Applied Machine Learning - Mini Project 2

Nishant Mishra, Shubham Chopra and Aarash Feizi

McGill University

Group 60

ABSTRACT

Aims. We analyze different Machine Learning models to process Reddit data and develop a supervised classification model that can predict what community a certain comment came from.
Methods. We experimented numerous models with different configurations for this task. The models chosen were both simple models, such as multi-class Naive Bayes and also pre-trained complex models, such as LSTMs. After fine-tuning the best performing models' hyper-parameters, to further boost the classification accuracy, we combined their output, using the stacking technique. By doing so, we were able to gain a higher accuracy compared to each model individually on the final validation data.
Results. The final stacked model was able to gain an accuracy of 60.5% on the validation data, and an accuracy of 60.077% on the test data in the competition.

Key words. **Natural Language Processing– Text classification – LSTM – ULMFiT – Naive Bayes – TF-IDF – Support Vector Machines**

1. Introduction

We analyze text from the website Reddit, and develop a multi-label classification model to predict which subreddit (group) a queried comment came from. Reddit is an online forum, where people discuss various topics from sports to cartoons, technology and video-games. The dataset is a list of comments from 20 different subreddits (groups/topics). This problem can be formulated as a type of Sentiment analysis problem, which is quite well-known in the Natural Language Processing (NLP) literature. Sentiment analysis is a computational approach toward identifying opinion, sentiment, and subjectivity in text.

For this dataset, we implement a Bernoulli Naive Bayes classifier, train and test it against the dataset. We also analyze various different models for improving the classification accuracy, including Support Vector Machines, Logistic Regression, k-Nearest Neighbours, the Ensemble method of Stacking and a Deep Learning model ULMFiT (**J.Howard and S.Ruder**, 2018[1]).

We compare the accuracy of these models for different Feature extraction methods, namely Term Frequency-Inverse document frequency (TF-IDF), Binary and Non-Binary Count Vectorizer. We also analyze the performance gain/loss after applying Dimensionality reduction methods on the dataset. In particular, we explore the Principle Component Analysis (PCA) inspired method of Latent Semantic Analysis (LSA).

1.1. Preliminaries

1.1.1. Naive Bayes

Naive Bayes methods are a set of probabilistic supervised learning algorithms centered on applying the Bayes' theorem with a strong "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states that given class variable y and dependent feature vector x1 through xn,

$$P(y \mid x1, .., xn) = \frac{P(y)P(x1, .., xn \mid y)}{P(x1, .., xn)}$$

Using the naive conditional independence assumption that,

$$P(xi \mid y, x1, .., xi-1, xi+1, ..xn) = P(xi \mid y)$$

Since $P(x1, .., xn)$ is constant given the input, we can use the following classification rule,

$$P(y \mid x1, .., xn) \propto P(y)\prod_{i=1}^{n} P(xi \mid y)$$

$$\propto log(P(y)) + \sum_{i=1}^{n} log(P(xi \mid y))$$

1.1.2. Multiclass Bernoulli Naive Bayes

For data that is distributed according to Multivariate Bernoulli distributions we use this Naive Bayes algorithm; i.e., each feature is assumed to be a binary-valued variable. The Learning Parameters are,

$$\theta_k = P(y = k)\,; \quad k = 1, 2, .., K - 1$$

$$\theta_{j,k} = P(x_j = 1 \mid y = k)\,; \quad k = 1, 2, ..., K$$

The Decision Rule for Multiclass Bernoulli Naive Bayes can be given as,

$$P(y \mid x) = log(\theta_k) + \sum_{j=1}^{m} x_j\, log(\theta_{j,k}) + (1 - x_j)\, log(1 - \theta_{j,k})$$

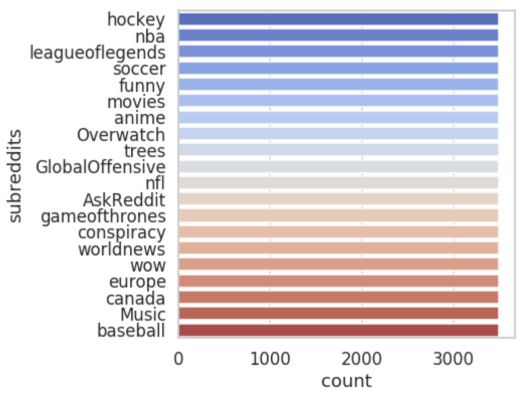

Fig. 1. Class Labels for the input DataSet.

