CS 410 Course Project – Final Report

Source code and test set predictions

Code:

https://github.com/wfcwfcwfcwfc/CourseProject/blob/main/cs410-classification-contest.ipynb

Test set predictions:

https://github.com/wfcwfcwfcwfc/CourseProject/blob/main/cs410-classification-contest-result.txt

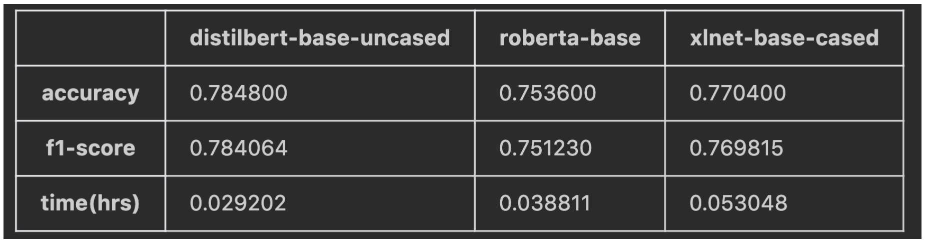
Explain your model, and how you perform the training. Describe your experiments with other methods that you may have tried and any hyperparameter tuning.

The classifier uses BERT transformer deep learning framework at its core. BERT is a recent NLP framework based on Transformer and self-attention architecture. It serves the "encoder" in the transformer model and is widely adopted in text generation and text classification.

Training a BERT model generally has two stages: pre-training and fine-tuning. Pre-training aims at providing BERT a general understanding of a language. This step builds the embeddings and trains the parameters. Fine-tuning is optimizing BERT for certain specific tasks. Pre-training requires large language corpus and tremendous computing power. A common practice is to use existing pre-trained model and fine-tuning for the specific task. In this scenario, I use "bert-large-cased-whole-word-masking" from hugging-face as pre-trained model, then fine-tuned with the training data provided.

After fine-tuning, the model is capable to perform predictions. The performance with default parameters beats the baseline.

I also explored other pre-trained models like "distilbert-base-uncased", "roberta-base", "xlnet-base-cased". They all have smaller number of parameters compared to "bert-large-cased-whole-word-masking". The performance is good on training set but does not pass the baseline in test data.



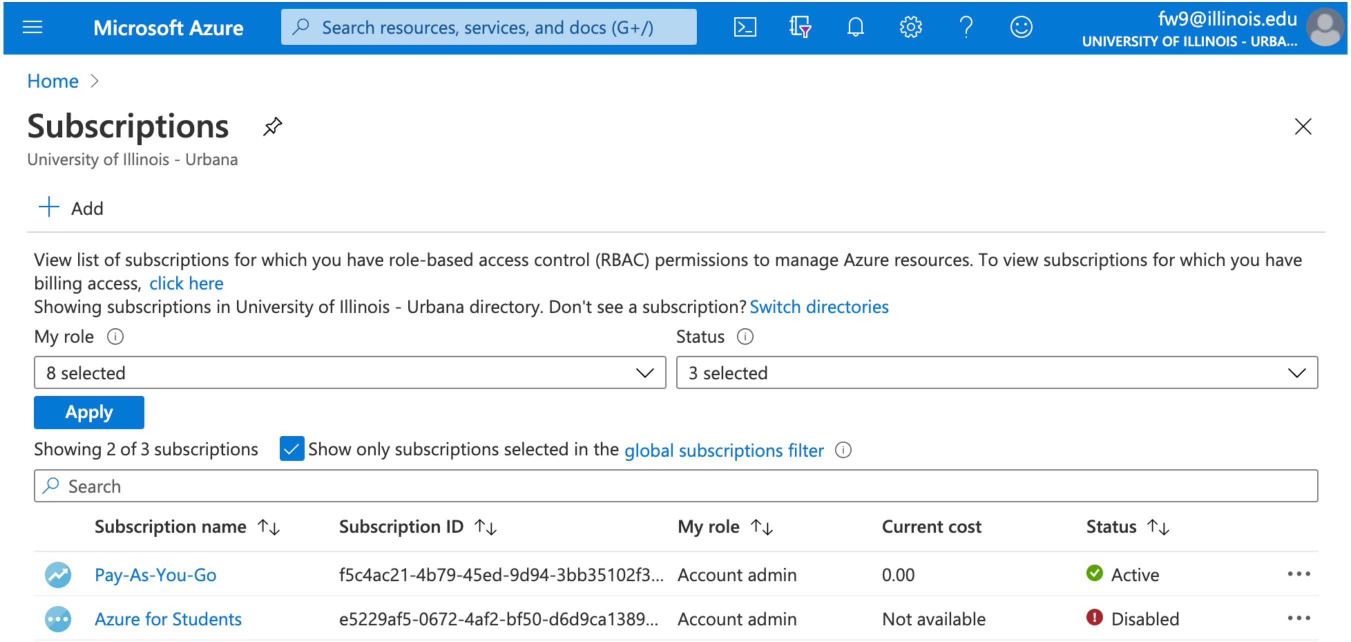
	distilbert-base-uncased	roberta-base	xlnet-base-cased
accuracy	0.784800	0.753600	0.770400
f1-score	0.784064	0.751230	0.769815
time(hrs)	0.029202	0.038811	0.053048

