App Store Review Analyzer

Robert Ervin
97170118
CIS 481 Computational Learning: Theory and Practice Winter 2016

Goal

- Determine why some users leave low-rated reviews
  - Reason: App ranking is directly affected by the ratio of high-to-low reviews. The less low reviews we have, the higher the ranking is. The higher the the ranking is, the more visible it is in the app store, and the more downloads it gets.

Data Collection

- Select 18 of the top shopping apps, and scrape the most popular 500 reviews for each.
- Save those in a MongoDB database for persistence

Methodology

- Cluster the users into groups
  - Reason: This will allow us to dig into the clusters to analyze the types of words used in each group.
- Analyze the clusters that tend to have reviews with low ratings
  - Reason: This will give us insight into what types of words the users who didn’t enjoy the app use.

Algorithms

- Term Frequency, Inverse Document Frequency vectorizer (TF-IDF)
  - Reason: Converts review strings to term-weight matrix. This essentially creates our features.
- K-Means clustering
  - Reason: Sufficient out-of-the-box clustering algorithm. It gives us the necessary levers to play with the parameters that we will need to.
- Principal Component Analysis (PCA)
  - Reason: Good way to view results graphically in 2 dimensions.

Technology

- Language: Python
- Libraries:
  - Numpy
  - Scikit-Learn
  - Matplot
  - MongoDB
  - IPython Notebook

Procedure

- For each rating, randomly select 500 reviews
- Extract features from all reviews
- Find optimal number of clusters for K-Means
- Cluster the reviews
- Analyze the results

Feature Extraction

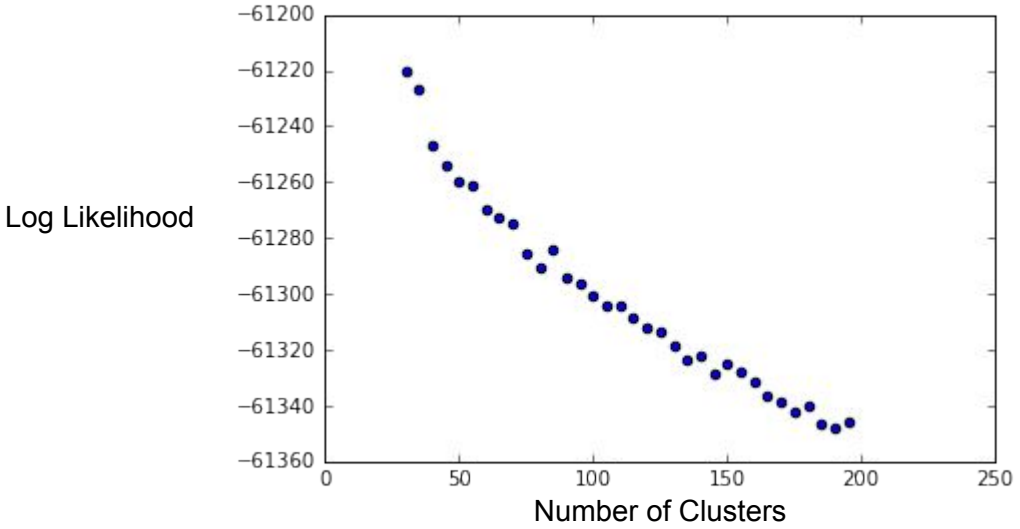
- TF-IDF
  - Term Frequency (tf): The number of times a word occurs in a review
  - Inverse Document Frequency (idf): Weight the words that occur less frequently in the set of reviews more than the words that occur more frequently
  - weight(term) = tf(term) * idf(term)
- Stop Words
  - Ignore common words. e.g.) and, in, the him, her, by, has
- Extra Parameters:
  - Ignore words that occur in more than 80% of the documents
  - Ignore words that occur in less than 2% of the documents
- Results
  - The feature space had 132 terms (dimensions) with these parameters

Optimal Number of Clusters

- Assign a single hold out set
- Using a set of possible cluster sizes, train k-means on the data and compute the log-likelihood for each cluster.
- Maximize the log likelihood among all cluster sizes to obtain the best number of clusters.

