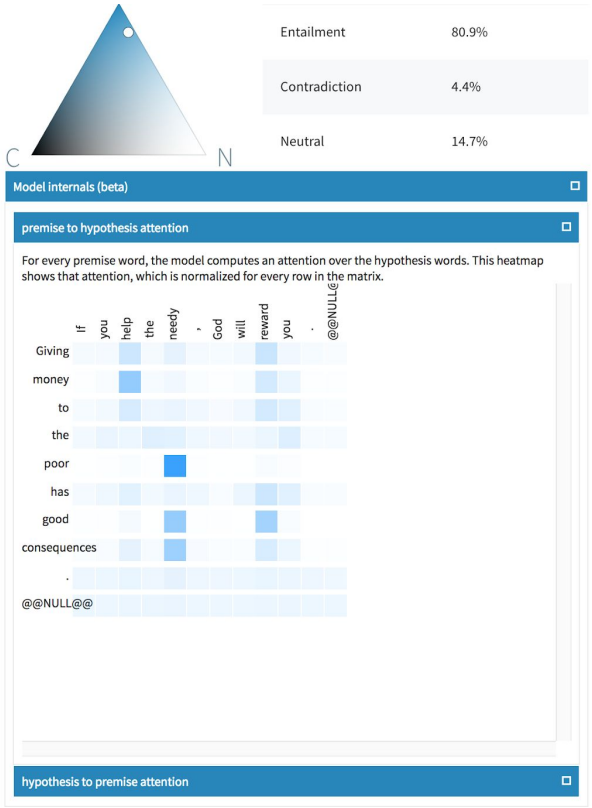
Using the Command Line Tool

In fact, we provide a command line tool that handles most common AllenNLP tasks, so in practice you probably wouldn't read the params or call `train_model` yourself (although you can), you'd just run the command

```
$ allennlp train tutorials/tagger/experiment.jsonnet \
                 -s /tmp/serialization_dir \
                 --include-package tutorials.tagger.config_allennlp
```

Interactive Demos and Visualization of Model Internals





Higher-Level NLP Abstractions as Library Primitives

- Field + Instance (nice representation of examples)
  - `question_field = TextField("What is the …", token_indexers)`
  - `instance = Instance({"question": question_field, "passage": passage_field)`
- Vocabulary (map word <-> index, label <-> index, etc)
- TokenIndexer (map token -> [index1, ..., indexn]
  - could be one per word
  - could be one per character
  - could be one per wordpiece
- TokenEmbedder (map indices -> embedding vectors)
- Seq2VecEncoder (map [v1, ..., vn] -> w)
- Seq2SeqEncoder (map [v1, .., vn] -> [w1, ..., wn])
- ...and many more

"I want to try using BERT vectors instead of GloVe vectors"

"Oh, great, now I'm going to make lots of changes to my code and maintain all these different versions so that my results are reproducible"

"I'll just make a new config file for the BERT version!"

```
"token_indexers": {
  "tokens": {
    "type": "single_id",
    "lowercase_tokens": true
  },


  "token_characters": {
    "type": "characters",
    "min_padding_length": 3
  }
}
```



```
"token_indexers": {
  "bert": {
      "type": "bert-pretrained",
      "pretrained_model": std.extVar("BERT_VOCAB"),
      "do_lowercase": false,
      "use_starting_offsets": true
  },
  "token_characters": {
    "type": "characters",
    "min_padding_length": 3
  }
}
```

```
"text_field_embedder": {
  "token_embedders": {
    "tokens": {
        "type": "embedding",
        "embedding_dim": 50,
        "pretrained_file": "/path/to/glove.tar.gz",
        "trainable": true
    },
    "token_characters": {
        "type": "character_encoding",
        "embedding": {
            "embedding_dim": 16
        },
        "encoder": {
            "type": "cnn",
            "embedding_dim": 16,
            "num_filters": 128,
            "ngram_filter_sizes": [3],
            "conv_layer_activation": "relu"
        }
      }
   },
},
```



```
"text_field_embedder": {
    "allow_unmatched_keys": true,
    "embedder_to_indexer_map": {
        "bert": ["bert", "bert-offsets"],
        "token_characters": ["token_characters"],
    },
    "token_embedders": {
        "bert": {
            "type": "bert-pretrained",
            "pretrained_model": std.extVar("BERT_WEIGHTS")
        },
        "token_characters": {
            "type": "character_encoding",
            "embedding": {
                "embedding_dim": 16
            },
            "encoder": {
                "type": "cnn",
                "embedding_dim": 16,
                "num_filters": 128,
                "ngram_filter_sizes": [3],
                "conv_layer_activation": "relu"
            }
        }
    }
},
```

"Want to Reproduce My Results?"