On engineering side, the model was implemented with PyTorch and deployed on Microsoft Azure ML Studio. It provided convenient middleware for ML tasks for easy deployment and prototyping. All the fine-tuning was done on a single compute, GPU instance and running time is less than 10 minutes. Compute GPU instance pre-installed with CUDA 10.1. All code is contained in the notebook shown in the beginning of this document. I used the "NLP Best Practices" library as well as "NLP Utilities" library to build the classifier. These libraries also provided example templates which is referenced in this project.

Demo and Tutorial

Environment Setup

Sign up for Microsoft Azure. Create a subscription that allows you to use GPU instances. Student email get $100 free credit.


Microsoft Azure
Search resources, services, and docs (G+/)
fw9@illinois.edu
UNIVERSITY OF ILLINOIS - URBA...
Home
+ Add

Subscriptions

University of Illinois - Urbana

View list of subscriptions for which you have role-based access control (RBAC) permissions to manage Azure resources. To view subscriptions for which you have billing access, click here

Showing subscriptions in University of Illinois - Urbana directory. Don't see a subscription? Switch directories

My role: 8 selected

Status: 3 selected

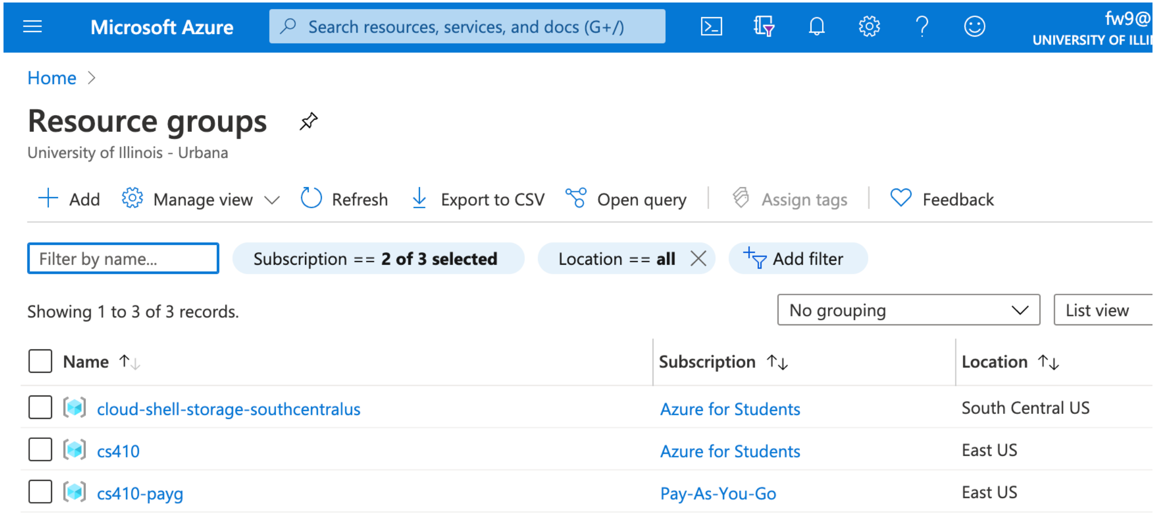
Apply

Showing 2 of 3 subscriptions ☑ Show only subscriptions selected in the global subscriptions filter

Search

Subscription name	Subscription ID	My role	Current cost	Status
Pay-As-You-Go	f5c4ac21-4b79-45ed-9d94-3bb35102f3...	Account admin	0.00	✓ Active
Azure for Students	e5229af5-0672-4af2-bf50-d6d9ca1389...	Account admin	Not available	! Disabled

Create resource group and ML workspace



Resource groups

University of Illinois - Urbana

Filter by name...

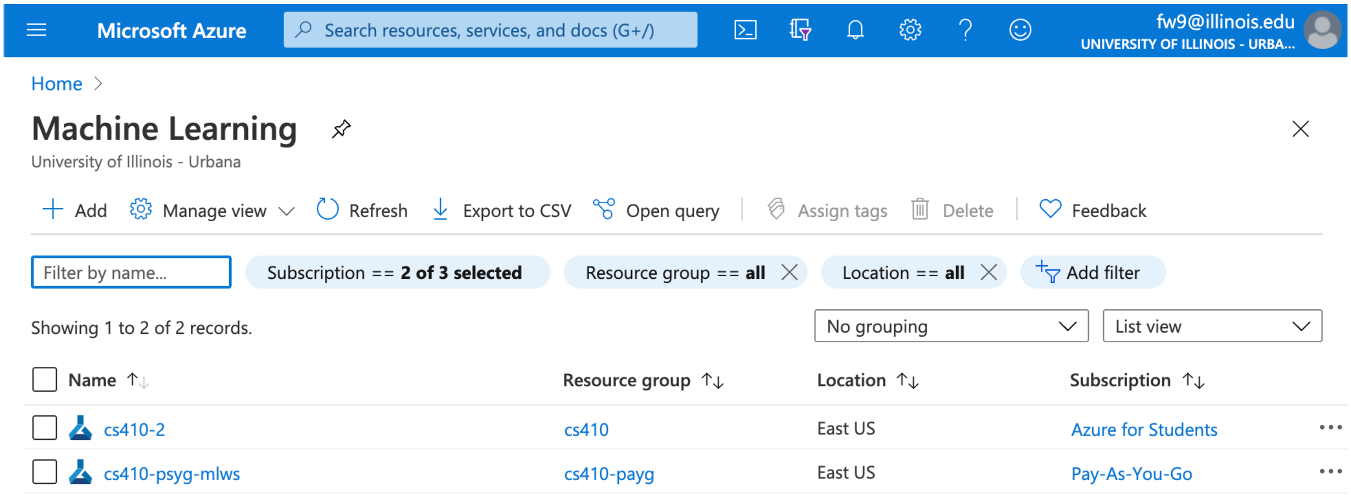
Subscription == **2 of 3 selected**

Location == **all**

Showing 1 to 3 of 3 records.

