IST707 Data Analytics

Lie detection

**Scenario:**

Some people claimed that machine learning algorithms can figure out whether a person is lying or not. Do you believe that? To test this claim, we have collected a collection of customer reviews, some are true some are fake, and you are going to test how well you can use the algorithms we’ve covered for fake review detection. Note that this data set also has sentiment label for each review, and you should try to predict that as well.

For both tasks, try Naïve Bayes and SVM (note: you can use the sentiment analysis package to generate sentiment analysis instead of the existing column, but don’t use sentiment analysis to **predict** sentiment), but I’d like you to explore different strategies, and report your results. For example:

1. Different preprocessing methods – e.g., with or without stop-words, lemmatization, reducing the specific tokens you’ve used to maximize information gain.
2. With either category (lie / sentiment) does it help to include the other category as a feature? For the lie feature
3. Use a topic model & sentiment analysis module to generate additional features and use these in combination with / instead of raw tokens.

Try to get the best results you can.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| General Strategy | Parameter settings | Sentiment | | | Lie | | |
| F1 | Precision | Recall | F1 | Precision | Recall |
| (spark) Naïve Bayes | smoothing=1.0, modelType="multinomial" | 0.86 | 1.00 | 0.75 | 0.60 | 0.43 | 1.00 |
| (spark) Naïve Bayes | smoothing=20.0, modelType="multinomial" | 0.86 | 1.00 | 0.75 | 0.55 | 0.**38** | 1.00 |
| (spark)  Support Vector Machine | Linear Kernel  maxIter=10, regParam=0.1 | 0.67 | 1.00 | 0.50 | 0.50 | 0.33 | 1.00 |
| (sklearn)  SVM | SVC kernel  CountVectorizer | 0.49 | 1.00 | 0.66 | 0.45 | 0.42 | 0.48 |
| (sklearn)  SVM | SVC kernel  CountVectorizer  (stop words) | 0.59 | 0.42 | 1.00 | 0.33 | 0.46 | 0.26 |
| (sklearn)  SVM | SVC kernel  TFIDFVectorizer  (stop words) | 0.71 | 0.70 | 0.73 | 0.30 | 0.50 | 0.22 |

**In a write-up following the table, please explain the rationale for the strategies you have chosen, including the theoretical foundation for your choice. Also explain your parameter tuning approach for every attempted strategy. Where did you start? How much of a difference did parameter tuning make? Why?**

I have chosen to use Naïve Bayes and Support Vector Machine. I used both PySpark and Sklearn to build different models and explore text mining outcomes. I found that PySpark had much better results due to the text preprocessing and feature engineering done with Machine Learning Pipelines. I started with feature extraction and transformation. I used Spark to create a Machine Learning Pipeline containing a tokenizer, stop-words remover, count vectorizer (removes words that appear 5 or less times), and an IDF (transforms the filtered words into a TFIDF sparse matrix). A Support Vector Machine Model is used with a Linear Kernel. For parameter tuning the Naïve Bayes, it was a trade-off between accuracy and precision/recall. Increasing the smoothing parameter (the technique that handles the problem of zero probability) increased the accuracy of the model but lowered precision and recall.

In Sklearn, I found that not removing stop words in the vectorizers increased the f1 score and recall for the lie detection SVM model with C Kernel. The SVM model with a linear kernel had higher scores after removing stop words for lie detection. Overall, the Spark Naïve Bayes model achieved the highest classification report for both lie detection and sentiment analysis. Sentiment analysis achieves perfect precision. It is much harder to increase precision for lie detection because most models seem to assume everyone is a liar as it yields perfect recall and very low precision.

For each task and feature set (sentiment classification / lie detection), use the Rank module (Gini and Information Gain Ratio) to rank the features and list top 10 features from each method. Based on these top features, can you understand what patterns the classifiers have learned from the data?

Entropy and Gini are measurements of impurity, or how heterogenous the features are. Information gain quantifies impurity and selects features that minimizes entropy (degree of randomness). This is useful for features selection purposes, to see which X variables have lower probability of having higher entropy. Gini Index uses binary splits to compute differences in entropy, scoring from 0 to 1. The following calculation shows the impurity of the filtered words from the Count Vectorizer pipeline:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Compare performance difference in sentiment classification and lie detection, and tell us which task is harder, and try to explain why.

Sentiment analysis achieves precision very easily, it is much harder to increase precision for lie detection because most models seem to assume everyone is a liar (high recall). Sentiment analysis is easily analyzed and classified based on vocabulary, lie detection requires an intuition of deception that cannot be constructed from vocabulary. A successful approach could be using surface features and lexical features, that looks at frequency and use of words without taking into consideration context.