Pykédex: Using Python to Predict Pokémon

Brendan Matthys

Abstract

This document is my final project for SI 630: Natural Language Processing

The Pokémon universe is an ever-growing group of unique creatures. When the series started out in 1996, there were 151 Pokémon, but ten years later, that number had nearly tripled (Wikipedia, 2023). Nowadays, there are over one thousand Pokémon, which makes keeping track of all of them very difficult.

Imagine you are walking through the forest and all of a sudden, you run into a Pokémon and don't recognize it, but you want to know what it is. This project is here to help that issue.

The primary model involved here is a Bert For Sequence Classification Model. There were multiple variations of said model created, with the primary variation between each model being the data that it was being trained on. Between each new model, a new component was added to the data set. The first model was just scraped Pokédex data and image-to-text descriptions of each Pokémon, the second was the same data as the first, but ran for more epochs, the third added entries from a Pokémon Wiki, and the fourth model included everything, but dropped stop words.

The main results were that compared to a baseline of randomly picking a Pokémon, the models performed exceedingly well. The best model included all the data and dropped the stopwords. For the Bulbapedia data, it performed almost 150 times better, and for the user-annotated data it performed almost 20 times better.

The implications for this are that some NLP-focused models are strong enough to produce accurate results for classifiers with over 1000 levels. This data was also limited, so it also showed the power of text augmentation in data creation.

1 Introduction

The goal of this project was to create an algorithm that can take in a text description of a Pokémon and accurately predict which Pokémon the text is referring to. This model could be helpful for those who are picturing a Pokémon, but cannot remember its name. Inevitably, this would end up being a Pokédex, but in a broader sense, this concept could be extracted from fiction and applied to reality, such as with real animals.

This projected implemented a Bert-based model for sequence classification in order to create a classifier for 1008 levels. I think one thing I did well here was include enough data and diverse enough data for the model to sufficiently train on.

There has been a classifier created for the Pokémon series in the past. However, that was not to classify the Pokémon themselves, but just their type. Classifying 18 types of Pokémon is a much lower level task than classifying between 1008 Pokémon themselves, so a different type of model was necessary to use here. My situation here was harder to achieve ideal results, but that just made getting the model to work that much more of a motivation.

I think that those who enjoyed this series in the past and those who still enjoy it in the present would want to see this. I think it would matter to different groups of people because it would be nostalgic and remind them of the creativity and enjoyment that a simple show and trading card game created for them in their childhoods.

One thing I learned from this project was that the quality of the data one has makes a difference. The variation from the different models was primarily the data that was being used and the epochs being run, so I think it was very interesting to see how the F1 score changes between models.

2 Data

The first section of data for this project was going to be scraped off a Pokemon Database website. The linked web page shows a list of all known Pokemon, and each Pokémon has its own page. On their page, each Pokémon has multiple "Pokédex" entries, where multiple descriptions from each of the games are written about them.

The first step of my data collection process was to use BeautifulSoup [1] to get each entry for each Pokemon, and make a dataframe containing all the entries. This in itself resulted in 12,596 rows of data, with the most prevalent Pokémon in the dataset being Pikachu. Since there are 1,008 different Pokemon in the data frame, it makes sense that this isn't enough data here for the model to satisfactorily learn about each Pokémon. There are many ways around this, but the primary method I used to get around this was data augmentation.

Using the 'nlpaug' library [2] in Python, I was able to slightly augment each description from the Pokédex 10 unique times. Now, instead of having 13,000 rows of data at my discretion, I was able to utilize around 140,000. This same technique was performed as well for both of the data sources I am going to further mention below.

One other means of collecting unique data was focusing on what the Pokémon look like. I used a model created by Salesforce called BLIP [3] for image to text data creation. To gather all the images, I used the Bing-Image-Downloader [4] library in Python, and then after checking and cleaning all the images to make sure they looked accurate, I ran it through the image to text algorithm. This created 1,008 new data points. To generate more training data, I also used the nlpaug library on these outputs. One concern here is that if a possibly inaccurate image interpretation was rephrased 10 times, it has potential to negatively affect the model.

The final data source used for model was the Pokémon Wiki from the Pokémon Fandom website. This source had a full Wikipedia-esque page for each and every Pokémon. I decided that if someone was to describe a Pokémon they see in the wild, they would most likely describe what it looks like and how it acts. Therefore, I only cared to scrape the "Physiology" and "Behavior" sections of each Pokémon. There were sections about them being featured in the video games, but we can't learn anything useful for the Pykédex from that. This source also required extensive usage of the BeautifulSoup library.

[1] https://pypi.org/project/beautifulsoup4/
[2] https://pypi.org/project/nlpaug/0.0.5/
[3] https://github.com/salesforce/BLIP
[4] https://pypi.org/project/bing-image-downloader/

3 Related Work

I found myself in deep thought when considering the plausibility of creating an NLP solution to a problem of this magnitude. I saw some papers regarding using something called BERT for large scale multi classification problems. In the following paper, Nugroho et al. (2021) compares BERT with another software called Spark NLP. Bert, although slower, is incredibly accurate, which is what I'm looking for in such a high class model. Therefore, I think Bert is the model I will choose to implement here.

