Aiden Kempen / Lucas Pacheco

Dr. Emily King

DSCI 478

21 February 2024

Contradictory, My Dear Watson - Kaggle Competition

Introduction

Natural language processing is a branch of machine learning that focuses on teaching computers how to understand human written language. Recent years have seen tremendous progress in the ability of models to classify, summarize, and even generate text. But is the statistical pattern recognition of these models enough to make the leap to logical inference? If one sentence says the sky is blue and another says it's raining, can the computer know that these are logically contradictory? Natural language inference (NLI) is the branch of machine learning that tries to address this question. This introductory Kaggle competition is a quick dive into the world of NLI, with the goal of building a model to detect logical contradiction in text data. We successfully fine tuned various pre-trained NLP deep learning models to the dataset at hand. Our best models achieved accuracies around 60% and the worst were doing no better than guessing with random chance.

For the rest of the paper, we will first explore the problem at hand and describe various approaches to modeling. Next, we will analyze our results in the discussion section and describe possible steps for improvement. Finally, we will conclude with a key takeaways section, which summarizes our learning across the entire project.

Problem Description

Our main task with the Contradictory My Dear Watson Kaggle project was to detect contradiction and entailment in multilingual texts using TPUs. More in depth we had a dataset with two sets of sentences all in different languages. One group of sentences were premises and the other group were hypotheses. Our goal was to determine whether these two sentence groups were contradicting, neutral, or entailing for each index. We could use Kaggle's free TPUs to speed up the process if we needed to. There were many ways we wanted to attack this problem, but not all were successful or time efficient.

At first we brainstormed comparing the sentences using vector semantics and cosine similarities. We thought since the sentences would be vectors we could compare their directions to determine whether they contradicted, were neutral, or entailing. BERT had a pretrained model named Sentence BERT where you could compare the cosine similarity between the sentences that were encoded. After doing this for a while we realized that this pretrained model was extremely helpful for determining whether sentences had similar or unsimilar words within them, but was not useful for our main goal of determining the meaning of each sentence and comparing them. This was going to be a much more complicated task.

We then looked into what it takes to build a natural language processing model, and realized that they are extremely tedious and time consuming to create. This option may have given us the best results in this project, but we didn't have the time, resources, or knowledge to create one from scratch. However, we did find that there were machine learning frameworks that had pretrained language processing models available to the public, that we could use and potentially build upon. The main ones we found were BERT, RoBERTa, DistilBERT, and XLM-RoBERTa which were all downloadable through Kaggle and the Keras package which was extremely helpful since we were working within Kaggle. We will go more into detail on each of these frameworks later in the paper.

Methods

To start we got the Contradictory My Dear Watson training and test data by loading it through the Kaggle csv using pandas. Around 60% of the sentences in the training data set were written in english. The rest of the sentences were in fourteen different languages that took up about 3% of the training data each. Next we needed to format the data in a way that was suitable to input into the machine learning frameworks we were using. We struggled to find the specific ways to input the hypothesis, premise, and label variables into the machine learning frameworks so we used the BERT tutorial in the kaggle notebook to help us format the data.

We first created a helper function that would split the data frame into the two sentence groups and the labels corresponding to them. We then used Tensorflow to create a dataset that took the sentence values and label values and made them their own objects. After this we created a new training set and a validation set from the Tensorflow dataset with a validation split of 0.15. Finally, we preprocessed both the training set and validation set by using the map() function on the split data while also using a batch size of 16. There were also functions added to the end of the preprocessing that were meant to tune performance.

Now that we had the data set up in a usable preprocessed way for the machine learning frameworks we could start building models. The first model was built with regular BERT to establish a baseline. BERT is a language representation model that is designed to pre-train representations of text by looking at the text from both the left and right context. This was perfect for this project because we needed a model that would be able to fully learn the representation of a sentence and compare it to the representation of another sentence.

We created the first BERT classifier using the pretrained bert_tiny_en_uncased model which was trained on English Wikipedia and Books Corpus. Then we fit this classifier to the preprocessed training and validation datasets we created earlier, and got a validation accuracy of around 44%. We weren't quite happy with this accuracy so we decided to create the same BERT classifier except with a higher learning rate and ended up with around the same validation accuracy. To try and raise the accuracy we created a third BERT classifier with the pretrained bert_base_multi model trained on Wikipedias of 104 languages. Once we fit this model we got a higher validation accuracy rate of around 58%.

