Sentiment Analysis on Rotten Tomatoes Movie Reviews

Yuliya Astapova, William Murdy, Alexander Tairbekov

The Problem

- Training set of sentences from movie reviews
  - Each labeled with a Sentiment Value from 0 to 4
- Test set of similar sentences
  - Original test set not labeled
- Used Naive Bayes and KNN to train classifiers to determine Sentiment Value of sentences
- Used bag of words as features

Training Set

- First extracted only complete sentences
- Processed each sentence to remove non-letter characters
- Removed stopwords (a, the, is, not, etc.)
  - Set of English stopwords from Natural Language Toolkit
- Used Scikit-learn to learn Bag of Words vocabulary

Bag of Words

- Turn text into vector representations
  - Vectorization
- Sentences described by word occurrences
  - Completely ignores word position
- Represented as matrix
  - Sentence (row) x Word in Vocabulary (col)

Naive Bayes

Our Own Implementation

- Model:

$P(Y, W_1 \ldots W_n) = P(Y) \prod P(W_i \mid Y)$

  - Y = The category; in our case, 0 - 4
  - W_i = A word

Scikit-Learn Implementation

- Used the Multinomial Naive Bayes implementation
- Often used for multiclass text classification
- Others available:
  - Bernoulli - assumes binary-valued features
  - Gaussian - assumes Gaussian distribution of features

k-Nearest Neighbors

- Load training data into DB
  - Construct the parse tree for the sentence
  - Each phrase is hashed and assigned the sentiment value
  - Feature vector is determined by structure of parse tree
- Assign features for test data
  - Construct the parse tree for the test sentence
  - Each leaf of the parse tree is either found in DB or assigned to be a neutral sentiment
  - Select all subtrees from training data that match the parse tree, and use sentiments as features to run KNN

Our Naive Bayes in Detail

- Training
  - To train, first calculated prior probabilities
  - Number of sentences in each category divided by total sentences in the training set
  - For each word in vocab, calculated probability of that word showing up in each category
  - For each word: 5 probabilities associated with each of possible categories
- Classification
  - For each word in the sentence:
    - Add up logs of probabilities of being in each category
    - Add log of prior probabilities for each category
    - Classify by greatest probability

Results

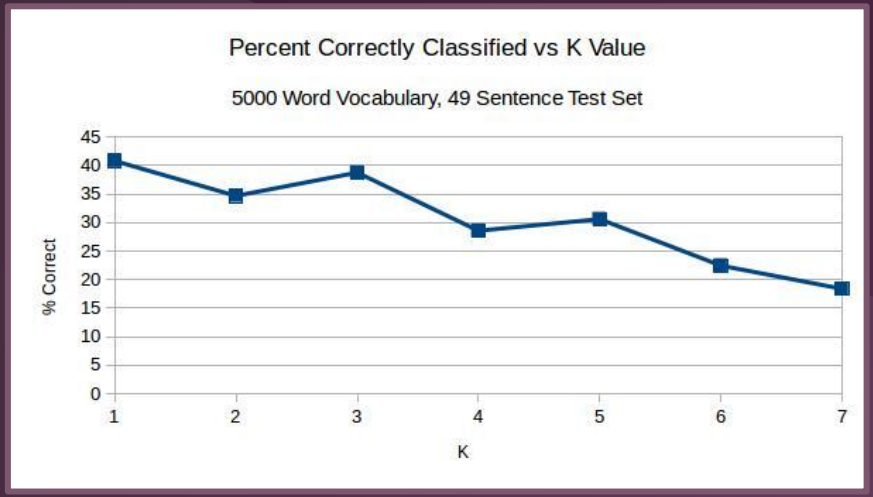
- Training on 8463 reviews with vocabulary of 5000 words
- Test on 49 sentences

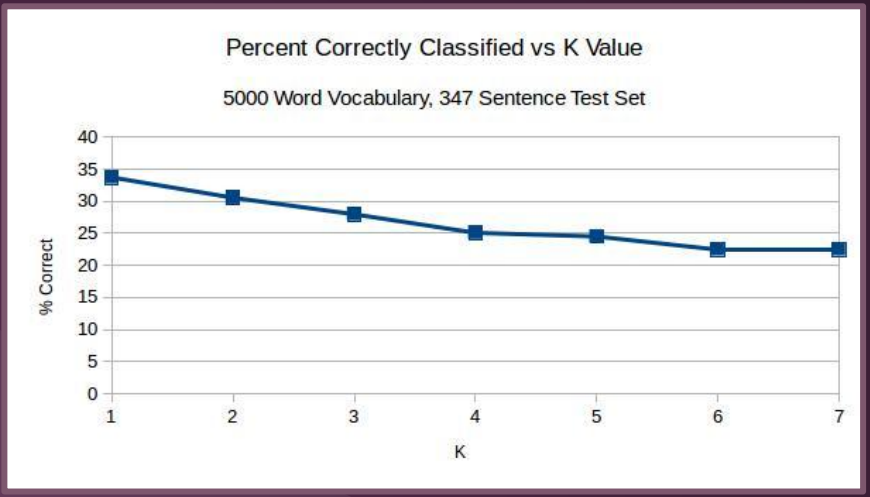
Scikit Naive Bayes:

