Classifying user Myers-Briggs type indicator based on text in forum posts

Elissa Ye

May 11th, 2018

05434 – Applied Machine Learning

Carnegie Mellon University

1. Introduction

The Myers-Briggs Type Indicator (MBTI) is one of the most popular and widely used personality inventories of today. Based on the conceptual theory proposed by Carl Jung, the MBTI assumes that people have specific preferences in how they construe their experiences, depending on their interests, needs, values, and motivation. The MBTI assessment consists of 93 multiple choice questions used to categorize people into 16 personality types. When developing the MBTI, Isabel Briggs Myers and Katherine Briggs identified preferences for four dichotomies of mental functions specified in Jung's theory [1]:

- Direction of energy and attention focus: Introversion (I) – Extroversion (E)
- How we absorb information: Intuition (N) – Sensing (S)
- How we make decisions: Thinking (T) – Feeling (F)
- The approach of our outward actions: Judging (J) – Perceiving (P)

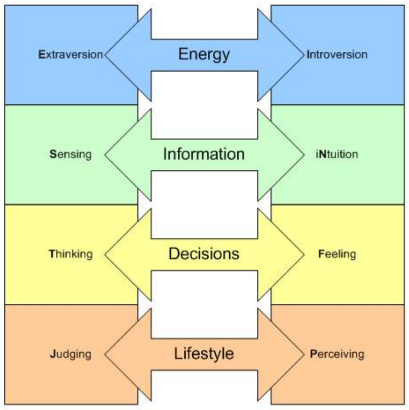

Figure 1: An illustration showing the 4 axes used for MBTI assessment.

The MBTI is widely employed for career counseling, business sectors, and academic advising. Guidance counselors would administer career workshops to assess students' MBTI and help them explore their interests. Organizational leaders could be coached based on the unique leadership styles indicated by their MBTI. Blume (1992) suggested that college students could improve their study habits by knowing their MBTI type and that different learning styles were associated with each preference [2]. An intuitive type is said to be better suited to the study of history than biology, he claimed.

In this paper, I will explore the possibility of determining a user's MBTI based on text communication. Specifically, the aim of this paper is to investigate whether text-based MBTI assessment could be performed with high accuracy using a machine learning algorithms. A dataset containing users' MBTI types and text entries in forum posts will be used to build and optimize a classifier. An effective MBTI classifier that can perform MBTI assessment based on text communication in lieu of a lengthy questionnaire could help autotomize and accelerate the assessment process and greatly increase the amount of psychometric data that could be collected online and thus be beneficially applied to both individuals and organizations.

2. Related Works

Previous works on classifying MBTI demonstrated some success in text-based classification. Brinks & White (2012) used various algorithms to detect MBTI based on brief samples of text communication from Twitter [3]. They attempted to perform binary classifications for each of the four MBTI dichotomies. They initially used a Naïve Bayes classifier, but that performed worse than prior distribution as the classifier made decisions based mostly on number of words. For example, the average number of tokens for an extrovert averaged around forty-three, while for introverts it was over three hundred. They then developed a multinomial Naïve Bayes event model classifier that avoided this selection dichotomy based on the number of tokens and improved their training accuracy to 96% for E vs. I, 83% for N vs. S, 84.6% for T vs. F, and 75.9% for P vs. J. Although L2-regularized logistic regression performed the best for training accuracy, test results showed that the multinomial Naïve Bayes event model performed the best, though with much lower accuracy (~65% average) , but still on par with trained human analysts.

