WASHINGTON STATE UNIVERSITY

2021

HAM/SPAM Message Classification – NATURAL LANGUAGE PROCESSING

CPTS-570 MACHINE LEARNING – WASHINGTON STATE UNIVERSITY - PULLMAN

PROJECT REPORT

Kulpreet Singh – 011771817 **Sahil Shrivastava - 011717650**

Abstract:

Emails, text messages, tweets, any form of message-delivery technologies have simplified many arduous tasks since various types of communication necessities are fully satisfied. On the contrary, these same technologies could complicate tasks when utilized outside of their purposes. Email, for example, plays a vital role in industrial business applications. They transfer sensitive/vital information from one source to another. If that information were to be stolen, lost, or leaked to some source that would utilize it negatively, then that would cause much harm to that specific individual/business.

Furthermore, the user receives arbitrary and non-sensical messages, which could be annoying and dangerous for the receiver's information security. Thus came the need for information security to detect and prevent these non-sensical/spam messages as an entity. For this project, we utilized the Kaggle dataset of ham-spam messages. Using NLTK libraries and various classifiers from the sci-kit learn library, we obtained different results of classification accuracies of the ham-spam classification. From these calculations, we obtained various visualizations and concluding results.

Introduction:

Unwanted or unsolicited messages or emails delivered to cause harm, or financial fraud, or to annoy a user is defined as "Spam." These messages tend to spread viruses attacking the individuals or businesses' computer framework resulting in either stealing information or causing financial hardships; furthermore, out of 80 billion emails delivered daily, at least 60% consist of spam emails.

Spam filtering/detection involves heavy pattern recognition, data analysis, mining, and extensive research utilizing machine learning algorithms. Once an algorithm is constructed, it first parses a message as input identifying any keywords or "stopwords" as a method to perform data cleaning. With the utilization of regular expressions, more complicated words which consist of alpha-numeric values are filtered out. Once the data is cleaned, it is then analyzed through multiple methods. For this project, we have utilized seven classification algorithms:

1. Multinomial Naïve Bayes (MNB)
2. Stochastic Gradient Descent (SGD)
3. Support Vector Machines (SVM)
4. Linear Regression (LR)
5. Decision Trees (DT)
6. Random Forests (RF)
7. K-Nearest Neighbor (KNN)

For each classifier, we have constructed a confusion matrix, graphical analysis depicting the length of messages on the abscissa and their classification label as ham/spam on the ordinate, along with word clouds showing the frequency of ham's occurrences and spam messages. Thus, an extensive and detailed analysis was done to show which classifier performed the best in the Natural Language Processing of HAM/SPAM Message classification.

Architecture & Procedure:

Programming Language: Python v3.9

Platform: Jupyter Notebook

Libraries utilized: Pandas, NumPy, NLTK, Scikit Learn, matplotlib, Word Cloud, seaborn, re (regular expressions)

1. **Initialization of Data:**
    1. Initially, all the necessary libraries were loaded, and the data was loaded as a CSV data frame.
    2. The columns were renamed from v1, v2 to label and message, respectively.
    3. Distribution of the ham and spam messages was made.
    4. This resulted in creating a bar graph showing the distribution of messages.

2. **Data Cleaning:**
    1. The text's useless features like punctuation, stop words, words like *receiving, receiver* to -> *reciev* were done. This process is called "Stemming."

```
#Removing all the useless features from the texts like punctuation words and stopwords
for i in range(0, len(texts)):
    text = re.sub('[^a-zA-Z]', ' ', texts['message'][i])
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = text.lower()
    text = text.split()
    text = [ps.stem(word) for word in text if not word in stopwords.words('english')]
    text = ' '.join(text)
    corp.append(text)
```

3. **Analysis of Ham/Spam**:
    1. Two word clouds were generated, showing the most common words in either HAM/SPAM Messages.

