DS-GA-1003 Final Project - Part-Time Bandits

Project 1: Fake Review Detection

Sammie Kim (sk7327), Antonio Robayo (amr1059), Tim Connor (tfc276),
Alex Spence (ajs811) (Member responsible for submissions)

May 17, 2020

1 Introduction

Machine learning models were developed on New York City restaurant reviews to detect genuine vs. fake reviews. It is important for companies such as Yelp to maintain genuine reviews on their platform to preserve the integrity of their product and continue making a profit. Too many fake reviews on their site would result in lost customers (restaurant users and restaurant owners) resulting in profit loss. The approach to developing a robust machine learning fake review detection model was to perform feature transformations on the existing data, perform modeling on each of the transformed data sets, and perform hyperparmeter tuning on each combination to select the result which maximized average precision (AP) and Area Under the ROC Curve (auROC) on the validation set. A 80% training 20% validation split was used. The combination which had the best results on the validation set was Logistic Regression with L2 regularization and therefore is the recommended model for implementation.

The contributions of each team member is shown below in Table 1.

Task	Sammie	Antonio	Tim	Alex
Data Transformations	5%	40%	50%	5%
Model Training	30%	30%	10%	30%
Model Evaluation	30%	30%	10%	30%
Report	25%	25%	25%	25%

Table 1: Contributions

2 Problem Definition and Algorithm

2.1 Task

Machine learning algorithms were developed to distinguish between genuine and fake restaurant reviews. The inputs of this problem includes the following:

- restaurant reviews
- features for each review (user rating, product rating, word count of review, etc. explained further in "data" section)
- feature transformations (i.e. n-gram, bag of words, up-sampling)
- data preprocessing (i.e. WordNetLemmatizer, SnowballStemmer)
- model types (L1/L2 logistic regression, naive Bayes, support vector machine)

- hyperparameter inputs for each model (e.g. regularization strength, choice of loss function)

The output of the problem is a fully functioning model which can predict whether a review is fake or genuine review to some accuracy. The final model is selected based on maximizing average precision and rocAUC on the validation set.

2.2 Algorithm

The binary classification algorithms used for model training were L1 and L2 Logistic Regression, Naive Bayes, and Support Vector Machine (SVM).

L1 and L2 Logistic Regression:

At a high level, Logistic Regression works by fitting an "S" shaped curve to observations based on the feature values and target variable. During model training, weights are assigned to each feature through gradient descent such that the weight vector is updated by a factor of the gradient of the cost function until the cost function is minimized. The cost function in Logistic Regression is the "Sigmoid" function which is an S shaped curve. In L1 and L2 logisitc regression, a factor of the L1-norm and L2-norm, respectively, is added to the cost function and is meant to penalize large weight vectors. This regularization helps to reduce overfitting of the training data. At the end of this process, weights are assigned to each feature, and the model is evaluated on the validation set to assess for generalization.

Naive Bayes:

Naive Bayes is a probabilistic model based off of Bayes Theorem, which states that the probability of a hypothesis (in this case, genuine vs. fake review) can be estimated if evidence is known about said hypothesis (the words present in a Yelp review). Naive Bayes assumes that our features are conditionally independent given the label. This means we are assuming that the presence of one word x_i is independent of another word x_j, where $i \neq j$. A more formal statement of our problem:

$$p(y|\mathcal{D}) \propto p(y) \prod_{i=1}^{n} p(x_i|y)$$

$$\hat{y} = \text{argmax}_y \; p(y) \prod_{i=1}^{n} p(x_i|y)$$

What this is saying is that the likelihood of a review being ’fake’ is given by the likelihood of the ’fake’ label (as observed in our data) multiplied by the likelihood of words x_i, for $i = 1, \ldots, n$, appearing together given that the label is ’fake’. When classifying a review, we take the more likely of the two classifications, ’fake’ vs. ’genuine’.

SVM:

Training a SVM model involves finding a line or hyperplane which separates data of two classes based on the feature values and target variable. The goal is to maximize the margin, which is the distance between the hyperplane and the support vectors. The support vectors are the data points which lie closest to the separating line or hyperplane. Once the optimal separating line/hyperplane is found, it is used to predict the target variables in a validation dataset to assess for generalization.

3 Experimental Evaluation

3.1 Data

Data processing pipelines were informed after careful review of the training data. Exploratory analysis revealed that 95% of reviews were in English, with no dominant second language. However, only ≈38% the corpus was recognized as English. Taken with the observation that 35% of words only appeared once, this suggests a high prevalence of misspellings in the data (slang would likely appear more frequently). As such, data processing was employed to mitigate the risk of over fitting to the sparse feature set:

- **Stemming & Lemmatization**: Both techniques attempted to eliminate suffixes and combine conjugates based-on their root. The Snowball Stemmer extracts the stem or root of a word by simply removing the end corresponding to a conjugation (e.g. `computing` → `comput`). The Lemmatizer, by contrast, derives the base or dictionary form of a word (e.g. `computing` → `compute`). Our results showed the Snowball Stemmer outperformed Lemmatization and a Porter Stemmer (all from Python's NLTK library).[1]

- **N-grams**: Consist of a contiguous sequence of n items, in this case words from a review. The idea behind incorporating n-grams was to account for context in reviews. For example, the word *bad* by itself may be an indicator of a negative review but there's no certainty to whether it's a descriptor of the restaurant experience. By contrast, *a bad meal* is more definitively an indicator of a negative review relating to the restaurant experience. For our models, we included unigrams, bigrams, and trigrams. Limitations in computation power and the curse of dimensionality prohibited us from testing larger n-gram ranges.