A lot of similar text-based personality profiling studies have been done based on the Big Five Personality Inventory, which models the five domains of personality, openness, conscientiousness, extroversion, agreeableness, and neuroticism, and could provide insight on successful algorithms and features space for appropriate for this study. Golbeck et al. (2011) tried to predict user personality on Twitter based on the Big Five Personality Inventory. They used a lexical approach to extract features and performed a regression analysis using Gaussian process [4]. Their results showed that they could predict personality within just over 10% of resolution, which was relatively fine-grained. Similarly, Wright & Chin (2004) used a support vector machine (SVM) to predict the Big Five personality of writers of free texts [5]. They extracted features such as bag of words, essay length, word sentiment, negation count and part-of-speech n-grams, and found a significant positive contribution of POS n-grams to personality prediction. Mairesse et al. (2007) found that SVM performed the best, with Naïve Bayes and Adaboost in second position, and that extraversion was the easiest trait to predict. Importantly, they found that feature selection significantly impacted performance, as some of the best models contained only a small subset of the feature set [6]. Markoviki et al. (2013) conducted personality mining on Facebook and also found SVM and Simple Minimal Optimization (SMO) to perform the best, and that better sampling features based on the Pearson's correlation coefficient was able to dramatically improve classifiers [7]. Argamon et al. (2009) also tried to characterize the author of a text by analyzing the text itself and distinguished content-based features and style-based features for authorship profiling. They implemented a Bayesian Multinomial Regression, which is a probabilistically well-founded multivariate variant of logistic regression with resistance to over fitting, and found that the most effective features were stylistic features such as function words and individual parts-of-speech [8]. In another study, Argamon et al. (2005) also confirmed the usefulness of functional lexical features for psychological profiling and the need for refinement of feature sets [9].

My study will use a bag-of-words approach to optimize multiclass MBTI classification in text communications and include both content-based and style-based features. Specifically, I will divide my dataset into development, cross-validation, and test sets (Section 3), explore the development set (Section 4), build a baseline model with my cross-validation set (Section 5), iteratively refine my feature space with error analysis (Section 6), tune my model parameters (Section 7), and test my final model on the test set (Section 8) .

3. Data Collection

The Myers-Briggs Personality Type dataset [10], shared on Kaggle, contains 8675 users' MBTI (4 Letter MBTI code) and a section of each of the last 50 things they have posted. The data was collected through the *PersonalityCafe* forum, as it provides a large selection of people and their MBTI personality type, as well as what they have posted on the forum. The MBTI type of users was obtained when the users entered their MBTI type upon joining the *PersonalityCaf*e community.

The dataset was provided as a csv file with 8675 rows and 2 columns, one for MBTI "Type", which is the designated class value, and one for "Post". The post column contained sections from 50 different posts, with each entry separated by '|||'). I partitioned the posts by the deliminter '|||' in excel to create columns for each of the 50 posts. I then named each post from post 1 to post 50 and deleted any posts that extended beyond post 50. I also deleted ' and ' that appeared at the beginning of entries in first column and at the end of entries in the last column. I deleted these artifacts using excel functions. Next, I randomized the instances by using =rand() in excel to create a column of random numbers and sorted the instances by that column. I partitioned the dataset into a development set, a cross-validation set, and a test set by taking 20%, 70%, and 10% of the data respectively. There were 1735 instances for the development set, 6073 instances for the cross-validation set, and 867 instances for the test set. However, for the purposes of this project, I took a random sample of 2075 instances from the cross-validation set to generate a smaller cross-validation set in order to accelerate the model building process.

The classifier I will build and optimize will use the MBTI "Type" as the nominal class (16 class values) and the Posts 1-50 as the text features for feature extraction.

4. Data Exploration

In order to get some ideas for setting up my baseline model, I performed some preliminary data exploration by applying different algorithms with different basic feature space settings to the development set. These algorithms included ZeroR, in which each instance is classified as the majority class (a good baseline for determining performance above chance), Naïve Bayes, Naïve Bayes Multinomial, Logistic Regression (L1), Logistic Regression (L2), and SVM, and the basic feature settings varied based on whether count occurrence, skip non-stop-words, bigram, stem N-grams were checked in LightSide. The 10-fold CV performance accuracy results implemented in LightSide are shown in Table 1. Note that punctuation was included in all cases, feature rare threshold was set to 5, and all algorithms were set to default settings.

ZeroR had an accuracy of 21.27% across all feature space settings. All other algorithms performed better than ZeroR in all cases, which meant that all other algorithms likely contributed to MBTI classification. Logistic Regression (L1& L2) performed relatively better than SVM and