4. **Machine Learning Implementation**:

    Classifiers Utilized:

    1. **Multinomial Naïve Bayes**: Here, a multinomial distribution is utilized for the CSV data, which calculates word count frequency in a text. The classifier has discrete features which calculate the feature count.

```
# Training using Naive Bayes Classifier

from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB().fit(X_train, y_train)

prednb =mnb.predict(X_test)
```

    2. **Decision Tree**: This classifier utilizes nodes, wherein each node is based on a feature. For each node, a decision is made based on the value of that specific feature. For our case here, each feature is a stop word, and based on the stop word, it counts its frequency. It classifies it as useful/unnecessary information. Finally, each node hails a leaf that corresponds to a label.

```
# Training using Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(random_state=0).fit(X_train, y_train)

preddtc = dtc.predict(X_test)
```

3. **Logistic Regression**: This is a statistical approach where the function utilizes binary dependent variables. It estimates the parameters and learns the conditional distribution P(y|x). This produces a linear decision boundary.

```
# Training using Logistic Regression

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state=0).fit(X_train, y_train)

predlr = lr.predict(X_test)
```

4. **K-Nearest Neighbors**: KNN is a non-parametric classifier that stores all training data. A specific numeric value of "k" is chosen, which calculates the Euclidean distance between each neighbor. Whichever distance value is smaller is clustered into that specific neighborhood.

```
# Training using KNN

from sklearn.neighbors import KNeighborsClassifier
spamKNN = KNeighborsClassifier(n_neighbors = 3).fit(X_train, y_train)

predKNN = spamKNN.predict(X_test)
```

5. **Support Vector Machine**: A hyperplane is constructed, which separates the data into two specific types of labels. Regression is performed, and whichever label lies on either side of the hyperplane, that label gets classified into that specific class. It is beneficial to use if the data is linearly separable; otherwise, it has difficulty generalizing.

```
# Training using Support Vector Machine

from sklearn import svm
spamSVM = svm.SVC(probability = True).fit(X_train,y_train)
predSVM = spamSVM.predict(X_test)
```

6. **Random Forest:** is a classifier that uses bagging and feature randomness when constructing each tree to create an uncorrelated forest of trees. The prediction is more accurate than that of an individual decision tree.

```
# Training using Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier
spamRF = RandomForestClassifier(random_state = 0).fit(X_train, y_train)
predRF = spamRF.predict(X_test)
```

7. **Stochastic Gradient Descent (Linear SVM)**: This classifier utilizes regularized linear models wherein the gradient of the loss function is calculated for each feature and is updated with a learning rate.

```
# Training using Stochastic Gradient Descent

from sklearn.linear_model import SGDClassifier
sgd = SGDClassifier().fit(X_train,y_train)
predsgd = sgd.predict(X_test)
```

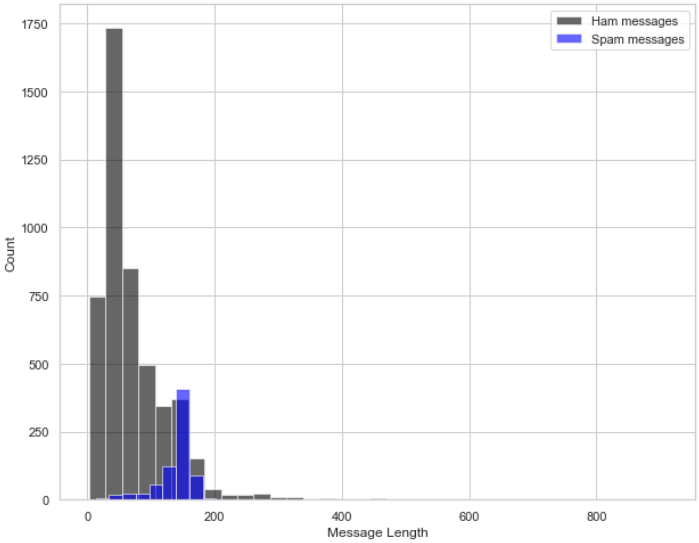
Results & Discussions:

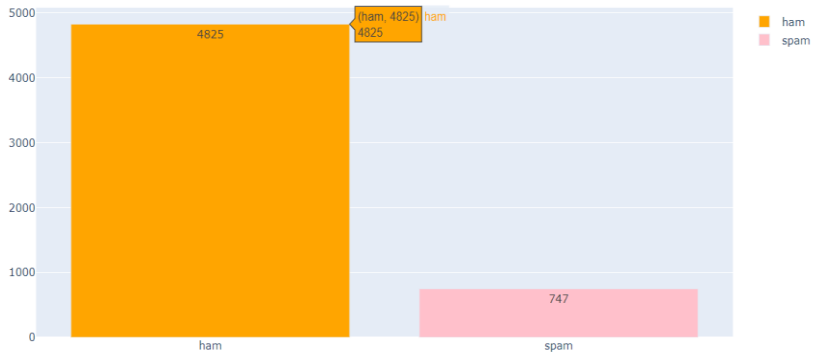
Data Acquisition & Initialization:

Initially, we obtained a bar graph from the dataset that depicted the spread/distribution of the messages. A look at the final Data frame on the right after the message lengths and cleaned messages have been added.

	label	message	message_len	clean_messages
0	ham	Go until jurong point, crazy.. Available only ...	111	go jurong point crazi avail bugi n great world...
1	ham	Ok lar... Joking wif u oni...	29	ok lar joke wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	free entri wkli comp win fa cup final tkt st m...
3	ham	U dun say so early hor... U c already then say...	49	u dun say earli hor u c alreadi say
4	ham	Nah I don't think he goes to usf, he lives aro...	61	nah think goe usf live around though
...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	160	nd time tri contact u u pound prize claim easi...
5568	ham	Will _ b going to esplanade fr home?	37	b go esplanad fr home
5569	ham	Pity, * was in mood for that. So...any other s...	57	piti mood suggest
5570	ham	The guy did some bitching but I acted like i'd...	125	guy bitch act like interest buy someth els nex...
5571	ham	Rofl. Its true to its name	26	rofl true name

5572 rows × 4 columns





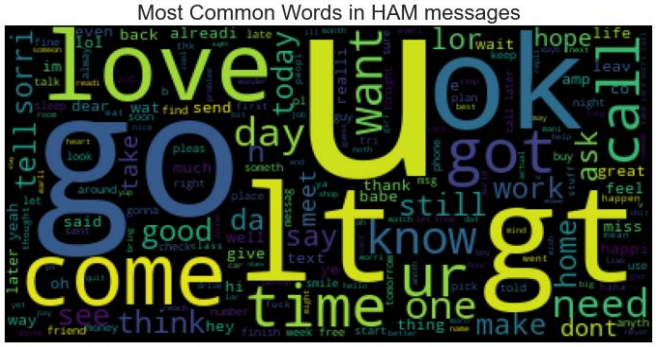
Bar Plot showing number of HAM/SPAM Messages

From the above graph, we can see that, in general, the distribution of ham messages is more significant than compared to spam. However, spam messages are pretty prominent and have weightage in the distribution.

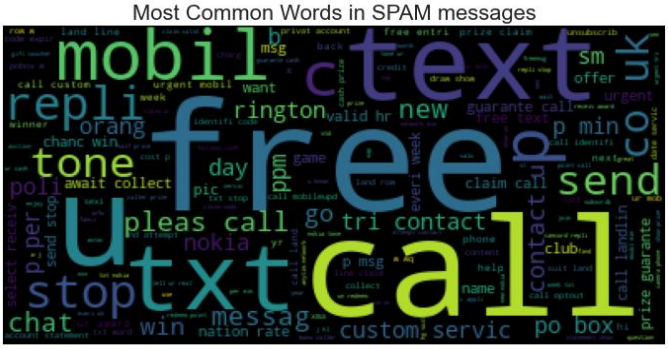
Data Cleaning & WordCloud Formation:

Then after using regular expression data cleaning, two word clouds were generated to show the frequency of words in HAM/SPAM Messages.

Most Common Words in HAM messages



Most Common Words in SPAM messages

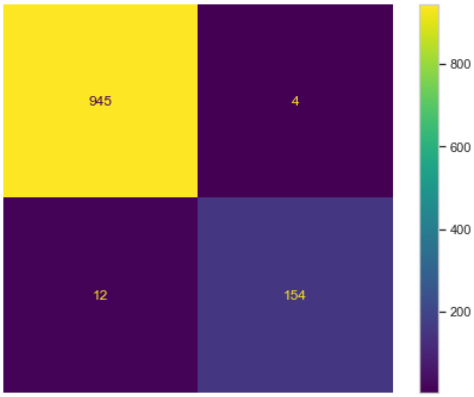


The image on the left is the most common word used in HAM Messages. They are ordinary and non-sensical words classified as helpful information for the receiver. The image on the right is the common words found in SPAM Messages. It can be observed that unnecessary words such as co, po, p.per, p, and msg are unnecessary information that is not useful to the receiver.