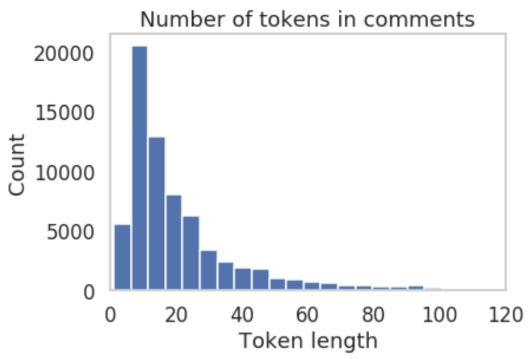

Fig. 2. Token Per Comment Distribution.

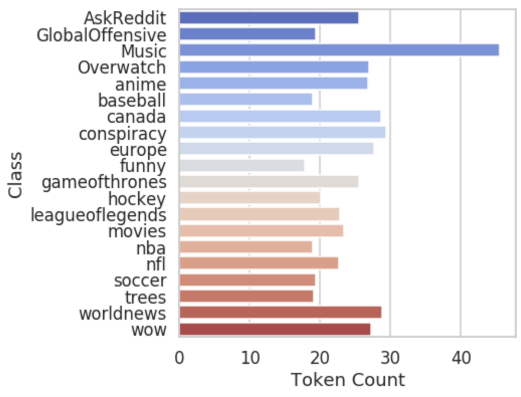

Fig. 3. Number of tokens per each class.

2. Related Work

Kowsari et al.[12], compare the performance of various models for text classification, including Logistic Regression, Naïve Bayes classifier, k-nearest Neighbor, Support Vector Machine, Decision Tree, Boosting, Bagging, Conditional Random Field (CRF), Random forest, and Deep Learning.

More recently, the community has been interested in Deep Learning based approaches for NLP. (**J.Howard and S.Ruder**, 2018[1]) describe a transfer-learning based model ULMFiT, where they propose a novel method to fine-tune a pre-trained NLP model to suit a given Text Classification task. **Liang et al.**[5], **Zhang et al.**[6], and **Sun et al.** [7] discuss novel deep learning models for Aspect based Sentiment Analysis.

3. Dataset and Preparation

The given dataset is a collection of 70000 Reddit comments (in English) that each belong to one of 20 different subreddit categories: AskReddit, GlobalOffensive, Music, Overwatch, Anime, Baseball, Canada, Conspiracy, Europe, Funny, Game of Thrones, Hockey, League of Legends, Movies, NBA, NFL, Soccer, Trees, World news, and WOW. First, we examined the balance between the number of comments per each class for classification, and as it can be seen in Figure 1, all subreddits have exactly 3500 comments. Second, as shown in Figure 2, we plotted the distribution of the number of words in each comment.

To preprocess the data, we cleaned the data by only keeping the alphabetic characters, assuming there is no information regarding the comment's class in numbers and other characters. We omitted Stop-Words, eliminated too Frequently and In-Frequently occurring words (different min_df and max_df parameters) and Lemmatized the remaining words. However, we did not perform stemming or omit hyperlinks, as after initial experiments, we concluded that stemming and omitting hyperlinks from the text decreased the classification accuracy on the validation set. For validation, we performed a 75-25 split, i.e., separated 75% of the data for training our models and kept 25% of the data for validation.

For the Logistic Regression and the Naive Bayes model, we vectorized the comments by two different approaches: count vectors and term frequency - inverse document frequency (TF-IDF). Also, we experimented different n-grams for features. According to our experiments, TF-IDF with unigrams achieved the most accuracy among other methods, as discussed in detail in section 5.1.

4. Proposed Approach

We trained and evaluated several models, which are quite well-known in the Machine Learning Literature including Bernoulli Naive Bayes (described in Section 1.1), Logistic Regression, Support Vector Machines, Decision Trees and k-nearest neighbour classifiers. We also stacked these methods to build a new model. Later, we also evaluated a Deep Learning model ULMFiT. The basics of these models are discussed briefly.