No grouping

	Name	Subscription	Location
□	cloud-shell-storage-southcentralus	Azure for Students	South Central US
□	cs410	Azure for Students	East US
□	cs410-payg	Pay-As-You-Go	East US



Machine Learning

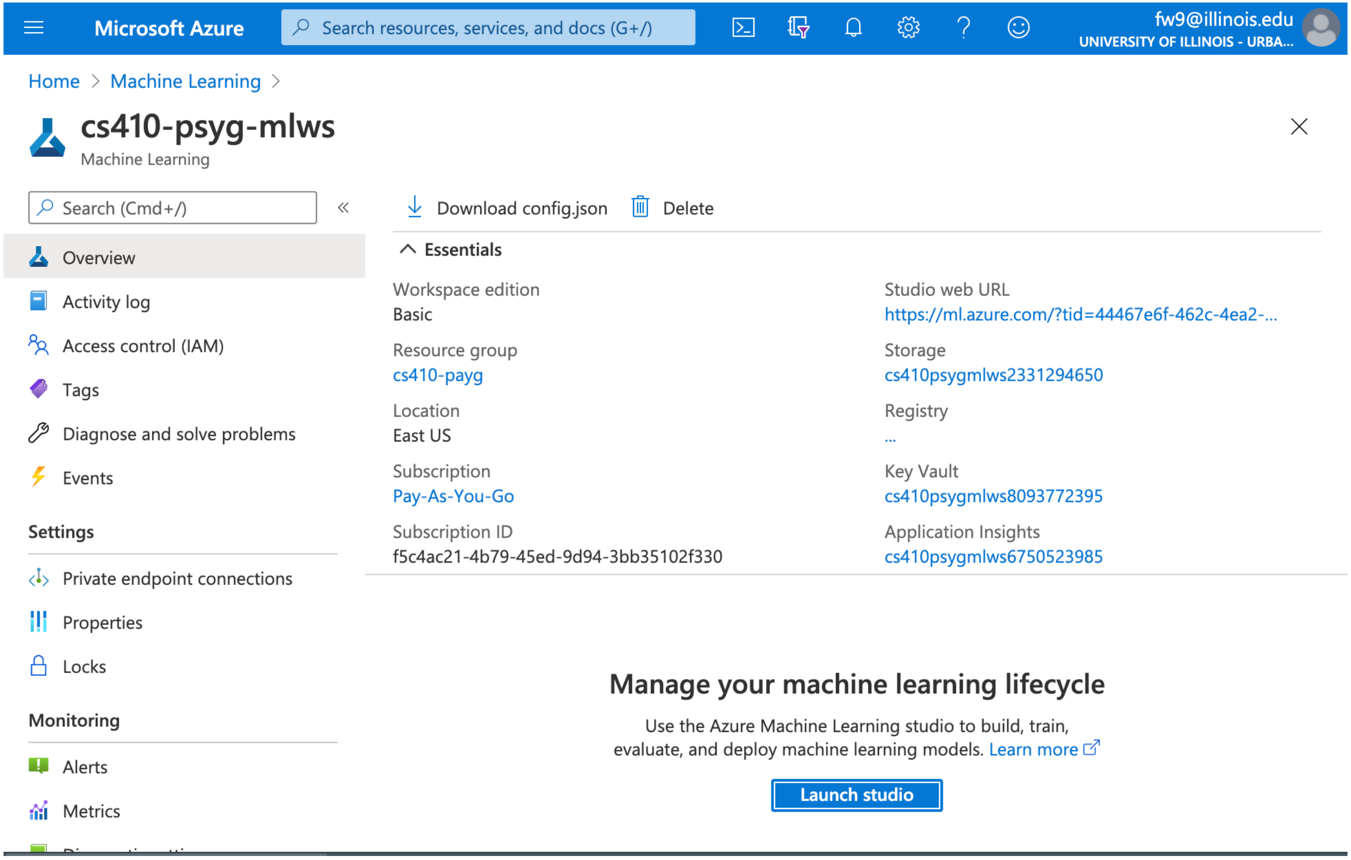
University of Illinois - Urbana

Subscription == **2 of 3 selected** | Resource group == **all** | Location == **all**

Showing 1 to 2 of 2 records.