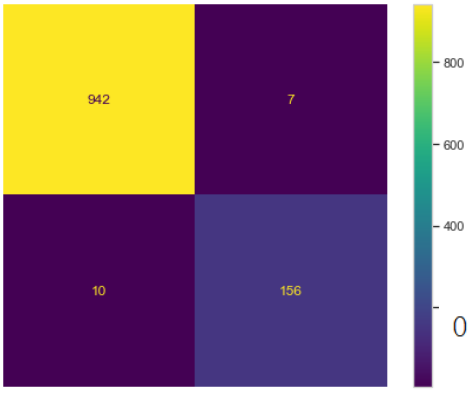
Classification Results:

The two best classifiers are the Stochastic Gradient Descent (SGD) classifier and Multinomial Naïve Bayes (MNB) Classifier which performed the best on Natural Language Processing and yielded a classification prediction rate 98.56% and 98.47%. Below are the confusion matrix of SGD and MNB.

SGD ->



0.9856502242152466

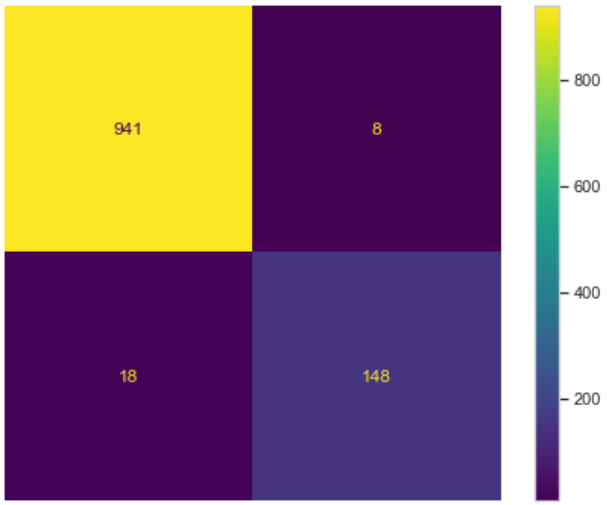


<-MNB

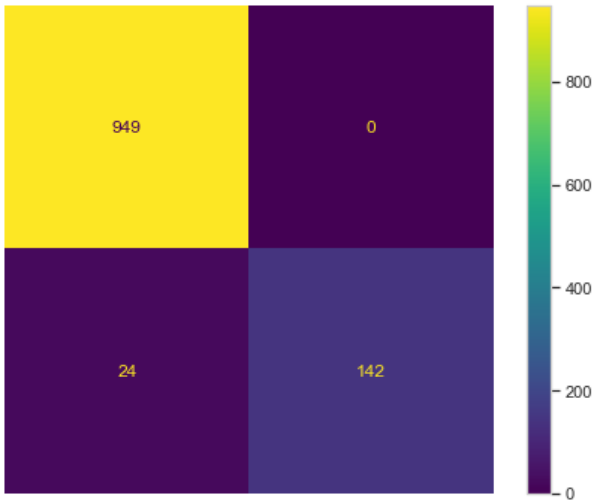
0.9847533632286996

This makes sense since Naïve Bayes has always played an essential role and yields accurate NLTK analysis results. Stochastic Gradient Descent surprisingly has high classification accuracy since it estimates the features (words) one at a time. The entire model is updated with a learning rate.