4.1. Logistic Regression

Logistic Regression is a probabilistic supervised learning algorithm which learns a decision boundary between classes. It models the relative probabilities of the classes as the weighted linear combination of input features i.e,

$$\frac{P(y=1|x)}{P(y=0|x)} = \dot{\theta}^T X.$$

The individual probability of classes are then calculated by applying the logistic function, the *sigmoid* function, to convert them to probability scores for each class.

$$P(y=1|x) = \sigma(\theta' x).$$

Logistic Regression is a Discriminative model in that it directly models the class probabilities. Learning in logistic Regression is done using gradient descent optimization of log likelihood or the cross entropy loss given by:

$$log(L(D)) = \sum_{i=1}^{n} y_i \sigma(w^T x_i) + (1-y_i)(1-\sigma(w^T x_i)).$$

4.2. Support Vector Machines

Support Vector Machines are a supervised non-probabilistic binary linear classifier, i.e., it classifies data based on a decision boundary. For a given training dataset of the form (($\boldsymbol{x1}$, y1),..,($\boldsymbol{xn}$, yn)), a hyperplane can be written as $\boldsymbol{w}.\boldsymbol{x} - b = 0$. That is, we need to find the maximum-margin hyperplane that divides the group of points $\boldsymbol{xi}$ for which $\boldsymbol{yi}$ = 1 from the group of points for which $\boldsymbol{yi}$=-1, which is defined so that the distance between the hyperplane and the nearest point $\boldsymbol{xi}$ from either group is maximized.
For data that is linearly separable, for $\boldsymbol{w}$ distance between the decision hyperplane and a point,

$$\boldsymbol{w}.\boldsymbol{x} - b >= 1, if y_i = 1.$$

$$\boldsymbol{w}.\boldsymbol{x} - b <= -1, if y_i = 0.$$

We can put this together to get the optimization problem:
"Minimize $||\boldsymbol{w}||$ subject to for yi($\boldsymbol{w}.\boldsymbol{x}$ - b) >= 1; i = 1,..,n."
For data that is not linearly separable, we define a hinge loss function $max(0, 1 - y_i(\boldsymbol{w}.\boldsymbol{x}_i - b)$. Our problem then can be written as a minimization problem, where we wish to minimize, such that,

$$\frac{1}{n}\sum_{i=1} n\, max(0, 1 - yi(\boldsymbol{w}.\boldsymbol{x}_i - b) + \lambda(||\boldsymbol{w}||^2)$$

is minimum for the dataset. Here, the parameter λ determines the trade-off between increasing the margin size and ensuring that the $\boldsymbol{xi}$ lies on the correct side of the margin. Thus, for sufficiently small values of λ, the second term in the loss function will become negligible, and the SVM will make decisions similar to the hard boundary SVM.

4.3. k-Nearest Neighbours

k-Nearest Neighbours-based classification is a type of lazy learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

4.4. Stacking

Stacking is the process of combining the predictions of several conceptually different base estimators built with a given learning algorithm, for improving robustness over a single estimator. This can be achieved by using a majority vote or the average predicted probabilities (soft vote) to predict the class labels for a queried point. Such a stacked classifier can be useful for a set of equally well performing models and balance out their individual weaknesses. In our experiments, we use a Soft Vote (average the probabilities) of 'n' different estimators to predict the class labels.

4.5. Universal Language Model Fine-tuning (ULMFiT)

ULMFiT is a transfer learning approach in Natural Language Processing where the key idea is to take a pre-trained language model on a huge corpus, (in our case, Wikitext-103 corpus(103M tokens)) and then fine-tune it on the downstream task data, such as our Reddit comment data data in a discriminative way. Now we used this fine tuned language model as the encoder for downstream tasks, Text Classification in our case. The classifier we used with this encoder was a 3 layer LSTM model followed by linear models with BatchNorm and dropout and corresponding activations at each layer.

Table 1. Accuracy on Logistic Regression with different configurations.

Vector	N-gram	Remove Stop-words	Accuracy
Count	unigram	Yes	53.67%
Count	bigram	Yes	53.49%
TF-IDF	unigram	Yes	54.19%
TF-IDF	bigram	Yes	52.71%
TF-IDF	unigram	No	54.23%
TF-IDF	trigram	Yes	51.64%

Table 2. Accuracy on Naive Bayes with different configurations.

Vector	N-gram	Remove Stop-words	Accuracy
Count	unigram	Yes	54.64%
Count	bigram	Yes	53.91%
TF-IDF	unigram	Yes	56.16%
TF-IDF	bigram	Yes	55.53%
TF-IDF	unigram	No	55.86%
TF-IDF	trigram	Yes	55.2%

Table 3. Accuracy on SVM with different configurations.