Naïve Bayes Multinomial in all feature space settings. Naïve Bayes' performance was relatively weaker and performed close to chance, likely because Naïve Bayes is relies strongly on prior probabilities. The highest accuracy (55.68%) was obtained with a combination of Logistic Regression (L1) and a "Unigram + Bigram + Count Occurrence" setting. However, that setting yielded ~70000 features, which may more likely be overfitting compared with other feature space settings. Ideally, I would select Logistic Regression (L2) as the algorithm for my baseline model, but that is too computationally expensive given the number of features and the time scope of this project. The parameter tuning would take too long in Weka (which I confirmed in practice). Instead, I selected Naïve Bayes Multinomial model with accuracy performance just second to the Logistic Regression models. Because performance accuracy does not seem to differ much across different basic feature space settings, I will select the basic feature space with the least number of features (~9000 features), "Unigram + Stem N-grams + Count Occurrence," for my baseline model. Although it may seem that skipping non-stopwords improves performance, I decided to keep the stopwords in my feature space with respect to previous work that emphasized the importance of style-based features [8]. The chosen combination of a Naïve Bayes Multinomial model and feature space setting that extracted stem unigrams and counted feature occurrences had a performance accuracy of 44.15%, which is decently above chance and appropriate for my baseline model.

	Unigram	Unigram + Count Occurrence	Unigram + skip non-stopwords + Count Occurrence	Unigram + Bigram + Count Occurrence	Unigram + Stem N-grams + Count Occurrence
ZeroR	21.27%	21.27%	21.27%	21.27%	21.27%
Naïve Bayes	37.64%	22.88%	26.33%	27.38%	23.8%
Naïve Bayes Multinomial	NA	42.07%	46.53%	41.04%	44.15%
Logistic Regression (L1)	48.88%	52.97%	52.56%	55.68%	52.22%
Logistic Regression (L2)	45.88%	52.22%	54.53%	53.54%	53.14%
SVM	44.38%	40.12%	41.42%	40.58%	38.62%
# of Features	12777	12777	12702	69476	9077

Table 1: Performance accuracy for several algorithms and feature space settings using 10-fold cross-validation on the development set in LightSide. The number of features for each feature space is also shown. Naïve Bayes Multinomial could not be implemented without checking count occurrences and is thus not shown.

When exploring the development dataset qualitatively, I noticed a lot of usage of emoticons and sharing of youtube links and a dominant proportion of people with introverted and intuition types. This information may be useful to consider when performing error analysis on the feature space and optimizing the model later on.

5. Baseline Performance

Applying a 10-fold cross-validation on the cross-validation set, the Naïve Bayes Multinomial model with a basic feature space setting (with unigram, stem N-grams, and count occurrences checked in LightSide) had an accuracy of 44.99% and a kappa statistic of 0.3458. The feature space contained 9935 features. The MBTI type distribution for the results of the cross-validation is illustrated in Figure 2.

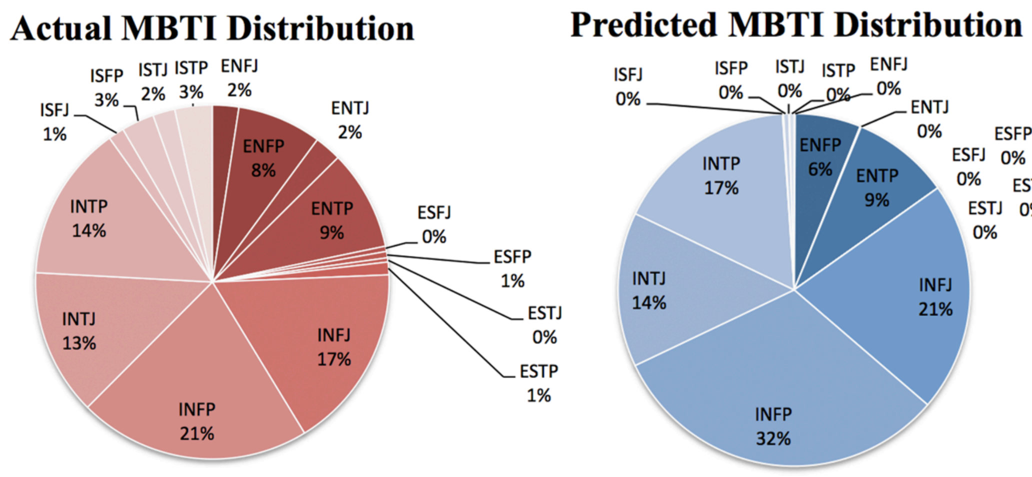

Figure 2: Pie charts showing the actual MBTI type distribution in the cross-validation set and the MBTI distribution based on predictions by the baseline model.

