Automated Personality Type Prediction using Myers-Briggs Type Indicator

Elizabeth McLaughlin - eis29@drexel.edu - ID 11764593

Abstract My focus is using Deep Learning tools to improve upon previous attempts at classifying personality type from written text. I am using the Myers-Briggs Personality Type Indicator, which is one of the more widely used today for practical applications that range from hiring to counseling. I test a variety of models to find an optimal approach. In the end, I was able to outperform the previous works with a binary classification model for the 4 categories of personality, using a sigmoid activation function and cross entropy as my loss. Average accuracy across the 4 categories is 84%.

Background

MBTI is an extension of the theories developed by psychiatrist and psychoanalyst Carl Jung, who believed that what appears as random behavior is actually drawn from differences in the way people prefer to use their mental capacities. He narrowed this down to the ways in which individuals perceive the world around them, and how they formulate conclusions within it in the form of judging. His opinion is that for certain categories of choice within these, behaviors become quite predicable. MBTI is an introspective self-report attempting to classify individuals into 16 distinctive personality types. The test is one of the more widely used psychological instruments today. It's currently applicable in many areas where a thoughtful approach to personality is taken, such as in hiring to determine if a person is a good fit, for coaching, counseling, or educating. Figure 1 shows the 16 personality types, which you can see is essentially a combination of 4 binary categories.

FIGURE 1. *Personality types key (Amirhosseini and Kazemian 2020)*

The quad concatenated for each of these binary categories is what makes up each of the 16 types. As stated above, an individual is assigned a type via an introspective questionaire. Research on extending this to textual data is unfortunately scarce, but the uses are exciting should it be found possible. For example, using machine learning methods on social media postings would open up a new channel for targeted marketing, dating website algorithms, and more.

Earlier approaches did not instill confidence that this extension is possible. See below by Rayne Hernandex and Ian Scott Knight. They built a recurrent neural network in hopes of capturing nuances in the text. They tested various layers in keras, and found their best results with a dense layer network using the sigmoid activation function. Their model is 4 separate binary classifiers, one for each of the categories in figure 1. Their accuracy is only slightly better than chance, as seen in figure 2 below.

Type	I/E	N/S	F/T	P/J
Accuracy	54.0%	52.9%	57.8%	52.9%

FIGURE 2. *Accuracy Results - 4 Binary Classifiers (Hernandex and Knight 2017)*

Later, Amirhosseini and Kazemian picked up the work and attempted to improve upon the scores by introducing extreme gradient boosting. Their results are shown below in figure 3. I stated in my project proposal that I intended to take the learning's of this course and attempt to improve further.

Binary Class	MBTI Personality Type	Accuracy after Configuration	Accuracy before Configuration	Difference
IE 0.84	Introversion (I)–Extroversion (E)	79.01%	78.17%	
NS 0.1	Intuition (I)–Sensing (S)	85.96%	86.06%	-
FT 2.41	Feeling (F)–Thinking (T)	74.19%	71.78%	
JP 0.28	Judging (J)–Perceiving (P)	65.42%	65.70%	-

FIGURE 3. *Comparison of accuracy prediction before and after configuration (Amirhosseini and Kazemian 2020)*

Data Prep

I used the same dataset as the previous two works mentioned - the (MBTI) Myers-Briggs Personality Type Dataset from Kaggle ((MBTI) Myers-Briggs Personality Type Dataset). It's a supervised set of 8,675 samples, each containing the MBTI type of the person (our target), with 50 of their social media posts as the only feature.
Before vectorizing the vocabulary, I took the below steps to ensure I was working with useful data:

1. All words to lowercase
2. Removed all symbols
3. Stemmed words
4. Tokenized the strings

Words were vectorized, and only the highest quantity were kept , measured in frequency among the entire dataset. I saved two version of my data to files for later use, one with max feature (number of vectorized words) at 500 and another set to 1000.

Experimentation / Results

I played around with a few different models and configurations. Initially I set up the models as multi-class classifiers with 16 targets representing our 16 personality types. Targets were set to integer values and one hot encoded. I ran many tests with different configurations, varying the below:

1. Epochs - 1000 vs 3000
2. Feature space size - 500 vs 1000
3. Optimizer - gradient descent with momentum vs adam

Model 1

To start, I set up a single layer network using the keras sequential model function. Data was passed through a sigmoid activation function, and cost was computed as categorical cross entropy.

Results for the different configurations tested are as follows:

	$features = 500$	$features = 1000$
e=1000 / opt=sgd	63%	63%
e=3000 / opt=sgd	59%	57%
e=1000 / opt=adam	55%	48%
e=3000 / opt=adam	53%	47%

TABLE 1. *Accuracy Model 1*
100 -v- 300 epochs
gradient descent w/ momentum (sgd) -v- adam optimizer

I was please to see my results were already on par with the previous works discussed, though not impressive. An observation at first glance is that we are suffering from the curse of dimensionality as we increase our feature space to 1000 terms. Our performance degrades in all but one of our tests. Additionally, it appears that for our problem, the gradient descent with momentum (sgd) optimizer outperforms adam.

Model 2

After testing model 1, I read that the sequential models are not ideal for multiple outputs and created another using the keras general model function. This interestingly performed slightly worse... moving on!

