PREDICTING PERSONALITY TYPES FROM USER COMMENTS

An exploration of natural language processing (NLP)

CODE: link

(packages used include Numpy, Pandas,

(accuracies for all experiments can be found in appendix)

**Introduction**

I have always been fascinated by personality types. Skeptical of its scientific validity, I have still found it to be a pretty reasonable description of the characteristics my friends and I possess. This project explores the significance of personality types in determining the use of language, in particular online comments in forums. The machine learning task is to predict personality type of a user based on their comments on an online forum. The project has far-reaching implications if you consider the ever-increasing availability of user data on online forums and social media platforms. Based on a user’s comments and messages, what can a platform learn about its users to better mediate discussion and provide more meaningful content?

**Dataset**

The dataset, titled “(MBTI) Myers-Briggs Personality Type Dataset” was found on [Kaggle](https://www.kaggle.com/datasnaek/mbti-type). It contains 8600 examples (rows) in which each row represents a specific user, containing entries of personality type (4 letters based on the MTBI type) and their 50 most recent comments on an online forum called PersonalityCafe.com. The dataset was compiled by Mitchell J, by scraping data from the website. Here is an example of the dataset:

|  |  |
| --- | --- |
| MBTI Type | Posts |
| INFJ | 'http://www.youtube.com/watch?v=qsXHcwe3krw|||... |
| ENTP | 'I'm finding the lack of me in these posts ver... |
| INTP | 'Good one \_\_\_\_\_ https://www.youtube.com/wat... |
| INTJ | 'Dear INTP, I enjoyed our conversation the o... |
| ENTJ | 'You're fired.|||That's another silly misconce... |

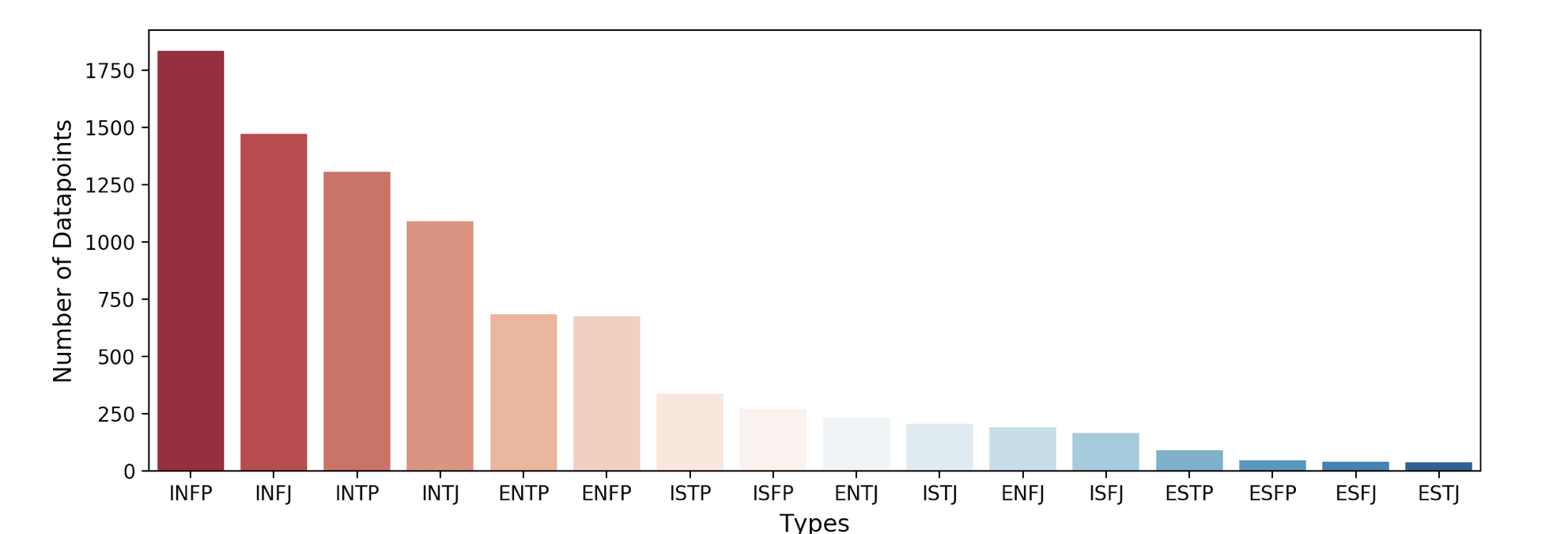
Each user comment in ‘Posts’ column is separated by three vertical bars “|||”. The following is an example row of one user:

|  |  |
| --- | --- |
| INTP | 'Good one \_\_\_\_\_ https://www.youtube.com/watch?v=fHiGbolFFGw|||Of course, to which I say I know; that's my blessing and my curse.|||Does being absolutely positive that you and your best friend could be an amazing couple count? If so, than yes. Or it's more I could be madly in love in case I reconciled my feelings (which at...|||No, I didn't; thank you for a link!|||So-called Ti-Si loop (and it can stem from any current topic/obsession) can be deadly. It's like when you're stuck in your own thoughts, and your mind just wanders in circles… |