Name	Resource group	Location	Subscription
□ cs410-2	cs410	East US	Azure for Students
□ cs410-psyg-mlws	cs410-payg	East US	Pay-As-You-Go

In the created workspace, launch ML studio.



cs410-psyg-mlws

Machine Learning

Essentials

Workspace edition
Basic

Resource group
cs410-payg

Location
East US

Subscription
Pay-As-You-Go

Subscription ID
f5c4ac21-4b79-45ed-9d94-3bb35102f330

Studio web URL
https://ml.azure.com/?tid=44467e6f-462c-4ea2-...

Storage
cs410psygmlws2331294650

Registry
...

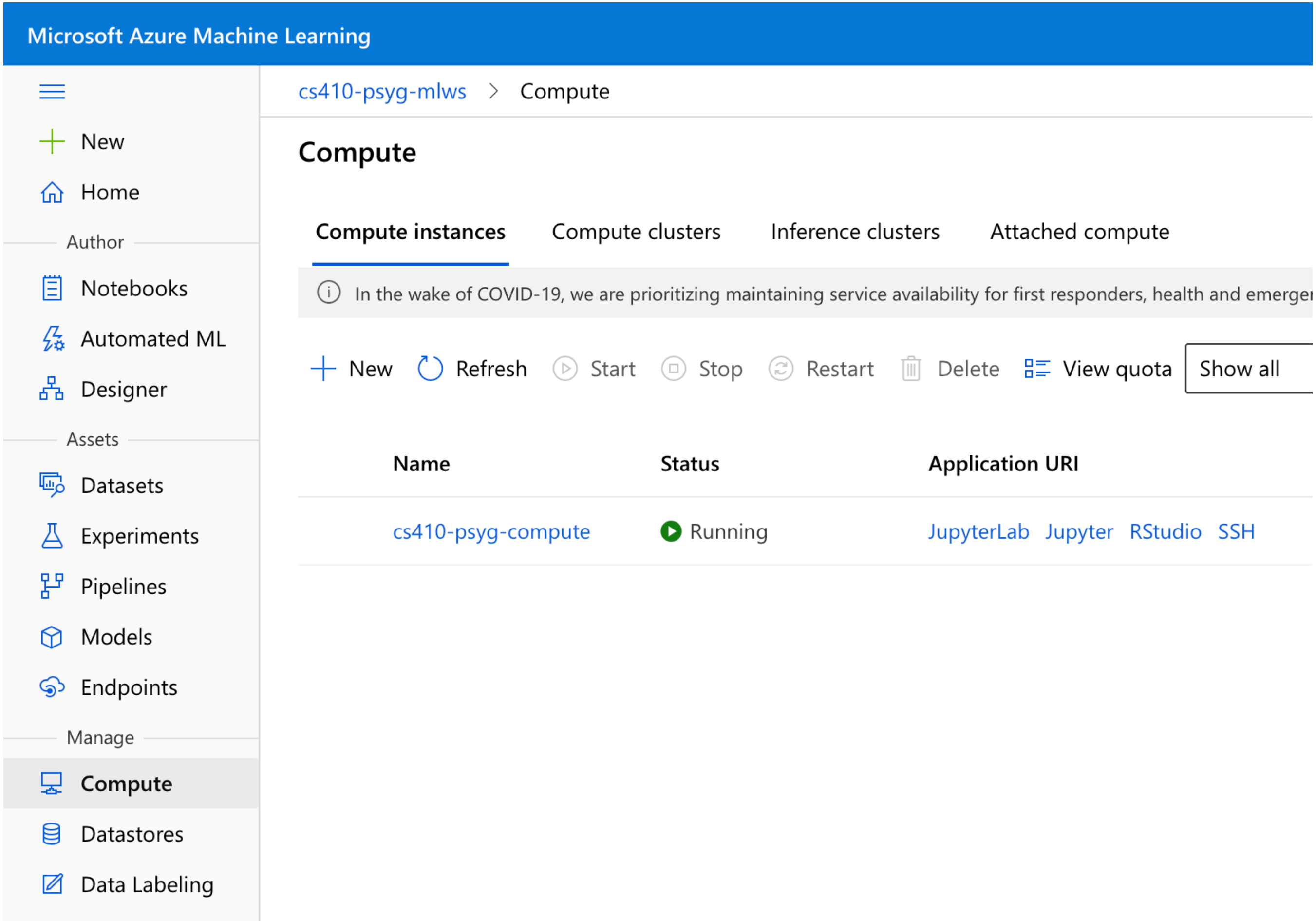
Key Vault
cs410psygmlws8093772395

Application Insights
cs410psygmlws6750523985

Manage your machine learning lifecycle

Use the Azure Machine Learning studio to build, train, evaluate, and deploy machine learning models. Learn more

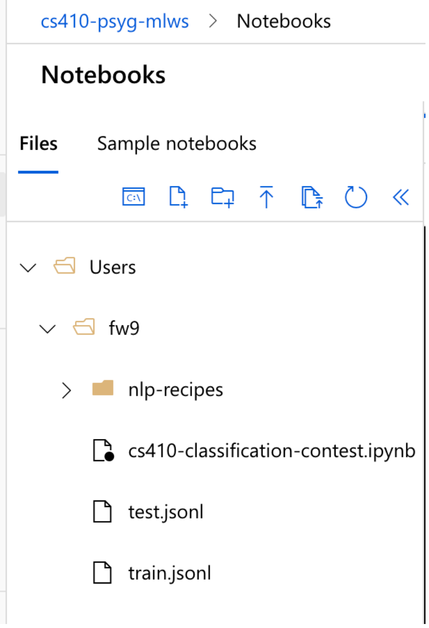
Create GPU compute instance and start.