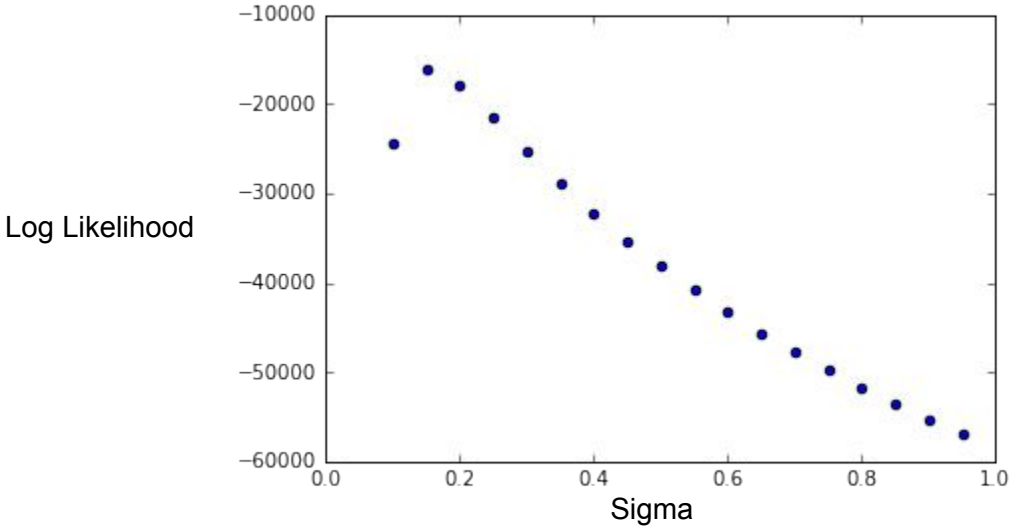
K-Means Assumption

- K-Means assumes that the variance is always 1
- With this assumption, our optimal clustering technique showed continually-decreasing log-likelihoods.