One notable issue with the baseline model is that it never predicted an instance to be a minority type, which includes ISFJ, ISFP, ISTJ, ISTP, ENFJ, ENTJ, ESFP, ESFJ, ESTJ, and ESTP in this case. The Naïve Bayes Multinomial may still depend too strongly on the prior distribution. Consequently, I considered implementing a *CostSensitiveClassifier* that punishes for misclassification of minority classes later in the optimization stage.

6. Error Analysis

The procedure of error analysis involved first building a model from the cross-validation set, testing it on the development set, examining error cells in the resulting confusion matrix with a high number of instances of misclassification, and looking for problematic features with either a high horizontal absolute difference or a low vertical absolute difference. I prioritized examining problematic features with high feature weights and searched for specific instances that may hint possible causes for high errors.

Using the development set as the supplied test set, the model obtained an accuracy of 46.46% and a kappa statistic of 0.3628, which is close to the baseline model. One problem I noticed in error cell INFP/INFJ were features involving apostrophes, such as n't, 've, 're, and 's. Their horizontal/vertical absolute differences and feature influence are shown in Table 2.

Feature	Horizontal Absolute difference	Feature Influence
n’t	1.9375	-0.0339
’ve	0.9994	0.2311
’re	0.2246	0.6535
’s	1.4107	0.034

Table 2: Horizontal absolute differences and features influences for problematic features found in error cell INFP/INFJ.

Problematic Features	**Proposed Features**	**Problematic Features**	**Proposed Features**
n’t	hasn.t|hasnt|has not	’d	I’d| I would
	didn.t|didnt|did not		You’d| you would
	isn.t|isnt|is not		they’d| they would
	haven.t|havent|have not		she’d| she would
	aren.t|arent|are not		he’d| he would
	wouldn.t|wouldnt|would not		we’d| we would
	won.t|wont|will not		lol’d
	can.t|cant|can not	person	personality
	don.t|dont|do not		Personality Café
	couldn.t|couldnt|could not		Personal|personally
	doesn.t|doesnt|does not	quit	quite
’ve	You’ve|you have		quit
	I’ve| I have	si	Si
’re	they’re|they are		sis
	you’re|you are	Busi	business|businesses
’s	that’s|that is		busy
	there’s|there is	ab	abs
	he’s|he is		AB
	she’s|she is	jrr	JRR Tolkien| Tolkien
	it’s| it is	not	cannot
doe	does		not
	doesn’t|does not	do	do
us	us		don’t
	use|used	’m	I’m|I am
	using	string	string theory

Table 3: Problematic features identified based on high absolute horizontal difference or low vertical difference and feature weight and proposed features added to the features space as regular expressions.

These features involving apostrophes often do not distinguish between the whole expressions in the actual text, such that an instance of ‘hasn’t’ and an instance of ‘wouldn’t’ would end up belonging to the same ‘n’t’ feature. To amend this, I proposed creating individual features for expressions related to these problematic features involving apostrophes, as listed in Table 3. Including these proposed features as regular expressions in the new feature space (9956 features), the model yielded an accuracy of 45.03 and a kappa statistic of 0.3464, which shows only insignificant improvement (p = 0.808, t = -1.091) compared with the baseline model. Nonetheless, I decided to include these features in the new feature space, considering the difficulty of achieving significant improvement given a feature space with already containing thousands of features.

After several iterations of error analysis, as listed in Table 3, I did not get any significant improvement/decrease in performance by incorporating these regular expressions. I decided to include all the new features in the feature space anyway, since I would be performing feature selection later on. The model after error analysis (9982 attributes) had an overall accuracy of 0.4484 and kappa statistic of 0.343, with an insignificant decrease (p = 0.631, t=-0.242) in performance compared with the baseline model.