DT ->



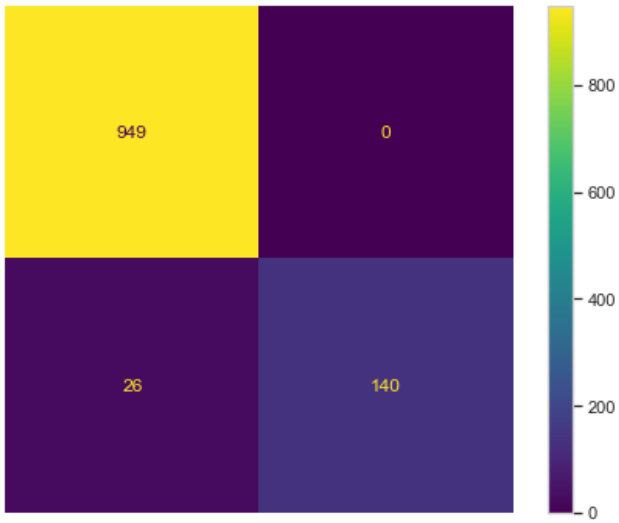
0.9766816143497757



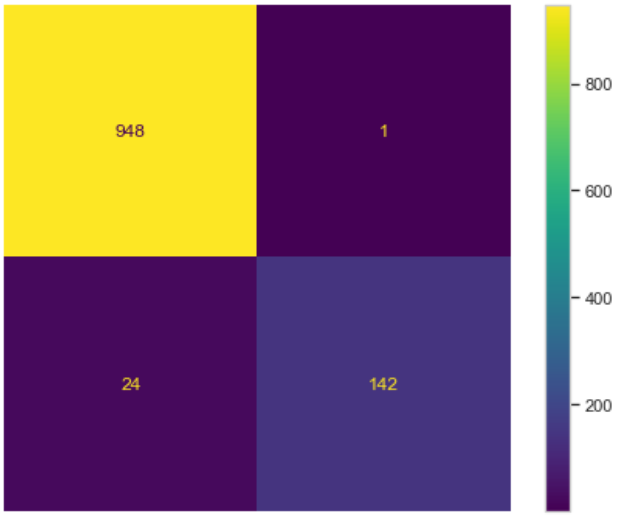
<- LR

0.97847533632287

SVM -

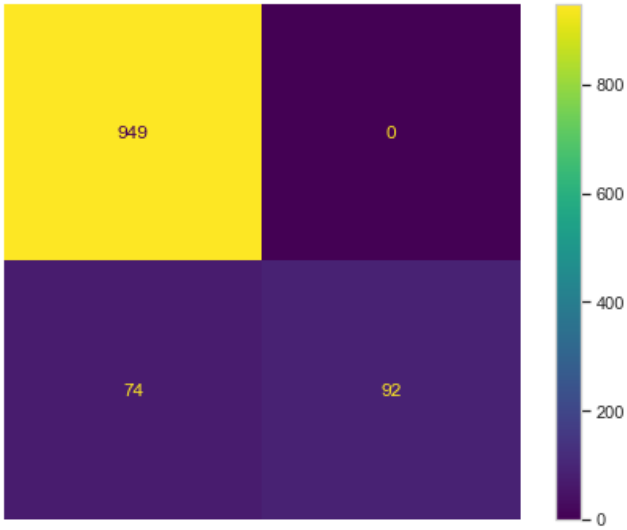


0.9766816143497757



<- RF

0.9775784753363229



<- KNN

0.9336322869955157

The algorithm with the lowest classification accuracy is KNN since KNN relies on clustering data. When coming to ham/spam classification, clustering is a wildly inaccurate method, thus yielding poor results.

Conclusion:

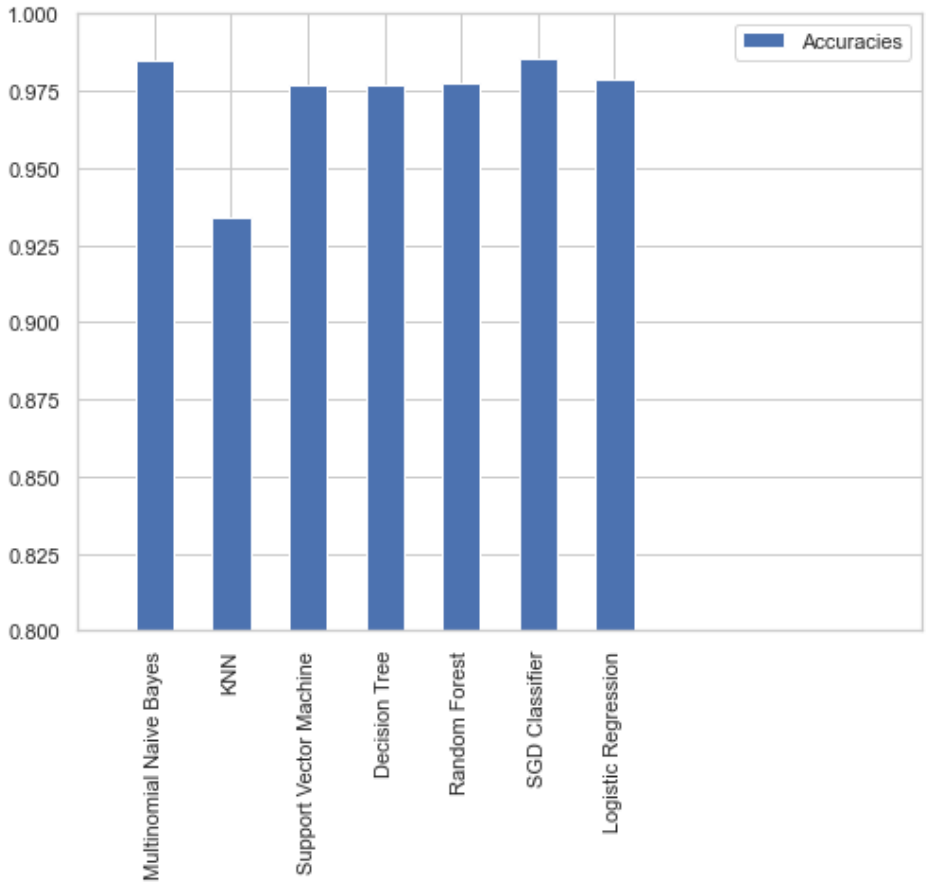
As it is observed:

- SGD Performed the best, giving 98.56%
- MNB: 98.47%
- LR: 97.84%
- RF: 97.77%
- SVM & DT: 97.66%
- KNN: 93.3%

As seen from the classification accuracies, SGD Classifier seems to perform the best out of all the classifiers used in this project. Linear SVM's and Naïve Bayes have always been the topmost used Text Classification algorithms. We might wonder why a linear SVM works much better than our standard SVM method used in the project.
This is simply because text data is easily linearly separable. Hence, it makes the feature distinction for the linear SVM much more accessible and gives us the highest accuracy of all the classifiers we used.

In conclusion, our model gives us excellent accuracy, with all of them being over 90%. We also see that the models create an exceedingly small percentage of False Positives, which is a critical factor during spam classification so that valuable information is not classified as garbage which may cause the user more harm than good.

We also tried to use the Voting Classifier to get even better accuracy. However, even that could not provide better accuracy when compared to linear SVM (SGD Classifier).



References:

- Awad, Mariette & Saab, Salwa & Mitri, Nicholas. (2014). Ham or Spam? A comparative study for some Content-based Classification Algorithms for Email Filtering. Proceedings of the Mediterranean Electrotechnical Conference - MELECON. 10.1109/MELCON.2014.6820574.
- V. K. Vijayan, K. R. Bindu and L. Parameswaran, "A comprehensive study of text classification algorithms," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, pp. 1109-1113, doi: 10.1109/ICACCI.2017.8125990.
- A. Basu, C. Walters and M. Shepherd, "Support vector machines for text categorization," 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the, 2003, pp. 7 pp.-, doi: 10.1109/HICSS.2003.1174243.
- M. A. Kumar and M. Gopal, "An Investigation on Linear SVM and its Variants for Text Categorization," 2010 Second International Conference on Machine Learning and Computing, 2010, pp. 27-31, doi: 10.1109/ICMLC.2010.64.
- https://towardsdatascience.com/spam-or-ham-introduction-to-natural-language_processing part-2-a0093185aebd
- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8404347
- http://www.piazza.com/class_profile/get_resource/ksmc0219anw4mf/ktkojsi4hfb2vq
- http://www.piazza.com/class_profile/get_resource/ksmc0219anw4mf/kt0n3xanr784jd
- http://www.piazza.com/class_profile/get_resource/ksmc0219anw4mf/ktxj7dbc9vm72z
- http://www.piazza.com/class_profile/get_resource/ksmc0219anw4mf/ku7jtkylo7z3by