Vector	N-gram	Remove Stop-words	Accuracy
Count	unigram	Yes	53.4%
Count	bigram	Yes	53.1%
TF-IDF	unigram	Yes	55.23%
TF-IDF	bigram	Yes	55.61%
TF-IDF	unigram	No	55.37%
TF-IDF	trigram	Yes	55.41%

4.6. Dimensionality Reduction - Latent Semantic Analysis (LSA)

Latent Semantic Analysis (LSA) is a technique in NLP, in particular distributional semantics, for analyzing relationships between a set of documents and the terms they contain by producing a set of conceptual features related to the documents and terms (**S. Deerwester et. al**, 1988 [14]). LSA is based on the assumption that words close in meaning will occur in similar pieces of the text (the distributional hypothesis). A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. Paragraphs are then compared by taking the cosine of the angle between the two vectors formed by any two columns. These feature values can be used to reduce dimensions by pruning un-important features. Values close to 1 represent very similarity in paragraphs, while values close to 0 represent dissimilarity in paragraphs.

5. Results and Discussions

5.1. Analysis of different Feature Extraction Methods

For initial feature extraction, we used different Feature extraction methods, namely Term Frequency-Inverse document frequency

Table 4. Accuracy with different classifiers.

Classifier	Hyper-Parameters	Accuracy
Naive Bayes (Ours)	Smoothing, α = 0.01	51.4%
Naive Bayes (sklearn)	Smoothing, α = 0.01	57.52%
Logistic Regression	Solver = lbfgs	56.54%
SVM	Hinge Loss	57.61%
kNN	k = 100	54.37%
ULMFiT	Fine-Tuned	58.123%

Table 5. Run-Times of different classifiers.

Classifier	Hyper-Parameters	Run-Times
Naive Bayes (Ours)	Smoothing, α = 0.01	9.43s
Naive Bayes (sklearn)	Smoothing, α = 0.01	0.43s
Logistic Reg.	Solver = lbfgs	18.45s
SVM	Hinge Loss	5.78s
kNN	k = 100	5.37s
ULMFiT	3+10 epochs	> 2 hours

Table 6. Accuracy after feature reduction with LSA.

Classifier	Features	Hyper-Parameters	Accuracy
Naive Bayes	5000	α = 0.01	51.0%
Naive Bayes	10000	α = 0.01	53.42%
Naive Bayes	25000	α = 0.01	54.52%
Logistic Reg	5000	Solver = lbfgs	52.678%
Logistic Reg	10000	Solver = lbfgs	53.74%
SVM	5000	Hinge Loss	52.61%
SVM	10000	Hinge Loss	53.56%
SVM	25000	Hinge Loss	54.34%
kNN	25000	k = 100	49.77%

(TF-IDF), and Binary Count Vectorizer. In addition to the preprocessing described in Section 3, we tested different n-grams for features, including uni, bi and tri-grams. The comparison of validation set accuracy of these different approaches using Bernoulli Naive Bayes Models, Logistic Regression and SVM is shown in Tables 1, 2 and 3, respectively. As visible, we achieved the best results using unigrams with the tf-idf vectorizer. Hence, we continue to use these for the remaining tests in this report.

5.2. Analysis of different Machine Learning Models

We tested the models described in Section 4 on the dataset, and analyzed the performance while fine-tuning the hyper-parameters for each. The tabulated results for these analysis is shown in Table 4.

5.2.1. Run-Times of tested Classifiers

The Run-Times of the classifiers tested are shown in Table 5. As can be seen, — performs the fastest, but is the least accurate. Also, ULMFiT achieves the highest accuracy of all, however, takes several hours (4-5) to Fine-Tune.

5.2.2. Dimensionality Reduction with LSA

We use the Latent Semantic Analysis (LSA) technique as described in Section 4.6, to reduce the number of dimensions/features. Our intuition behind dimensionality reduction was two-fold, (1) to decrease the run-times of the classifiers and (2) to increase the accuracy by removing un-neccesary features and retaining only features with good co-variance with the output labels. However, as can be seen in Table 6, reducing the number of features results in compromising accuracy, although the run-times are reduced.