The second part of my project is text generation based on the Pokemon that the model selects. This was something I had no idea on how to even start, so the following article was helpful. In their article, Iqbal and Qureshi (2020) attempts to utilize different types of models for text generating and assesses which is the best for their data. They landed on the Variational Auto Encoders as being the best generator for discrete text. VAE is an unsupervised learning method that works with even unlabelled data, and is very accurate. This could be really useful for an extension of this project that I plan on implementing.

In their paper, Nguyen (2021) tests different metrics that are unique to evaluating text generation. In this paper, they try Corpus BLEU, Self-BLEU, and NLI. They finally come to the conclusion that the best evaluation metric for text generation is Self-BLEU. This is useful because as someone who knows nothing about how to evaluate a text generation, this is a really good starting place. The dataset the researcher used to generate the text was different to mine in terms of context, but this is still useful information for text evaluation.

After doing more research on the correct metric, I stumbled into an article written by prestigious AI/ML writer Tuychiev (2021). Bex had agglomerated, from many sources, experienced data scientists reflecting on their experiences with high classification evaluation, and they collectively suggested that F1 score was more effective if your data was more imbalanced. However, if one was working with balanced data, then ROC-AUC score was

the ideal metric. My data is imbalanaced so I went with the F1 score to assess my model accuracy.

Once I found the metric I thought was best, it was important to solidify which model to use. Luckily, the article from Lu et al. (2022) et al. was very helpful in influencing my decision. In their paper, they compared various deep learning models' performances with data that had high class imbalance. They found that BERT performed just fine compared to the others, but the training and tuning was computationally costly. Luckily, I had access to a GPU for this project, so that wasn't an issue and I proceeded with BERT.

4 Methods

As mentioned in the Data section, the entire dataset that I had went through a data augmentation overhaul, and by the time the final model was being tested, there were over 300,000 rows of data - sufficient for Bert and PyTorch to train on.

Before conducting any training, we needed to use a Class Converter to convert the Pokémon names themselves to labels. For prediction's sake, it was easy to convert back as well.

In terms of modeling, I used the BertForSequenceClassification [5] model and used the "bert-base-uncased" pretrained model as well as its tokenizer. From my research, this was the best model to use for this specific NLP task. I also used the AdamW optimizer for this project, as it is easy to use and very computationally efficient.

The first model that was trained on mostly Pokédex data. This was approximately 140,000 rows of data after augmentation. This data was initially what I thought would be very good to train on, as it has the basics of what a Pokémon's appearance is, and how they behave. However, this data was biased towards mainstream Pokémon, as 15% of the Pokémon in this initial scrape only had two entries or less about them. This first model also included the text-to-image data.

The second model was loaded from the pretrained version of where the first model stopped, and ran for an equal amount of epochs. It featured both the Pokédex data and the image-to-text data. This was trained using the same method as above.

The third model started from the pretrained model from the second model, but contained double the data as the first model, since the Pokémon Wiki data was included. However, this data was more illustrative for the lesser represented Pokémon in the earlier models, so it resulted in a much stronger model.

Figure 1: Pykédex when prompted with "Ice Cream Cone

The fourth model used the exact same data as the third model, however it was processed differently. For this model, the model was retrained with the stop words removed. The reason here is that the stopwords didn't have much practical use and could be interfering with the model's ability to predict. If this was the case, the only way to find out was by creating another version of the model.

5 Evaluation and Results

Based on the high number of classes in the dataset for the model, I reasoned that it should have three guesses as to which Pokémon was the correct one. There are 1008 classes here, and Pokémon that share strongly similar traits, so I thought it was only fair to have three guesses at it.

In the example above, the Pykédex model is prompted with "Ice Cream Cone". It predicts three Pokémon, all of which could resemble an ice cream cone. Giving the model three chances is beneficial in a scenario like this where it's doing the right thing, but there are just multiple correct answers due to the sheer number of Pokémon out there. Penalizing for guessing "Vaniluxe" over "Vanillish" here would be completely unreasonable.

In addition, the models were compared against a "baseline" that chose three random Pokémon for each row of validation data.

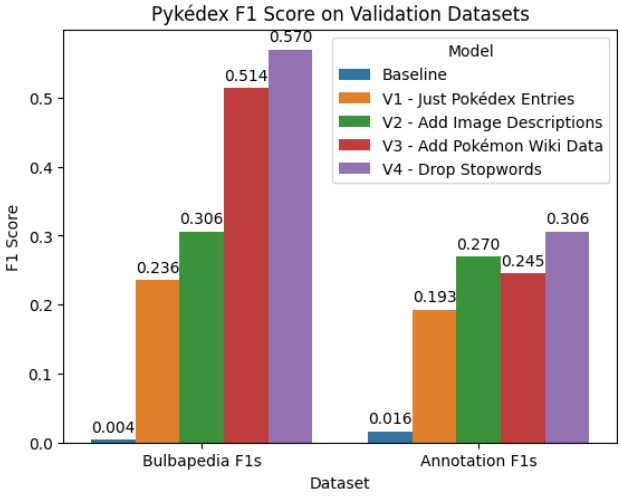
In terms of validation data, I used two separate datasets: Bulbapedia data and Annotation data.