There are 4 dichotomies in the MBTI type classification that are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Extraversion (E) – Introversion (I) | Sensing (S) –  Intuition (N) | Thinking (T) –  Feeling (F) | Judging (J) –  Perceiving (P) |

This is how the dataset was distributed with respect to the 16 distinct output classes:



Upon analyzing this distribution I noticed that the dataset was heavily skewed, with certain output classes having more examples than others. I then analyzed the distribution of each dichotomy individually, and noticed that the distribution of the third dichotomy of “Thinking (T) — Feeling (F)” and that of the fourth dichotomy of “Judging (J) — Perceiving (P)” was more evenly split than that of any other dichotomies. Upon testing several machine learning algorithms using output class as third dichotomy and fourth dichotomy respectively, I found that accuracies were greater when considering output class as the third dichotomy. For this reason, I decided to alter my machine learning to predicting the third dichotomy of ‘Thinking (T) — Feeling (F)’ instead of the personality type of the user, changing the classification from multi-output to binary. This sounded like a more reasonable task to accomplish given that my dataset consisted of only 8600 examples, which isn’t very large for a task involving NLP.

**Preprocessing**

I decided to use the bag-of-words approach to extract features from the comments of my users because, as discussed in class, it is a simple and effective technique. The bag-of-words model represents text as a set of words used within it, disregarding grammar and word order but keeping the multiplicity of words used in the document. I decided to use bag-of-words instead of an n-gram model (such as bigrams or trigrams) because the comments in the dataset are random comments covering several different topics. Higher order n-grams would only be effective given a common thread of discussion in which we could imagine certain pairs or triplets of words holding significance.

The first phase of preprocessing was to clean the text, which involved the following steps:

|  |
| --- |
| * Remove all characters (special characters and numbers) except letters of the alphabet |
| * Make all letters lowercase |
| * Lemmatize the document |
| * Remove all stopwords |
| * Replace all urls of websites with the word “link” |
| * Remove all words correspoding to the 16 personality types (such as ENFP, INTP, etc.) |

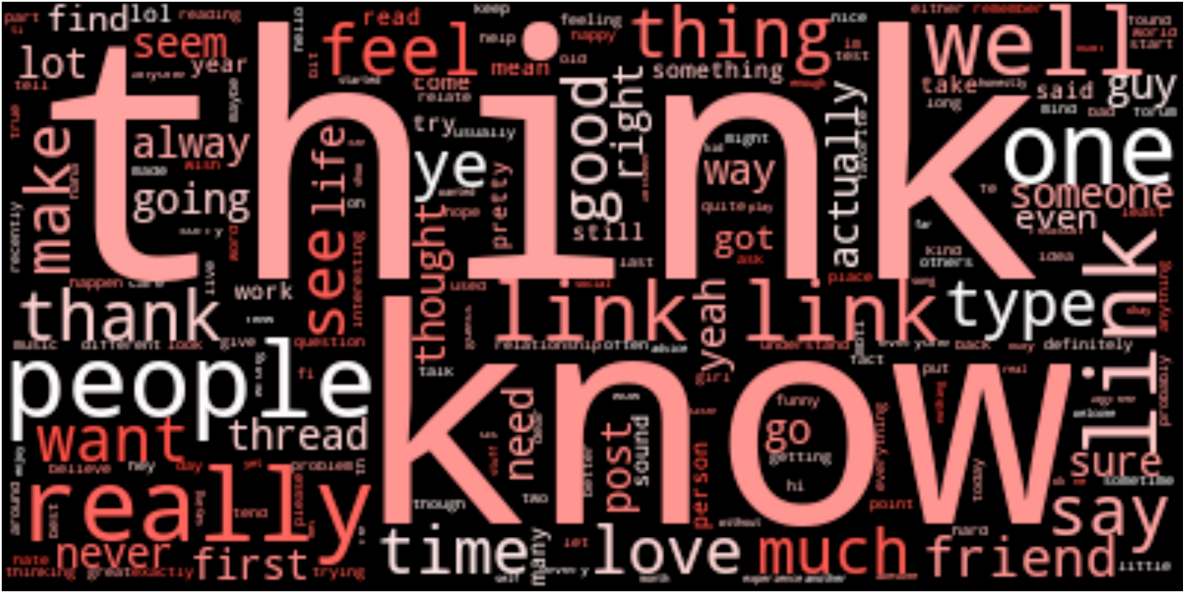
I chose to use lemmatization instead of stemming in my implementation. I used lemmatization instead of stemming because, after testing datasets created using stemming and lemmatizing respectively on learning algorithms, I noticed that lemmatization led to higher accuracy by ~1%. The choice was partially random because a difference of accuracy of 1% isn’t statistically significant. The lemmatization and stemming was carried out using the NLTK package’s WordNetLemmatizer and PorterStemmer respectively. Lemmatization and stemming both involve reducing a word to its common base form in order to reduce the size of the vocabulary in the document and group together words that could share the same meaning, however, they differ slightly in their implementations. For instance, stemming might reduce the words ‘studies’ and ‘studying’ to ‘studi’ and ‘study’ respectively while lemmatizing would reduce both to ‘study’. Lemmatization returns a more meaningful base form of the word, taking into account the context the word is being used in.