- **Word Count & TF-IDF**: Term frequency-Inverse document frequency (TF-IDF) improves on word count by accounting for the frequency of a word in the full corpus. Further, TF-IDF values are normalized to sum to 1 for each review. This prevents review length from influencing our textual analysis.

- **Remove infrequent, non-english and stopping words**: While TF-IDF reduces the risk of overfitting infrequent words. Shrinking the input space led to faster training, potentially reducing approximation error (improved tuning) optimization error. English words were identified using Python's Enchant library. Built on a spell-checker, this library recognizes conjugates and proper nouns unlike other common corpus like WordNet.[2]

- **Upsampling**: The training data was comprised of 90 % genuine reviews vs. 10% fake reviews. Upsampling to to parity (i.e. 50% real 50% fake) was attempted to circumvent issues associated with exhibited class imbalance. Fortunately, the evaluation metrics (AUC and Average Precision) are known to be more robust to class imbalance.[3] The empirical results were consistent with the theory. Upsampling did not improve results, and in some cases detracted (presumably due to overfitting to the minority class examples).

Summary variables were engineered to supplement the tokenized word features.

Feature	Rationale
Review Features	
Word Count	Evaluate review length explicitly since it is stripped out via TF-IDF
Percent English	Fake reviews may be more likely to contain misspellings
User & Product Features	
Average Rating	Fake reviews may skew towards extremes (promoting self or harming competition)
Number of Reviews	Fake users may be more active than regular users
Review Intensity	Modification of of reviews (reviews / (1 + last review date - first review date))

Table 2: Additional Features

[1] Bird et Al (2009)
[2] Kelly (2011)
[3] Davis & Goadrich (2006)

3.2 Methodology

The approach to developing a robust machine learning fake review detection model was to to perform feature transformations on the existing data, train models on each of the transformed data sets, and perform hyperparmeter tuning for different combinations of parameters to select the result which maximizes average precision (AP) and Area Under the ROC Curve (auROC) on the validation set.

For both types of Logistic Regression, regularization parameter, lambda, was tuned on each training set. `np.logspace` was used to create a list of evenly spaced intervals of base 10 to model with each of the transformed data sets. Models were trained using scikit-learn and the result was used to predict target variables of the validation set. The AP and auROC on the validation sets were evaluated, and lambda range was fine-tuned until the validation performance converged. Shown in Table 3, The best results for L1 Logistic Regression were achieved with TF-IDF on single words, lambda=0.25 and surprisingly no additional preprocessing. For L2 Logistic Regression, the best performing combination was included the ngram transformation coupled with TFIDF, WordNetLemmatizer preprocessing, and a lambda of 1.678.

For SVM, regularized linear models were implemented with stochastic gradient descent (SGD), also using scikit-learn.[4] Hinge, Perception and Log Loss were explored as options to measure model fit, and compared results from L1 and L2 regularization terms that penalize model complexity. The results showed that L2 outperformed L1 across all loss functions. Out of the three loss functions, Percepton performed the poorest, Hinge resulted in relatively lower auROC and higher AP vs. the opposite for Log Loss. Since our dataset is heavily imbalanced, the loss function with a higher AP seemed more logical as it can provide us with a more accurate prediction of future classification performance. As a result, the best performing SVM combination was the one with Hinge loss and L2 regularization.

3.3 Results

Below, 'lemmatizer' and 'stemmer' refer to `WordNetLemmatizer` and `SnowballStemmer` respectively.

Model	Transformation	Preprocess	Dataset	Parameters	AP	auROC
Logistic Regression (L2)	ngram-TFIDF	lemmatizer	val	$\lambda = 1.678$	0.24	0.74
SVM (L2)	ngram-TFIDF	stemmer	val	loss='hinge', alpha=1e-3	0.23	0.71
Naive Bayes	ngram-TFIDF	stemmer	val	alpha=1	0.21	0.70
Logistic Regression (L1)	words-TFIDF	none	val	$\lambda = 0.250$	0.23	0.73

Table 3: Model Results

4 Conclusions

After some error analysis, it is revealed that the distinction between fake and genuine reviews is very subtle. For example, a close read of two misclassified instances in the validation set show that fake reviews are quite convincing to both our model and humans alike:

> *omg this place was so good. best mexican food so far. not pricey either. Great mole burrito i had. Cant wait to go back. wish they had longer hours*
> **Actual**: Fake
> **Predicted**: Genuine
>
> *A bit on the cozy side and I wouldn't bring your kids there, but overall we had a great experience. My fiancee's entire family came and we are a LARGE group of people. They handled our craziness quite well.*

[4] Pedregosa et al (2011)

Actual: Fake
Predicted: Genuine

Improvements to the model development approach would involve using neural networks to learn an underlying embedding representation for both genuine and fake reviews as some of the fake reviews deal with hyperbole. Training a model that is able to understand this to some degree will help with identifying fake reviews, which is where all of the current models fall short. The models used in this report are based on frequency of ngrams through TFIDF rather than understanding meaning.

5 Bibliography

1. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011
2. Kelly, Ryan (2011). PyEnchant. 2011. Retrieved May 2020. https://pyenchant.github.io/pyenchant/index.html
3. Bird, Steven, Ewan Klein, and Edward Loper (2009), Natural Language Processing with Python, O'Reilly Media.
4. Davis, Jess Mark Goadrich. The Relationship Between Pricision-Recall and ROC Curves. 2006. In the Proceedings of the 23rd International Conference on Machine Learning (ICML).