- create a virtual environment
- clone my GitHub repo and pip install its dependencies
  - including a specific version of AllenNLP
  - which includes a specific version of PyTorch
- each experiment has its own JSON configuration file
- `allennlp train specific_experiment.json -s /tmp/results`

Reproducibility and Beaker

What is Beaker?

- Kubernetes-based platform for rapid experimentation
- Specify experiments as Docker containers + config.yaml
- Request GPUs + Memory + etc
- Upload or mount existing datasets
- Track results
- (Currently runs on GKE, working on a "runs on your machines" version)
- Mostly internal-only for now (sorry)

What is Beaker?

```
beaker experiment run \
  --name wordcount-moby \
  --blueprint examples/wordcount \
  --source examples/moby:/input \
  --result-path /output
```

ex_ndcfzinlot02

Re-run

NER with char + BERT (using [SEP] and [CLS])

description

Created **2 months ago** by joelg

Tasks | Experiment Graph | Spec | Permissions

training

training

Succeeded Finished **2 months ago**. Ran for **2 hours**. Created by: joelg

Description	None
Logs	2506 Lines (377.3KiB) Show Full Log Download Full Log
Results	ds_og42u9uwhhld

logs

parameters

metrics

Metrics


```
{
  "best_epoch": 37,
  "best_validation_accuracy": 0.9904209337642615,
  "best_validation_accuracy3": 0.9915307036330361,
  "best_validation_f1-measure-overall": 0.95274928535391,
  "best_validation_loss": 51.06252528171913,
  "best_validation_precision-overall": 0.9519489247311828,
  "best_validation_recall-overall": 0.9535509929316729,
  "epoch": 61,
  "training_accuracy": 0.998340053334381,
  "training_accuracy3": 0.9985414078115715,
  "training_duration": "01:52:06",
  "training_epochs": 61,
  "training_f1-measure-overall": 0.9909963602306816,
  "training_loss": 3.9061280575665562,
  "training_precision-overall": 0.9913550804871817,
  "training_recall-overall": 0.9906378994850845,
  "training_start_epoch": 0,
  "validation_accuracy": 0.9893111638954869,
  "validation_accuracy3": 0.9906740391729294,
  "validation_f1-measure-overall": 0.9481381860972855,
  "validation_loss": 65.0678040747549,
  "validation_precision-overall": 0.947103274559194,
  "validation_recall-overall": 0.9491753618310333
}
```


Program Arguments	python -m allennlp.run train /config.json -s /output --file-friendly-logging

Environment Variables

Key	Value
BERT_VOCAB	/data/bert-vocab
BERT_WEIGHTS	/data/bert-weights
NER_TEST_A_PATH	/conll2003/eng.testa
NER_TEST_B_PATH	/conll2003/eng.testb
NER_TRAIN_DATA_PATH	/conll2003/eng.train
dataset_reader_coding_scheme	BIOUL
dataset_reader_tag_label	ner
dataset_reader_token_indexers_bert_do_lowerc…	False
dataset_reader_token_indexers_bert_pretrained…	/data/bert-vocab
dataset_reader_token_indexers_bert_type	bert-pretrained

Show all

Requirements	1 GPU 0 CPU 0.0B Memory
Blueprint	bp_w3b6mh1llks8
Experiment	ex_ndcfzinlot02

docker image

Input Datasets

Dataset ID	Mount Path
bert-base-cased-vocab	/data/bert-vocab
bert-base-cased	/data/bert-weights
conll2003	/conll2003
ds_5oc1fv432odm	/config.json