Stopwords were removed based on the list of stopwords provided in NLTK corpus. Stopwords refer to the most common words in language that do not help in featurizing text because they are used everywhere, such as ‘at’, ‘the’, and ‘which’. Removing them is a form of feature selection that prevents our learner to be distracted by noise.

Urls of websites were replaced the word “link” because it would become tricky to consider the different websites users mentioned. The word “link” was used so that the training algorithm could still factor in which users used urls and which didn’t. For future exploration, it could be relevant to create a separate feature for which website the user referenced in their comments.

I chose to remove all words corresponding to the 16 personality types (such as ENFP and INTP) because people are likely to talk about their own personality type on an online forum like PersonalityCafe.com. Not removing these words would be similar to giving the output class as a feature to the learner, in which case the resulting accuracy would be an exagerrated estimation of the relationship between the features and output class. Upon trying several learning algorithms with and without removing the personality types, I noticed that the results were as I had expected: the accuracy of Naïve Bayes and Logistic Regression decreased by ~4% and ~5% respectively.

At this stage, I was curious to see which words were most frequently being used by users in the comments. To visualize this, I used the WordCloud package to generate the following image in which the size of each word corresponds to its document frequency.



**Vectorization**

After testing the features with both a count vectorizer and tf-idf vectorizer provided by SciKit’s library, I decided to use the tf-idf vectorizer. The tf-idf vectorizer produced an almost identical accuracy to the count vectorizer for logistic regression, and produced a ~1.5% better accuracy for Naïve Bayes. Because there was not enough statistical significance to choose one vectorizer over another, I chose tf-idf because it is considered a more sophisticated and meaningful approach. Both vectorizers create features from the text by compiling the unique words in the vocabulary of the entire dataset (i.e. all examples) and analyzing the frequency of each unique word for every example. The count vectorizer calculates frequency by simply looking at the number of occurences of a word in an example. Tf-idf, which stands for “term-frequency, inverse-document frequency” uses the same approach but assigns a lower weight to terms that appear more frequently in the document, as these words are less likely to be great predictors of the output class compared to words that are more specific to the given example.

During vectorization, I changed the parameters of ‘maximum features’ and ‘maximum document frequency’ to find the optimum parameters of 1000 and 1 respectively. Changing ‘maximum features’ from 500 to 1000 increased accuracy by ~1% for logistic regression, while increasing it any more did not significantly increase the accuracy. When ‘maximum document frequency’ was reduced from 1 to 0.7 and then 0.5, the accuracy reduced by ~0.5% each time, suggesting that 1 was the optimum value for the parameter.

**Machine Learning algorithms**

I trained my vectorized features using ZeroR, logistic regression, naïve bayes, decision trees, and multilayer perceptrons using default training algorithms provided in the Weka software. Multilayer perceptrons did not finish training because of the large dimensionality of the feature space and the time-intensive perceptron update rule. The 10-fold cross validation accuracies of all other algorithms are noted in the appendix of this report. Logistic regression and Naïve Bayes performed the best, with accuracies far better than the accuracy of ZeroR:

|  |  |
| --- | --- |
| Training algorithm | 10-fold cross validation accuracies |
| ZeroR | 54.11% |
| Logistic Regression | 75.53% |
| Naïve Bayes | 70.70% |

It seems like logistic regression is the best algorithm for the task given a difference of ~5% between its accuracy and that of Naïve Bayes. I couldn’t formally compare statistical significance using contigency tables and methods such as Chi-Square and Fisher’s test because of the large number of features. Statistical significance will be an aspect of future research for this project.