[5] https://pypi.org/project/bert-for-sequence-classification/

Bulbapedia is another online Pokémon encyclopedia, one of the largest data sources that I did not train the model on. Specifically, the 'Biology' section for each Pokémon is what I was interested in, as that is what would be visually observant to someone who is looking at said Pokémon. For many Pokémon, this section of the Wikipedia gives a little too much information than would be helpful, such as its representation in other media, or the origin of its name. Therefore, only the first paragraph of the Biology section was used in this case. The justification for using this as a data set is that it is illustrative for each Pokémon, and verbally different than the Pokémon Wiki. Therefore, it presents to be a perfect validation dataset.

However the Pykédex, if used in a real scenario, wouldn't always receive perfect inputs because it is being fed a human interpretation of what the Pokémon looks like. Therefore, I felt it necessary to test with human annotations of the Pokémon images that were processed through the image-to-text algorithm. I had 15 people each give me very descriptive entries for 10 randomly selected Pokémon.

For both of these validation datasets, an adjusted F1 score was calculated. It was adjusted in that it was recorded as a success if the true label was predicted in the top 3 predictions and a failure otherwise.



The adjusted F1 scores were calculated, and all of the models performed exceedingly well for all models as compared to the baseline. All of the models also performed better on the Bulbapedia data than on the human-annotated data. This is most likely because humans can approach describing a particular Pokémon in a number of ways. Some used paragraphs, while others can use few words or a single sentence. The Bulbapedia data consistently used one paragraph per Pokémon, and that's why these two results may have diverged as such. I also find it interesting how the models get more an more accurate as the version numbers progress, with the model that has all data and excludes stopwords being the most accurate between both validation sets. I would expect this, as many of the descriptions had a high percentage of stop words, and this could have potentially slightly thrown the algorithm off beforehand.

6 Conclusion

From this project, I learned it is possible to create an accurate classifier for high class data. Initially, I didn't think this was going to be nearly as accurate as it was - not because of the technology, but because of the extreme number of classes.

The model that performed the best against both the Bulbapedia data and the human annotated data was the one that included the augmented Pokédex entries, the augmented image-to-text captions, and the augmented Pokémon Wiki Physiology and Behavior data, without any stopwords. The Bulbapedia data was more consistently thorough for each Pokémon than the human annotated data, and that proved to generate much better F1 scores.

If you would like to look at the code, you can find it here.

7 Other Things I Tried

NLP is a field where trial and error is very commonplace. Failure is expected at some parts, but we are always hoping for a light at the end of the tunnel. My failure was when I tried to create these models on my local computer. The models would not run at all, and I felt like I had no hope in getting this project up and running. My computer really didn't like the transformers libraries, and when I finally got it to work, the run time was much much longer than I was willing to bear. Luckily, I found the Great Lakes cluster, and was able to get most of the models up and running before the traffic got too hectic.

8 What You Would Have Done Differently or Next?

If I could have done this project again, I definitely would have added another NLP component to this project: text generation. I think that generating text for whichever Pokémon is predicted would

have been an interesting feature to add to the GUI, but I didn't quite have enough time to add that into this project.

I think it definitely would have been feasible though. Since my results made sense most of the time, it would have been reasonable to have a text generated about the predicted Pokémon.

9 Acknowledgements

I would like to thank Professor David Jurgens for inspiration on how to generate more data. I had never considered data augmentation before this. Being able to synthetically generate more reliable data has been very helpful, so thank you David for turning me in the right direction.

References

Touseef Iqbal and Shaima Qureshi. 2020. The survey: Text generation models in deep learning. *Journal of King Saud University-Computer and Information Sciences*.

Hongxia Lu, Louis Ehwerhemuepha, and Cyril Rakovski. 2022. A comparative study on deep learning models for text classification of unstructured medical notes with various levels of class imbalance. *BMC Medical Research Methodology*, 22(1).

An Nguyen. 2021. Language model evaluation in open-ended text generation. *arXiv preprint arXiv:2108.03578*.

Kuncahyo Setyo Nugroho, Anantha Yullian Sukmadewa, and Novanto Yudistira. 2021. Large-scale news classification using bert language model: Spark nlp approach. In *6th International Conference on Sustainable Information Engineering and Technology 2021*, pages 240–246.

Bex Tuychiev. 2021. Comprehensive guide to multiclass classification metrics. *Towards Data Science*.

Wikipedia. 2023. List of Pokémon — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=List%20of%20Pok%C3%A9mon&oldid=1150776732. [Online; accessed 19-April-2023].