- 48.9796% classified correctly
- Avg difference in scores: 0.7551

Our Naive Bayes:

- 40.8163% classified correctly
- Settled on K = 1

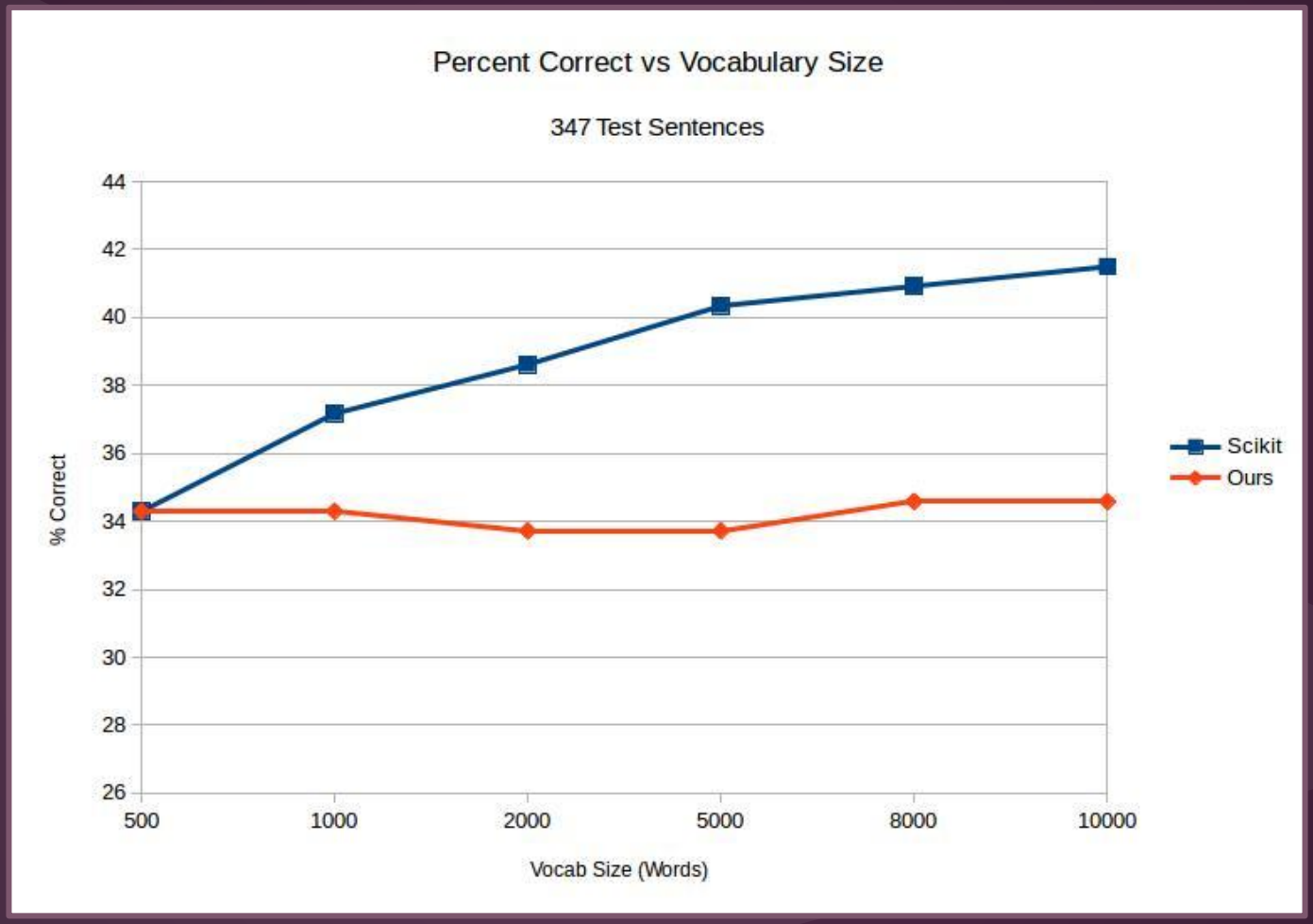




- Scikit: 48.9796% correct
- Ours: 40.8163% correct

- Scikit: 40.3458% correct
- Ours: 33.7176% correct

- Scikit: 8.6338% difference
- Ours: 7.0987% difference


Percent Correct vs Vocabulary Size
347 Test Sentences
% Correct
44
42
40
38
36
34
32
30
28
26
500
1000
2000
5000
8000
10000
Vocab Size (Words)
Scikit
Ours

Conclusions

- Scikit-Learn's implementation of Naive Bayes outperformed ours
- Accuracy increases with vocabulary size
- Bag of Words is not sufficient to describe text for sentiment analysis
  - Sarcasm, idioms, etc.
  - Sentence/phrase negation

Links

- Rotten Tomatoes Reviews Data
  - https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data
- Scikit-Learn
  - http://scikit-learn.org/dev/index.html
- Natural Language Toolkit
  - http://www.nltk.org/
- Kaggle Bag of Words Tutorial
  - https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-1-for-beginners-bag-of-words