Didn’t use precision and recall in this analysis because false negatives/false positives are not really that catastrophic. And there is even distribution.

I then performed feature selection on the dataset to find the words were the best predictors of the output class. This was the most insightful part of my project as it helped me see the correlation between words used in user comments and personality types. I used the following three approaches of feature selection: correlation based feature selection, information gain based feature selection, and learner based feature selection. I selected the features with the highest ranks in these feature selection and then looked at the number of occurences of each of these words in the dataset for the dichotomies of ‘Thinking (T)’ and ‘Feeling (F)’. I found it extremely insightful that the words we would generally associate with “thinking” and “feeling” were the ones that were the top-ranked features of each dichotomy, shown in the figure below. Once again, statistical significance of these values will be an aspect of future research for this project.

Words associated with “feeling” Words associated with “thinking”

|  |  |  |
| --- | --- | --- |
| Significant words (features) | Frequency of word for output class T | Frequency of word for output class F |
| feel | 13140 | 25431 |
| love | 8242 | 17122 |
| feeling | 4133 | 7543 |
| happy | 2301 | 4702 |
| beautiful | 532 | 1613 |
| really | 13927 | 21397 |
| heart | 1064 | 2490 |
| thank | 5468 | 10116 |
| felt | 1151 | 2304 |
| hope | 1794 | 3381 |
| sad | 1300 | 2554 |
| haha | 3095 | 5528 |
| deep | 1212 | 2357 |
| song | 1263 | 2910 |

|  |  |  |
| --- | --- | --- |
| Significant words (features) | Frequency of word for output class T | Frequency of word for output class F |
| information | 1239 | 958 |
| science | 1271 | 889 |
| logic | 2788 | 1871 |
| knowledge | 1114 | 714 |
| argument | 1236 | 731 |

**Conclusion**

The words that the feature selection ranked as significant are greatly insightful to the nature of what it means to be of a “Thinking (T)” or “Feeling (F)” type. It is amazing to see that the following words, which are less explicitly related to the “feeling” dichotomy but still one of its characteristics, are predictors of the dichotomy: love, beautiful, really, heart, thank, hope, haha and song. For the “thinking” dichotomy, the results are less interesting because the difference in the frequencies for each output class are not as significant. It is not surprising that words like “feel” and “feeling” are predictors of the “feeling” type, especially because the comments in the dataset were scraped from an online forum where people talk about their personality types. This is definitely a limitation of our dataset that needs to be addressed in future research by collecting data from other social media and discussion platforms. Furthermore, the dataset collected needs to be larger so that classification can be performed for all dichotomies.

**Other topics I explored**

A major area of learning for me in this project was of topic selection. In trying to choose a topic of interest and feasibility, I transitioned topics many times due to which my project proposal, status report, and final report are all on different topics.

|  |  |  |
| --- | --- | --- |
| Topic | Stage of final project | Learning |
| Predicting whether a sequence of dance keyframes is “good” or “bad” | Project Proposal and initial email to professor Downey. | There is no point of subjectively assigning data to art and then using machine learning on that subjective data. Data, for machine learning, needs to be “true” for any meaningful learning to happen. |
| Predicting “news publication” based on text of articles | Status Report | There needs to be a meaningful objective to any machine learning task— otherwise we are building technology simply for the sake of technology. |
| Performing “text summarization” | Conversation with professor Downey in office hours | “Text summarization” is a large unsolved problem in the world of machine learning. Additionally, any large-scale NLP tasks require extremely fast computing speeds and long training times. |

**Appendix**

|  |  |  |
| --- | --- | --- |
| Algorithm (Weka standard) | 10-fold CV accuracy | Comments |
| ZeroR | 60.415% | Started by looking at Judging-Perceiving dichotomy, stemming instead of lemmatizing, and without having removed words of ‘personality types’ from comments. |
| NB | 72.4957% |  |
| LO | 79.0317% |  |
|  |  |  |
| LO | 83.147% | Started looking at Thinking(T) - FEELING (F) dichotomy |
| DT (J48) | 72.634% |  |
|  |  |  |
| NB | 68.1268% | Removed {enfp, intp} from ‘posts’ |
| LO | 74.4323% |  |
|  |  |  |
| NB | 68.1499% | Started Lemmatizing |
| LO | 74.4438 % |  |
|  |  |  |
| NB | 69.8213 % | Changed to Tfidf Vectorizer |
| LO | 74.5245 % |  |
|  |  |  |
| LO | 73.6023 % | Tried with max\_df = 0.5. Changed back to 1. |
|  |  |  |
| ZeroR | 54.1095 % | Changed max\_features from 500 to 1000 |
| LO | 75.5274 % |  |
| NB | 70.6974 % |  |