7. Optimization

The Naïve Bayes multinomial algorithm itself does not have parameters to tune, so instead, I will be tuning the number of features to be selected for building the model using LightSide's *AttributeSelectedClassifier.* Within the *AttributeSelectedClassifier*, I chose Naïve Bayes Multinomial as the classifier, *CorrelationAttributeEval* as the evaluator, and *Ranker* as the search algorithm. The parameter I specifically tuned was *numToSelect* in *Ranker,* which specifies the number of attributes to retain. To tune this parameter, I followed the 3-stage procedure for parameter tuning. Results for parameter tuning are shown in Table 4. For stage 1, I decided to perform 10-fold cross-validations on the cross-validation set for numToSelect = -1, 20, 50, 100, 500, 1000, and 5000. In this case, setting the number of features to retain to 100 yielded the best accuracy (57.91%), and thus, I selected 100 as my best setting for stage 1. I proceeded to stage 3 in attempt to estimate my model performance for this setting. After saving the features extracted (from the cross-validation set) in LightSide as an arff file, I used the *StratifiedRemoveFolds* filter in Weka to create 4 train sets and 4 test sets. To calculate performances in the inner loop, I performed 10-fold cross-validation on the 4 training datasets. I then selected the optimal setting for each fold and obtained test performances by using the optimal setting for evaluating the train/test pairs of each fold. The estimated performance of the model with numToSelect = 100 would thus be 56.90%, which is the average of test performances for each fold.

	Performance given numToSelect							Optimal Setting	Test Performance
	-1	20	50	100	500	1000	5000		
Fold 1	43.54	53.76	58.97	58.14	54.92	51.77	45.85	50	56.45
Fold 2	44.82	54.34	55.56	56.78	53.83	50.48	47.07	50	56.07
Fold 3	41.84	50.97	54.82	55.78	54.43	51.29	45.37	100	59.85
Fold 4	42.61	52.25	56.62	57.39	53.41	51.29	44.99	100	55.21
Full	43.88	54.68	57.14	57.91	55.40	52.12	46.19	100	Avg = 56.90

Table 4: Results for parameter tuning of numToSelect in the search algorithm ranker. Note that numToSelect = -1 is the default setting.

A 2-tailed paired t-test was used to compare the test performances of each fold and the accuracies of each fold for the default setting (numToSelect = -1). The resulting p-value was 0.0027 (<0.05), which suggests significant improvement. The performance accuracy estimated for the final test set with the current model was 56.90%

Additionally, I tried implementing a *CostSensitiveClassifier* with a 16 x 16 cost matrix that penalized 0.001 every time a majority class value predicts an actual minority class value. The resulting 10-fold cross-validation on the CV set had an accuracy of 24.40% and kappa statistic of 0.1654, which dramatically decreased the performance of the initial model. Thus, I opted to not incorporate this classifier.

8. Final Result

To generate the final result, a model with the new feature space and optimized number of features was trained on the cross-validation set and then tested on the test set. This resulted in a performance accuracy of 58.59% and a kappa statistic of 0.5265. The percentages of correctly classified instances for each class are shown in Figure 3.

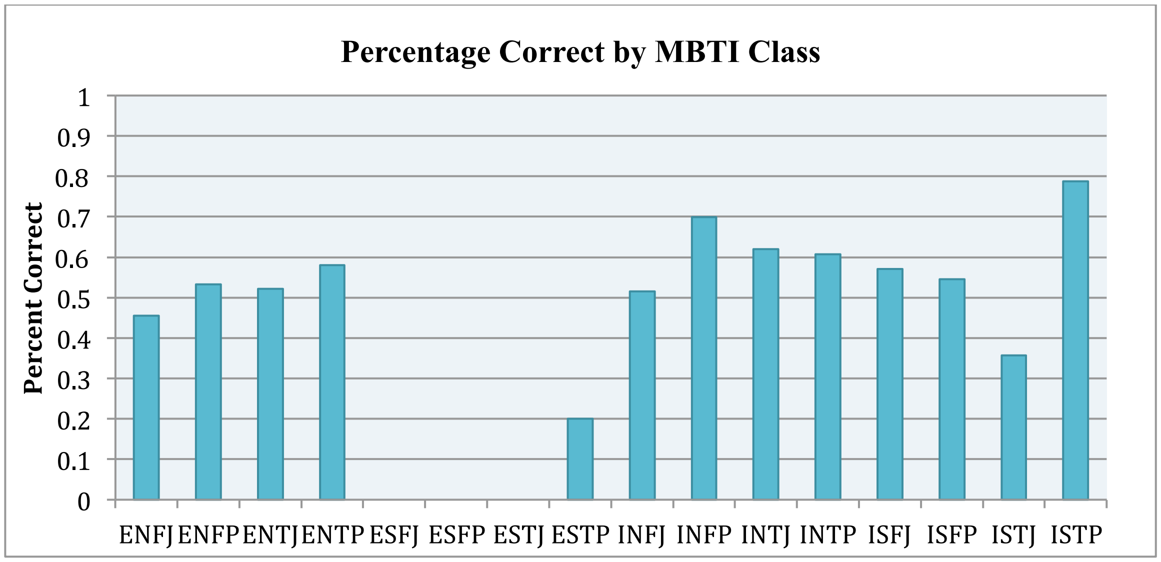

Figure 3: Bar graph showing percentage of correctly classified instances for each class value.