	$features = 500$	$features = 1000$
e=1000 / opt=sgd	57%	60%
e=3000 / opt=sgd	57%	54%

TABLE 2. *Accuracy Model 2*
100 -v- 300 epochs
gradient descent w/ momentum optimizer

Model 3

My next model performed the classification with 4 separate binary classifiers, one for each of the 4 categories:

1. IE = Introversion vs Extraversion
2. NS = Intuition vs Sensing
3. FT = Feeling vs Thinking
4. PJ = Perceiving vs Judging

Accuracy measures after breaking out the problem into separate classifiers were impressive. I tested the model with the same feature and epoch options as above.

	IE	NS	FT	PJ
500 features	85%	89%	84%	78%
1000 features	84%	89%	84%	79%

TABLE 3. *Accuracy Model 3 - Binary Classification of Ea Category*
1000 epochs
gradient descent w/ momentum optimizer

	IE	NS	FT	PJ
500 features	83%	88%	81%	75%
1000 features	80%	88%	82%	77%

TABLE 4. *Accuracy Model 3 - Binary Classification of Ea Category*
3000 epochs
gradient descent w/ momentum optimizer

Accuracy was almost identical for the two feature spaces. Our performance degraded slightly as we trained for more epochs though. This would indicate we reached our local minima around epochs=1000, and that perhaps past that we were overfitting our training data. It's worth noting that I chose to illustrate my models with 1000 epochs because visually that seemed to be the sweet spot for leveling out. See below visualization of our current model's cost vs epoch for both 1000 and 3000 epochs. Our J keeps improving, but our testing performs worse. As stated, this indicates overfitting and to back off on training iterations.

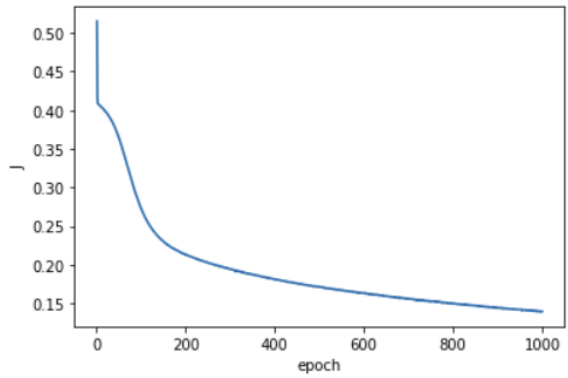

FIGURE 4. *Model 3 Cost (J) -v- Epoch @ 1000 epochs*

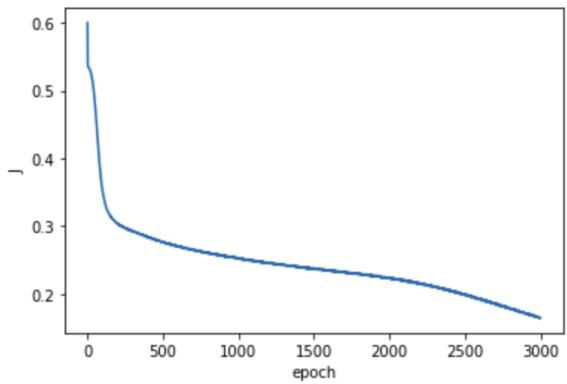

FIGURE 5. *Model 3 Cost (J) -v- Epoch @ 3000 epochs*

Model 4

While I was happy with my results from model 3, I thought it appropriate for our course to at least take a look at what we could do with a deeper model. My final model adds 2 layers to the previous. We have 2 hidden layers applying reLU activation functions, followed by the layer with the sigmoid activation followed finally by the cross entropy cost function. Again, I used the sgd optimizer and tested for the same epochs. You can see my deep model's performance below.

	IE	NS	FT	PJ
500 features	81%	89%	84%	78%
1000 features	81%	89%	84%	78%

TABLE 5. *Accuracy Model 4 - Deep Binary Classifier - Each Category*
100 epochs
gradient descent w/ momentum optimizer

Model 4 performed well, but no better than model 3 at 1000 epochs. I remember class discussion around the fact that you do reach a point of diminishing returns as you add more layers, but I was surprised that we didn't see any bump in accuracy.

Conclusion

can confidently state, that for the parameters given it is possible to classify personality via text! This is exciting for the reasons mentioned prior such as targeted advertising and other similar applications. Four separate binary classifiers outperformed my initial multi-class classifier. The shallow network at 1000 epochs of training, using a gradient descent with momentum optimizer outperformed the others in this category. Reasons for poor performance in other models were too large of a feature space early on, and overfitting with epochs at 3000.

Future Work

With more time, it would be interesting to see where else we can generate text. An issue I foresee is privacy rights for social media posts. If this is going to be a useful tool, we would need to find a way to source public data that is also useful. This would be an unsupervised dataset which would require a new model design. In addition, an observation I made while testing is that category 4 consistently performed worse than the others in our binary classifiers. Since it spanned across models, I would expect this has more to do with aspects of that category of personality traits. Is there a better way to classify this? Maybe the PJ category needs a different type or source of the data.

References

(MBTI) Myers-Briggs Personality Type Dataset. Available at <https://www.kaggle.com/datasnaek/mbti-type>.

Amirhosseini, Mohammad Hossein, and Hassan Kazemian. 2020. Machine Learning Approach to Personality Type Prediction Based on the Myers–Briggs Type Indicator®. *Multimodal Technologies and Interaction* 4 (1). ISSN: 2414-4088. https://doi.org/10.3390/mti4010009. Available at <https://www.mdpi.com/2414-4088/4/1/9>.

Hernandex, Rayne, and Ian Scott Knight. 2017. Predicting Myers-Briggs Type Indicator with Text Classification. *31st Conference on Neural Information Processing Systems (NIPS 2017),* Available at <https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6839354.pdf>.