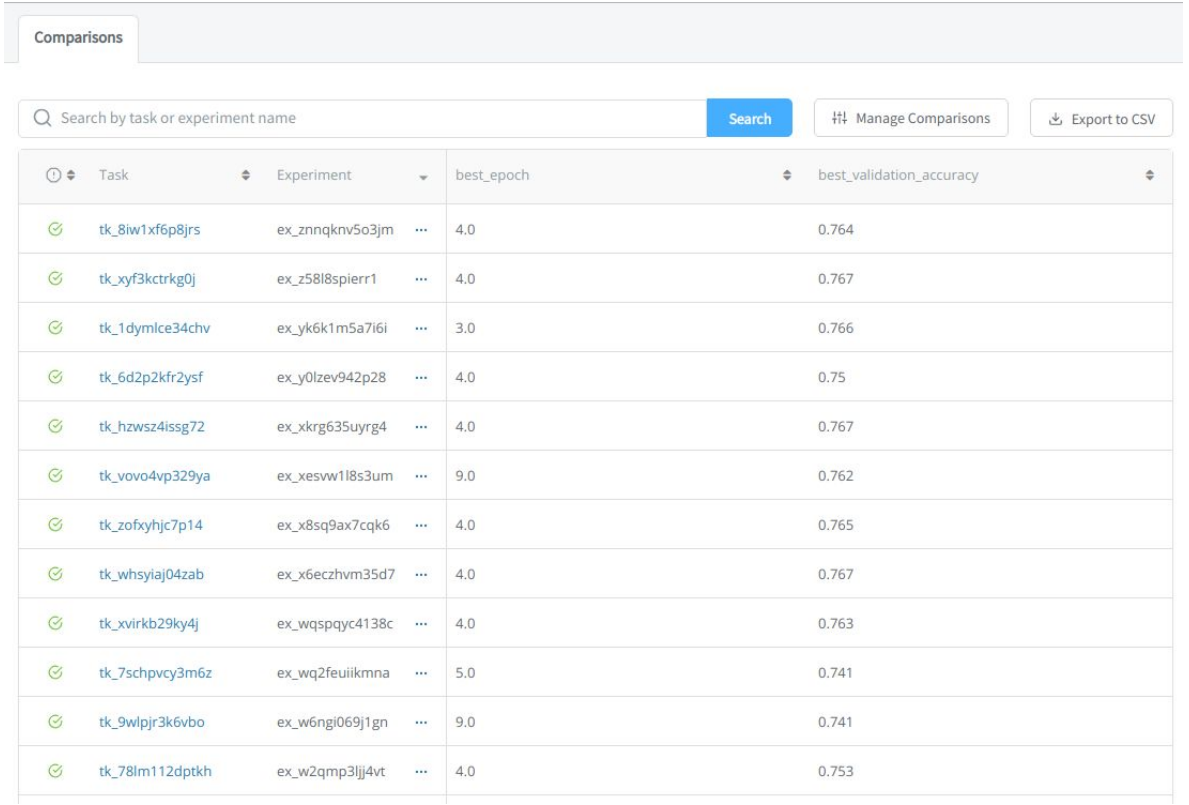
datasets

ID	tk_63w97m1997ba
Cost	$4.887

cost

Organize Experiments Into Groups

ideally you'd give them more descriptive names, though



Task	Experiment	best_epoch	best_validation_accuracy
tk_8iw1xf6p8jrs	ex_znnqknv5o3jm	4.0	0.764
tk_xyf3kctrkg0j	ex_z58l8spierr1	4.0	0.767
tk_1dymlce34chv	ex_yk6k1m5a7i6i	3.0	0.766
tk_6d2p2kfr2ysf	ex_y0lzev942p28	4.0	0.75
tk_hzwsz4issg72	ex_xkrg635uyrg4	4.0	0.767
tk_vovo4vp329ya	ex_xesvw1l8s3um	9.0	0.762
tk_zofxyhjc7p14	ex_x8sq9ax7cqk6	4.0	0.765
tk_whsyiaj04zab	ex_x6eczhvm35d7	4.0	0.767
tk_xvirkb29ky4j	ex_wqspqyc4138c	4.0	0.763
tk_7schpvcy3m6z	ex_wq2feuiikmna	5.0	0.741
tk_9wlpjr3k6vbo	ex_w6ngi069j1gn	9.0	0.741
tk_78lm112dptkh	ex_w2qmp3ljj4vt	4.0	0.753

Beaker and Reproducibility

- old code + new data => upload the dataset, reuse the blueprint
- new code + old data => create the blueprint, point at existing dataset
- want to see previous results?
  - inputs + logs + outputs stored "forever"
  - record of every experiment run + results
  - share with a link

To Sum Up

- Reproducibility is important for more than the obvious reasons
- Your choices of tools and processes make reproducibility easier or harder
- There are several ways that notebooks make reproducibility harder
- Search out tools that make reproducibility easier
- Adopt processes that make reproducibility easier

A Few Related Presentations

- I Don't Like Notebooks

  The talk that launched a thousand arguments. Heavy on the memes.

- Writing Code for NLP Research

  EMNLP 2018 tutorial from me + Matt Gardner + Mark Neumann, goes much deeper into "what good research code looks like"

- How Becoming Not a Data Scientist Made Me a Better Data Scientist

  Explores some similar themes in the context of "why data scientists should care about software engineering best practices"

Any questions?

me: @joelgrus

AI2: allenai.org

AllenNLP: allennlp.org

Beaker*: beaker.org

will tweet out slides from

@joelgrus and @ai2_allennlp