Next we continued to try and improve the model by using RoBERTa which is an optimized version of BERT that was trained to more carefully measure impacts of certain parameters and training data size. The RoBERTa classifier we created used the pretrained roberta_base_en model that was trained on English Wikipedia, BooksCorpus, CommonCraw, and OpenWebText. We then fit this model using the preprocessed training and validation data and ended up with a validation accuracy of around 61%.

Hoping for possibly higher accuracy we used DistilBERT which is a distilled version of BERT which is smaller, faster, cheaper, and lighter. We then used the pretrained disitil_bert_base_multi model which was also trained on Wikipedias of 104 languages. When we fit the DistilBERT model we got a validation accuracy of 54% which was a decrease in validation accuracy from some of our other approaches.

Finally, we used the final machine learning framework XLM-RoBERTa which is a multilingual model also based on BERT. We created the classifier using the pretrained xlm_roberta_base_multi model that was trained on CommonCrawl in 100 languages. Once we fit the model we received a final validation error of around 33% which was our worst validation accuracy so far. We had used XLM-RoBERTa thinking it may have a chance at performing the best, but overall was extremely underwhelming in its results.

Discussion

We achieved a strikingly moderate test accuracy of 62% with our best model, RoBERTa, yet it placed us relatively high on the Kaggle rolling leaderboard at 14th place. The most interesting part about this result is that this model had been trained only on english text! Our task involved NLP with 15 different languages, and our best model was trained only on English. How is this possible?

First, we must understand the distribution of the languages across the dataset. English takes up approximately 57% of the data, while the other 14 languages make up about 3% each. Hypothetically, a model that only knows English could classify all of the English observations correctly and be approaching 60% accuracy. Moreover, this model still has a ⅓ chance of correctly guessing the other languages, giving it another ⅓*42 = 14% accuracy. It's likely that our RoBERTa model was the best model simply because it was far better than the others at understanding the English portion of the dataset. But we'd still expect a multilingual model to perform better, especially since they have near identical architectures to their English-only relatives. We hypothesize that the multilingual models do not have a *strong* enough understanding of any of the languages - they are the NLI equivalent of jack of all trades, master of none. The classification task at hand is very difficult, and likely requires a sufficient amount of architectural complexity and loads of data to pick up on the statistical patterns governing logical inference. It’s entirely possible that we need an order of magnitude more data to properly fine-tune the multilingual models to achieve a higher accuracy.

Future Improvements

To improve the model going forward, we are considering using external data to help train our model. The 12,000 observation dataset provided on Kaggle is probably a small subset from the much larger Stanford Natural Language Inference (SNLI) dataset containing 570,000 rows (the data description on Kaggle links to TensorFlow datasets which then contains the SNLI dataset). Given sufficient computational resources, we could train our large multilingual models on this dataset to improve the inference ability. We hypothesize that this approach is what allowed some teams on the leaderboard to achieve much higher accuracies, in the 80-90% range, as using external data for model training is not uncommon for Kaggle competitions.

Key Takeaways

In this project, we learned the state-of-the-art approaches to solving natural language inference tasks with deep learning. We learned how raw text data must be tokenized and vectorized into a format that neural networks can interpret. We learned about the strengths and weaknesses of various model architectures. One key takeaway is that Natural language inference must be solved with sequential models, not bag-of-words models. The relative word order and grammatical structure is critical in determining the logical relationship between two sentences. Another takeaway is that properly fine-tuning a base model can take lots of data and computational resources. Additionally, we ran into memory errors when using very large data in the Kaggle workspace, so we now know the limitations of working on Kaggle notebooks. Most importantly, we learned that with the right data and architecture, we can teach a computer to recognize when two sentences are logically contradictory, at least with 62% accuracy!

Sources

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). *Bert: Pre-training of deep bidirectional Transformers for language understanding*. arXiv.org. https://arxiv.org/abs/1810.04805

Huilgol, Purva. "Top 4 Sentence Embedding Techniques Using Python." *Analytics Vidhya*, 13 Sept. 2023, www.analyticsvidhya.com/blog/2020/08/top-4-sentence-embedding-techniques-using-python/.

Liu, Yinhan, et al. "Roberta: A Robustly Optimized Bert Pretraining Approach." *arXiv.Org*, 26 July 2019, arxiv.org/abs/1907.11692.

Sanh, Victor, et al. "Distilbert, a Distilled Version of Bert: Smaller, Faster, Cheaper and Lighter." *arXiv.Org*, 1 Mar. 2020, arxiv.org/abs/1910.01108.

Team, Keras. "Keras Documentation: Kerasnlp Models." *Keras*, keras.io/api/keras_nlp/models/. Accessed 20 Feb. 2024.