Deep Learning

Text Classification & Framework Comparison

Shishir Timsina, Michael Schafer, Wenhe Ma, Brendan Szewczyk

University of Pittsburgh Capstone Project

Project Goals

1. **Build, train, and test two text classification models built in PyTorch and Tensorflow, respectively.**
2. **Compare these two frameworks and develop a decision matrix to help CGI understand where and when they might use PyTorch or Tensorflow in the future.**

Tools and Setup

In this project, we worked with **AWS**, and more specifically **Amazon Sagemaker** as a cloud solution for running our trained models. We worked primarily with **Python**, using tools like **numpy** and **pandas** for data manipulation. Our code was hosted in a shared **GitHub** repository. Most of our code was written in **Jupyter Notebooks** for added clarity.

We were provided a large dataset of news articles to use as a training set, and a holdout test set to make predictions on our data and evaluate the performance. We were tasked with distinguishing between five possible categories:

- Tech
- Business
- Sports
- Entertainment
- Politics

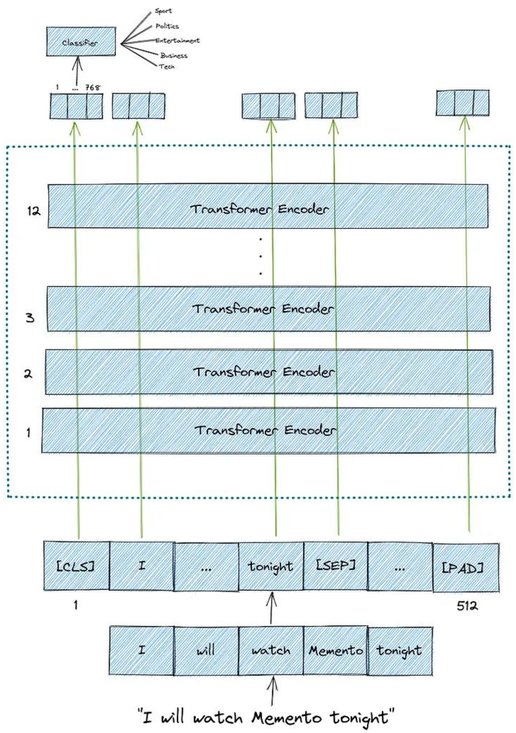
A sample of the larger dataset can be seen below:

	category	text
0	tech	tv future in the hands of viewers with home th...
1	business	worldcom boss left books alone former worldc...
2	sport	tigers wary of farrell gamble leicester say ...
3	sport	yeading face newcastle in fa cup premiership s...
4	entertainment	ocean s twelve raids box office ocean s twelve...

Model Development and Training with BERT

For many of us in the group, this was the first time we'd worked with these frameworks, so the model development and training processes were carefully laid out in the form of several milestones across the semester to help us ease into the work.

We built both models using a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model. This simplified the model building process and allowed us to control for any differences between the two frameworks.



BERT uses the same transformer technology first published by Google in their 2017 paper *Attention Is All You Need* [1]. This is also the **T** in ChatGP**T**. BERT was developed by Google in 2018, and we used a trained model provided by *Hugging Face*. The graph above shows the process of *"tokenizing"* the input into bite-sized pieces, pumping them through transformers, and then classifying the overall input.

Both models were built with BERT as a basis. New Jupyter notebooks were created as endpoints for accessing and serializing both models.

API and Explainability with LIME

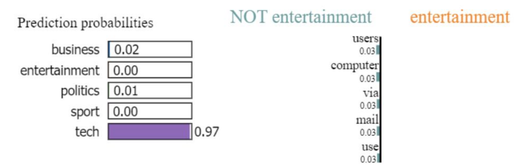
In the next phase of our research, we created a set of APIs using Python's FastAPI library, and implemented LIME (Local Interpretable Model-Agnostic Explanations) to better explain what the models are doing under the surface.

