Glenn Holzhauer
NLP

Briefly describe how you calculated the probability and used it for classification.

To calculate the probability of a sentence/word's classification, I trained my model by calculating the probability of each word's classification. If the word appeared in more reviews categorically negative, it would be weighted as a more negative word (by probability). These weights were specifically determined with the formulae given in class,

Naïve Bayes – training

- $c_{NB} = argmax_c P(c) \prod_{w_i \in D} P(w_i|c)$
- **What should we do?**
  - $c_{NB} = argmax_c \, [\log(P(c)) + \sum_{i \in positions} \log(P(w_i|c))]$
- So, what we need to learn here?
  - $p(c) = \frac{N_c}{N_{training\ doc}}$
  - $p(w_i|c) = \frac{count(w_i,c)}{\sum_{w \in V} count(w,c)}$

Then, an aggregated class probability from the summation of the word weights classifies the review in question. (If the words are cumulatively more negative or positive, the sentence will be classified as such.)

How did you handle the situation when a word is present in one class but missing in other class?

In the situation where a word is in one class but missing in another, Laplace smoothing is employed to catch all words not yet properly given class probabilities. The program keeps a total "vocabulary" list and iterates through it, adding any missed vocabulary into the class it is missing in, giving it the lowest base class probability. This means the numerator is 1 for the below equation.

- Laplace smoothing
  - $P(w_i|c) = \frac{count(w_i,c)+1}{\sum_{w \in V} count(w,c)+|V|}$

Glenn Holzhauer
NLP

After iterating, both the positive and negative collections of words have all vocabulary as it has been filled with the Laplace smoothing.

How did you handle the situation when a word is missing in both class?

When a word was missing in both classes, it was not considered for classification probabilities. We did not have enough information and only considered words that we had reference for. The Score function only iterated through previously trained vocabulary for classifying a review.

Recall, Precision, F1 score on dev set for SentimentAnalysis class

Recall was 0.6, precision was 1.0, and f1 score was 0.75.

Recall, Precision, F1 score on dev set for SentimentAnalysisisImproved class

Recall was 0.7, precision 1.0, and f1 score was 0.824.

What preprocessing have you added and what was your intuition behind adding those features.

In the improved class, I added a cleaning function to get rid of any and all punctuation and switch all review words to lowercase. With this consistency, the model works more as intended; a capitalized word should not carry different sentimental value, and thus differently cased words skewed and made the model less proficient. The same applied to a word tied to a punctuation which had little impact on the sentimental meaning of the word itself.

A performance comparison of SentimentAnalysis class vs. SentimentAnalysisisImproved class

Note: Adding additional features or pre-processing may not improve your performance of SentimentAnalysisisImproved. However still provide a comparative study of the both classifiers.
Here are few example questions that you may address for this comparison-
Which one is performing better?
Which pre-processing/feature of the improved version did not work as per your expectation?

Glenn Holzhauer
NLP

The improved model is performing better, as intended. The pre-processing feature that made it perform better was the elimination of capitalized letters and punctuation. Words that should have been recognized as the same were not properly recorded in training of the basic model, and the improvement was evident when these adjustments were made. Because the words were more properly weighted, classification was improved.

How correct was your intuition about a feature (after seeing the comparative results)?

My intuition was correct and yielded noticeably higher recall and f1-score, though the precision remained a sturdy 1.0 and was adequately successful in both the improved and default sentiment analysis. When the words that should have been bundled together were properly classified as the same, the model learned better.