To gather more insight on what features selection may have been selected, the cross-validation set and test set were combined into one dataset and imported into LightSide for feature extraction. Once the features were extracted, the combined dataset was exported as an arff file and then imported in Weka. In Weka, the *RemoveRange* filter was used to partition the instances of the dataset based on the original cross-validation set and test set. The resulting arff files are then used for training and testing in Weka with the optimized model settings. Figure 4 shows prior probabilities computed by the final model. Figure 5 shows the 100 important features selected in the test set based on rank.

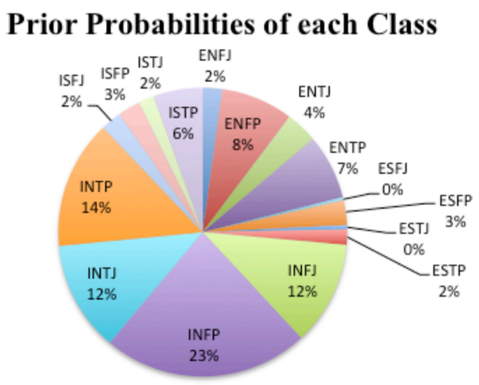

Figure 4: Pie chart representing prior probabilities of each class calculated by the final model.

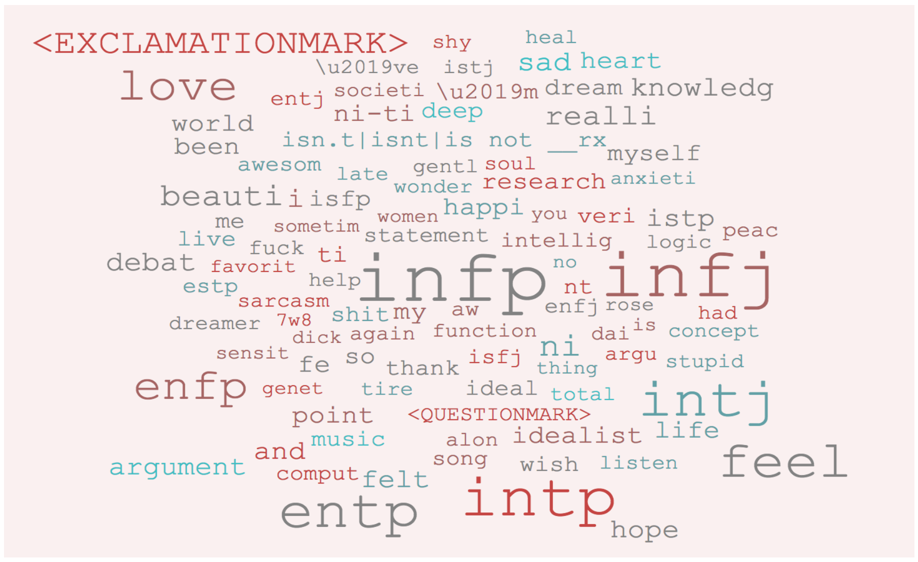

Figure 5: A word cloud showing the top 100 features selected in the final result. Word size is proportional to word rank based on the Pearson's correlation with class value.

9. Discussion

The final Naïve Bayes Multinomial model (NBM) selected 9982 features based on their Pearson's correlation with class value and resulted in a performance accuracy of 58.59% and kappa statistic of 0.5265. This was very close to the predicted 56.90% and demonstrated the validity and generalizability of the final model. Overall, the results support the hypothesis that text-based MBTI classification could be achieved with relatively high accuracy.