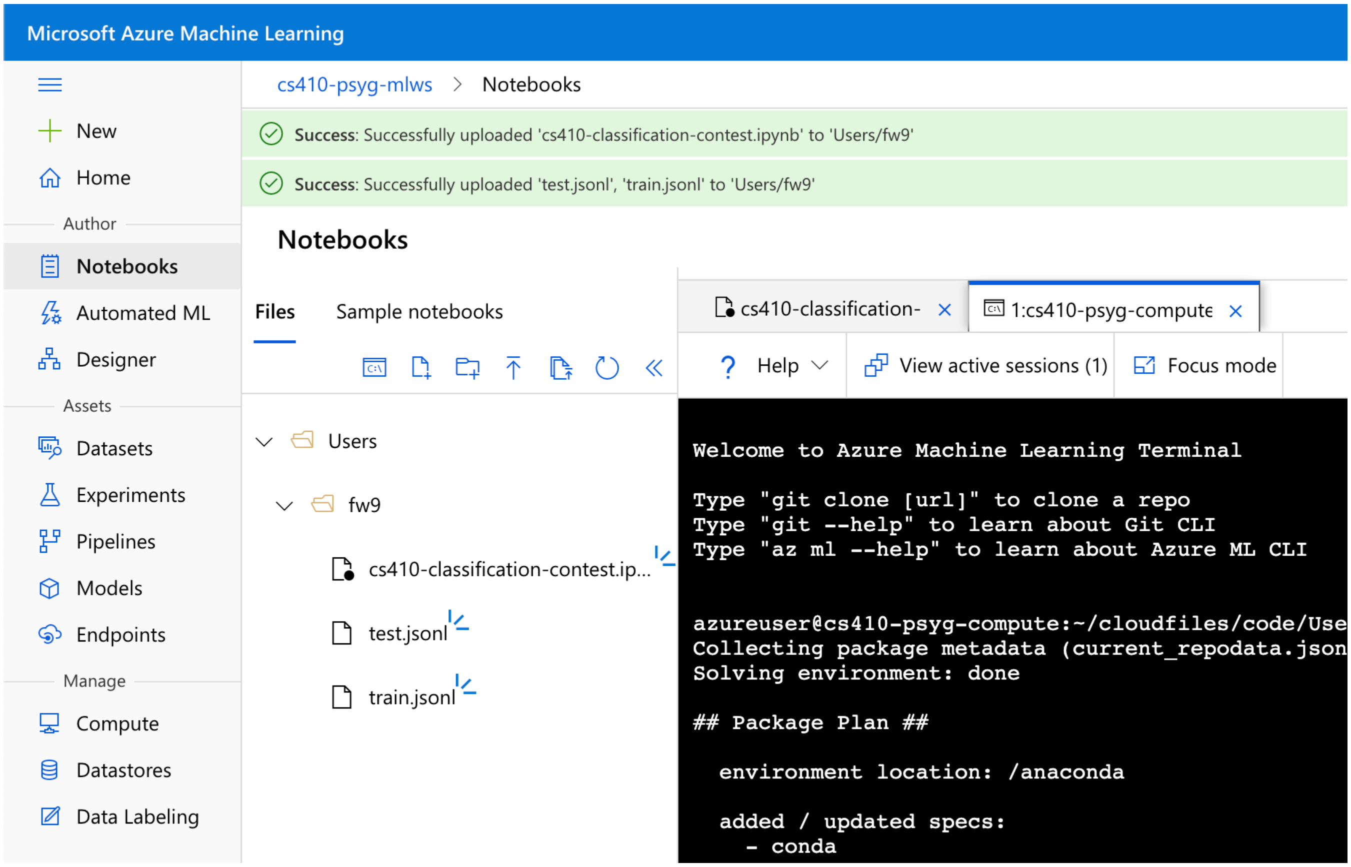


In the wake of COVID-19, we are prioritizing maintaining service availability for first responders, health and emerge

Name	Status	Application URI
cs410-psyg-compute	● Running	JupyterLab Jupyter RStudio SSH

Upload notebook file, data file and clone the NLP Library.





Success: Successfully uploaded 'cs410-classification-contest.ipynb' to 'Users/fw9'

Success: Successfully uploaded 'test.jsonl', 'train.jsonl' to 'Users/fw9'

```
Welcome to Azure Machine Learning Terminal

Type "git clone [url]" to clone a repo
Type "git --help" to learn about Git CLI
Type "az ml --help" to learn about Azure ML CLI


azureuser@cs410-psyg-compute:~/cloudfiles/code/Use
Collecting package metadata (current_repodata.json
Solving environment: done

## Package Plan ##

  environment location: /anaconda

  added / updated specs:
    - conda
```

Update Conda:

```
conda update -n base -c defaults conda
```

Generate conda env config:

```
cd nlp-recipes
python tools/generate_conda_file.py --gpu
```

Open nlp_gpu.yaml, update pytorch version from 1.4.0 to 1.5.1

Create Anaconda env

```
conda env create -n nlp_gpu -f nlp_gpu.yaml
conda activate nlp_gpu
```

Register this virtual env to notebook

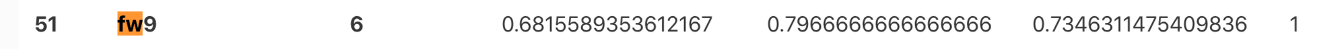
```
python -m ipykernel install --user --name nlp_gpu --display-name "Python (nlp_gpu)"

The environment setup is complete at this point.
```

Running the notebook

```
Open notebook and set kernel as ‘nlp_gpu’.
Run the notebook, and the result shows up in “answer.txt”
```

Intermediate output can be seen in the notebook. Performance passed the baseline.



51	fw9	6	0.6815589353612167	0.7966666666666666	0.7346311475409836	1