The **APIs** allowed us to quickly make predictions with each of the models individually, giving us an idea about how they performed in a real-world setting on new inputs. Each API takes a POST request with a data query that the user sends, and based on the input, will return a prediction using the models we trained. The API was dockerized and deployed in an ECS cluster in AWS.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 104ms/step
hobbit picture  four years away  lord of the rings ...
Actual label:entertainment
Predicted label: entertainment

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
game firm holds  cast  auditions video game firm b ...
Actual label:tech
Predicted label: tech

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
clarke plans migrant point scheme anyone planning  ...
Actual label:politics
Predicted label: politics

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
radcliffe will compete in london paula radcliffe w ...
Actual label:sport
Predicted label: sport

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
serena becomes world number two serena williams ha ...
Actual label:sport
Predicted label: sport
...
us tv special for tsunami relief a us television n ...
Actual label:entertainment
Predicted label: entertainment
```

an example of predictions being made – not the API interface

LIME gave us access to a more robust understanding about what specific words were being used by the model to make decisions about to classify the inputs. This can be seen in the image below, along with an interface that more closely resembles the one we used in our API:



Text with highlighted words

security warning over fbi virus the us federal bureau of investigation is warning that a computer virus is being spread via e-mails that purport to be from the fbi. the e-mails show that they have come from an fbi.gov address and tell recipients that they have accessed illegal websites. the messages warn that their internet use has been monitored by the fbi s internet fraud complaint center. an attachment in the e-mail contains the virus the fbi said. the message asks recipients to click on the attachment and answer some questions about their internet use. but rather than being a questionnaire the attachment contains a virus that infects the recipient s computer according to the agency. it is not clear what the virus does once it has infected a computer. users are warned never to open attachment from unsolicited e-mails or from people they do not know. recipients of this or similar solicitations should know that the fbi does not engage in the practice of sending unsolicited e-mails to the

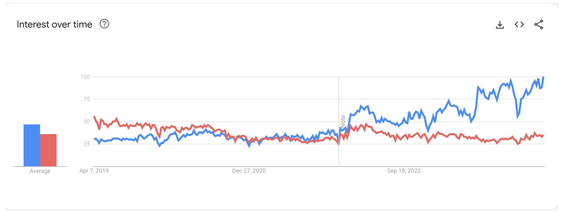
PyTorch vs Tensorflow

The major goal of our project was to take these frameworks and provide CGI with a comprehensive explanation about when to use each framework. We stumbled upon some major points of consideration in our personal exploration for both frameworks:

Tensorflow:
- Smaller file size and training times in model development
- Faster predictions
- Longer history of standard use and scalability in industry use
- Integrated better with *LIME*

PyTorch:
- Longer training times and larger files
- Had some difficulty integrating *LIME* as described by the sponsors
- More *"Pythonic"* in style; easier for longtime Python programmers to pick up

Ultimately, though, both of these models had a similarly high level of accuracy and both are being used in production environments. The general consensus upon further research is that PyTorch is currently in vogue and picking up steam in academic environments. It seems to be more easily customizable if you're building non-standard, custom machine learning models.



This google trends graph does a good job of showing how interest in PyTorch has eclipsed Tensorflow in recent years.

Both frameworks are perfectly capable of achieving the same levels of performance, and it will come down to a matter of preference for CGI's development team and hiring priorities.

Using PyTorch will certainly keep CGI at the bleeding edge and allow them to hire on experienced developers in a technology that appears to be gaining the edge.

If they're working with lots of existing Tensorflow code that's being scaled in quite large ways, and their development team is already working with this framework, there isn't a major reason to switch over to PyTorch at the moment, as skills between the frameworks translate with some adjustment.

Acknowledgements

BERT Graph taken from:
https://towardsdatascience.com/text-classification-with-bert-in-pytorch-887965e5820f

[1] Link to the paper *Attention is All You Need* published by Google in 2017: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

The likeness of Bert, the Sesame Street character, was used here under the fair use doctrine of the U.S. copyright statute. We do not own the likeness of Bert.