The results also support the assumption that NBM may be especially appropriate for text classification and modeling word counts. NBM is a probabilistic learning model similar to the simple Naïve Bayes, which considers prior probabilities and independent conditional probabilities, but uses a Multinomial model assumption for the conditional probabilities instead of a Gaussian assumption. The NBM estimates the conditional probability of a particular word given a class using the relative word frequency. It is not surprising that a NBM performed decently well, since the multinomial models are better at capturing the distribution of word frequency, which is a discrete data, as oppose to Gaussian models. The NBM is also a relatively simpler model compared to logistic regression and SVM, making it more adaptable to new data. Like Naïve Bayes, the NBM may be skewed towards the class distributions due to contribution of prior probabilities. In Figure 3, only 3 classes did not have any value for percent correct, but this could be just due to the class values rarely or ever occurring. Since most class values have some percent accuracy, the effect of any underlying skewed distributiong may not have been very dominant. As seen in Figure 4, the prior probabilities for this new model may have been relatively reflective of the actual distribution, since there doesn't seem to be much bias against minority classes. Nonetheless, the Naïve Bayes assumption, stating that the probability of each word event is independent of context and position, may not necessarily hold true in this case and could've negatively impacted the results.

As demonstrated by previous works [6] [7] [9], much of the model improvement occurred due to the reduction of feature space. Reducing the number of features perhaps eliminated uninformative features that may lead to overfitting and a lower test accuracy. Typically, the classifier needs a number of instances that's logarithmic to the number of features, so having too many features relative to instances may decrease performance. The Pearson's correlation for attribute selection, which is one of the most efficient and recommended approaches, is also shown to be effective for this study.

Looking at Figure 4, I noticed that a lot of the top features were the MBTI types themselves, which makes intuitive sense, as people would more likely mention their own MBTI type when discussing in a forum. There was also a good mix of content-based features and style-based features, which aligns with findings previous studies emphasizing the importance of functional lexical predictors [8] [9].

The biggest limitations of this study concerns the less representation of rare class values in the data, such as ESFJ, ESFP, and ESTJ, and the large number of 16 class values that needed to be predicted, making this classification task exceptionally difficult. Nonetheless, the results of the classifier were encouraging, with still potential for improvement, and definitely suggests a strong future for autotomizing personality profiling in text communications.

10. Future Works

Future work could perhaps involve binary classification of the MBTI based on the four main axes. Each binary classifier could be optimized and then integrated together to build a MBTI classifier with potentially significantly higher performance accuracy. Given that only about 50 to 100 features seemed to suffice for the final classifier to perform well, decision trees and rule-based algorithms could also be used for model building and could perhaps provide for deeper relational insights in what underlying mechanisms may distinguish users from each class. Consistent use of stratified folds could also help avoid the bias of skewed distribution in training models. Bigrams and other POS n-grams may be interesting to examine as well.

References

[1] The Myers & Briggs Foundation. (2018). Retrieved May 10, 2018, from http://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/

[2] Blume, S. (1992). Learning styles. *Your college experience: Strategies for success*, 49-64.

[3] Brinks, D., & White, H. (2012). Detection of Myers-Briggs Type Indicator via Text Based Computer-Mediated Communication.

[4] Golbeck, J., Robles, C., Edmondson, M., & Turner, K. (2011, October). Predicting personality from twitter. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on* (pp. 149-156). IEEE.

[5] Wright, W. R., & Chin, D. N. (2014, July). Personality profiling from text: introducing part-of-speech N-grams. In *International Conference on User Modeling, Adaptation, and Personalization* (pp. 243-253). Springer, Cham.

[6] Mairesse, F., Walker, M. A., Mehl, M. R., & Moore, R. K. (2007). Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of artificial intelligence research*, *30*, 457-500.

[7] Markovikj, D., Gievska, S., Kosinski, M., & Stillwell, D. (2013, June). Mining facebook data for predictive personality modeling. In *Proceedings of the 7th international AAAI conference on Weblogs and Social Media (ICWSM 2013), Boston, MA, USA* (pp. 23-26).

[8] Argamon, S., Koppel, M., Pennebaker, J. W., & Schler, J. (2009). Automatically profiling the author of an anonymous text. *Communications of the ACM*, *52*(2), 119-123.

[9] Argamon, S., Dhawle, S., Koppel, M., & Pennebaker, J. (2005). Lexical predictors of personality type.

[10] J, M. (2017, September 22). (MBTI) Myers-Briggs Personality Type Dataset | Kaggle. Retrieved March 10, 2018, from https://www.kaggle.com/datasnaek/mbti-type