*5.3. **Stacking - Models used for Kaggle Submission***

While using Multinomial Naive Bayes with appropriate smoothing function on tf-idf vectorizer carried us over the TA baseline with 57.9% accuracy, but going beyond that was quite difficult as most models would plateau somewhere near that value and nominal increases demanded a lot of fine tuning. We observed that the best results were obtained by stacking various combinations of the models described above. For the final submission, we used an ensemble classifier with *'soft'* voting by Stacking SVM, Naive Bayes and Logistic Regression at their optimum parameter settings.which gave an accuracy of 57.97% on our validation data and 58.011% on kaggle public leaderboard. Adding ULMFit to the stack and using a logistic regression on top as meta classifier further bolstered the accuracy to 60.1%.

6. Conclusions

1. The amount of data per class was quite sparse and overall the variance was high. Therefore most training models would Plateau after a certain point (57-58%), as it becomes increasingly difficult to model unique word-vector representations and inter relations.
2. Statistical Machine Learning Models of Naive Bayes, Logistic Regression and SVMs performed quite well. However, techniques like TF-IDF are very elementary for encoding text, as compared to the State of the Art in NLP, such as word2vec,BERT and word vectors representation.
3. Deep Learning models such as ULMFiT pre-trained on large available datasets helped in improved prediction accuracies. Also, training an LSTM on top of ULMFiT helped the model learn long-term dependencies in the text to further bolster the accuracy.
4. Stacking provided a considerable boost in increasing the final test accuracy. Individually, no model provided accuracy greater than 59%. However, Stacking various models (LR, Naive Bayes, SVM and ULMFiT) helped achieve an accuracy of 60.5% on the validation set and 60.077% on the the Competition Test Set. Mostly stacking helps by taking the strengths of all the models on various features.

7. 5. Future Work

More data points should be collected. Since, the per class data was relatively small, Naive Bayes, Logistic Regression, SVMs performed almost at par with complex Deep Learning models, despite making assumptions on the data. However, more advanced word encoding methods, used by modern models like BERT, (**Devlin et al.** []) and XLNet (**Yang et al.** []) should dramatically improve performance on the dataset. But most deep learning algorithms are very data intensive. Techniques for better feature selection should be researched. LSA did not help much in this case, however, better feature selection and dimensionality reduction techniques can help bolster the accuracy further.

8. 6. Statement of Contributions

This project is a collaborative effort between three team members. The tasks were evenly distributed and every member contributed equally.

References

[1] Howard, Jeremy, and Sebastian Ruder. (2018). "Universal language model fine-tuning for text classification."

[2] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

[3] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding.

[4] Jin, Chi; Wang, Liwei (2012). Dimensionality dependent PAC-Bayes margin bound. Advances in Neural Information Processing Systems.

[5] Yunlong Liang, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen and Jie Zhou (2019). A Novel Aspect-Guided Deep Transition Model for Aspect Based Sentiment Analysis.

[6] Chen Zhang, Qiuchi Li and Dawei Song. (2019). Aspect-based Sentiment Classification with Aspect-specific Graph Convolutional Networks.

[7] Kai Sun, Richong Zhang, Samuel Mensah, Yongyi Mao and Xudong Liu. (2019). Aspect-Level Sentiment Analysis Via Convolution over Dependency Tree.

[8] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke S. Zettlemoyer. (2018) Deep contextualized word representations, NAACL-HLT, 2018.

[9] Alexandra Chronopoulou, Christos Baziotis, Alexandros Potamianos. (NAACL-HLT, 2019). An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models.

[10] Bryan McCann, James Bradbury, Caiming Xiong, Richard Socher Learned in Translation: Contextualized Word Vectors, NIPS 2017.

[11] Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks" (PDF). Machine Learning. 20 (3): 273–297.

[12] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes and Donald Brown. ICL, 2019. Text Classification Algorithms: A Survey.

[13] Bo Liu. (2019). Anonymized BERT : An Augmentation Approach to the Gendered Pronoun Resolution Challenge.

[14] Deerwester; Scott C. (Chicago, IL), Dumais; Susan T. (Berkeley Heights, NJ), Furnas; George W. (Madison, NJ), Harshman; Richard A. (London, CA), Landauer; Thomas K. (Summit, NJ), Lochbaum; Karen E. (Chatham, NJ), Streeter; Lynn A. (Summit, NJ). Computer information retrieval using latent semantic structure.