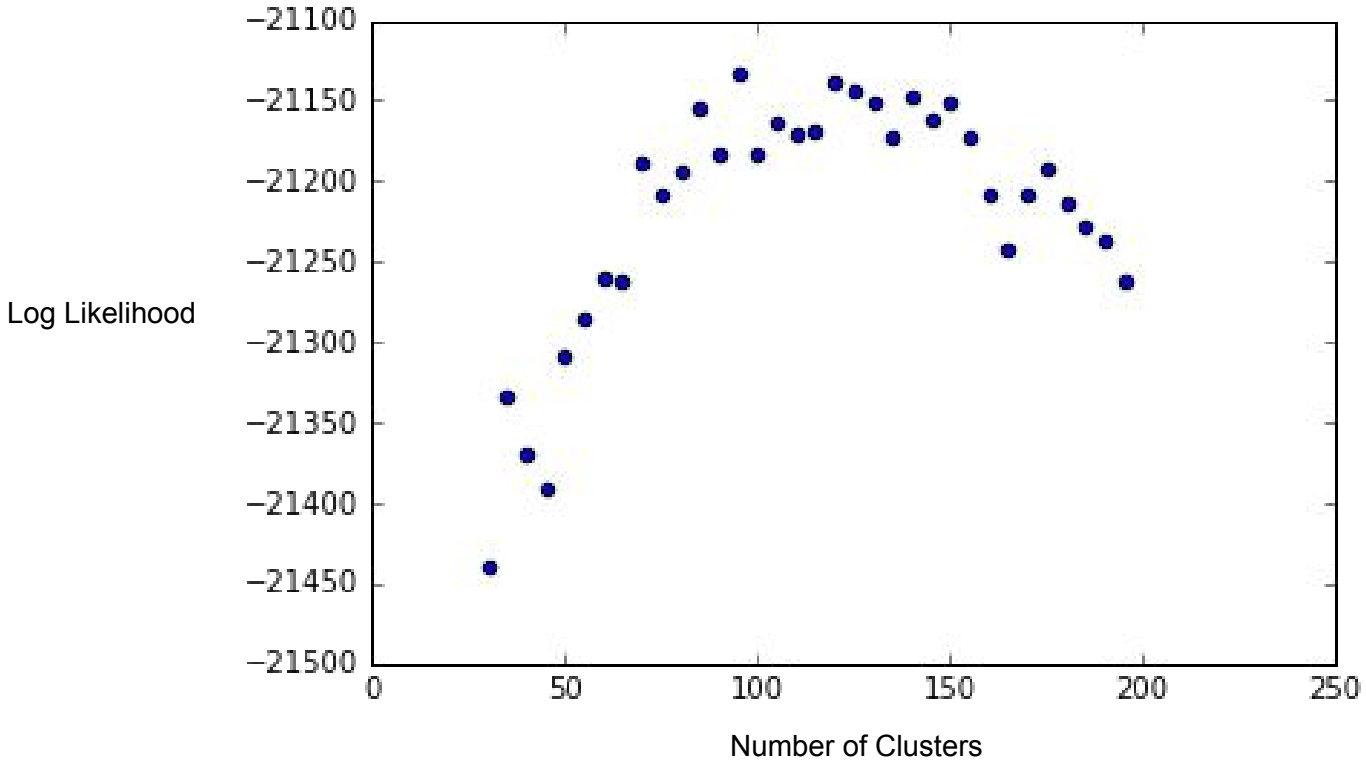


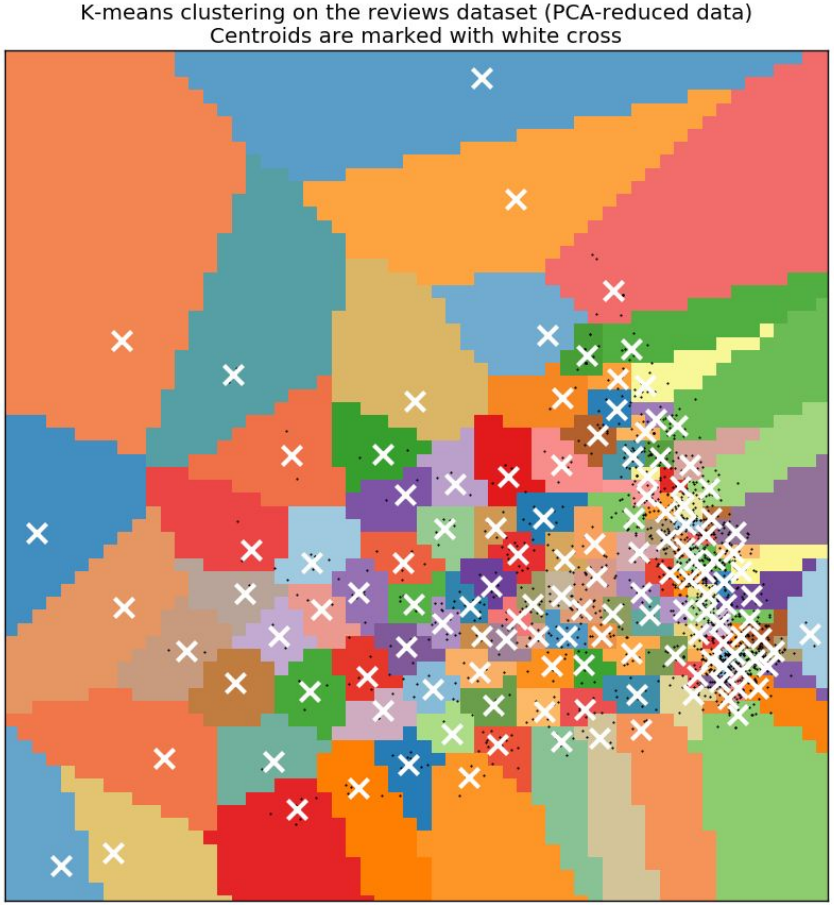
K-Means Assumption (continued)

- Now we had to optimize for sigma as well. We took this into account by simply fixing the number of clusters and iterating over different sigma values when computing the log likelihood.



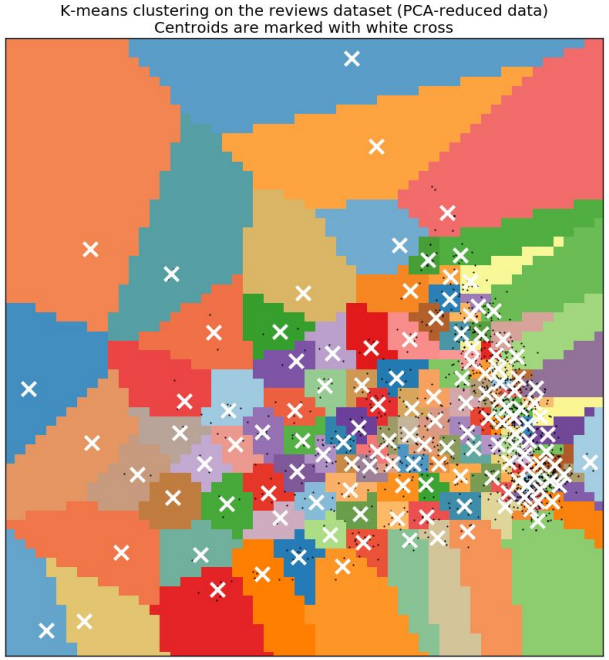
Optimal Number of Clusters with Optimal Sigma




K-means clustering on the reviews dataset (PCA-reduced data)
Centroids are marked with white cross

Cluster Analysis

- No “elbow” of centroids
- Most of the “vertical” axis seems to capture the feature pretty reasonably
- The “horizontal” axis doesn’t seem to capture the points all that well.



Clustering Notes

- The more features I introduced, the lower the optimal sigma became and the higher the number of optimal clusters became
- Had to balance number of features (which require more data), as well as variability of terms for intelligent analysis.

1 Star Reviews Analysis: Procedure

- Select the clusters with the most number of 1 star reviews, and analyze the top terms from the cluster.
- The threshold for a cluster being considered a top cluster was having at least 2% of the 1 star reviews.

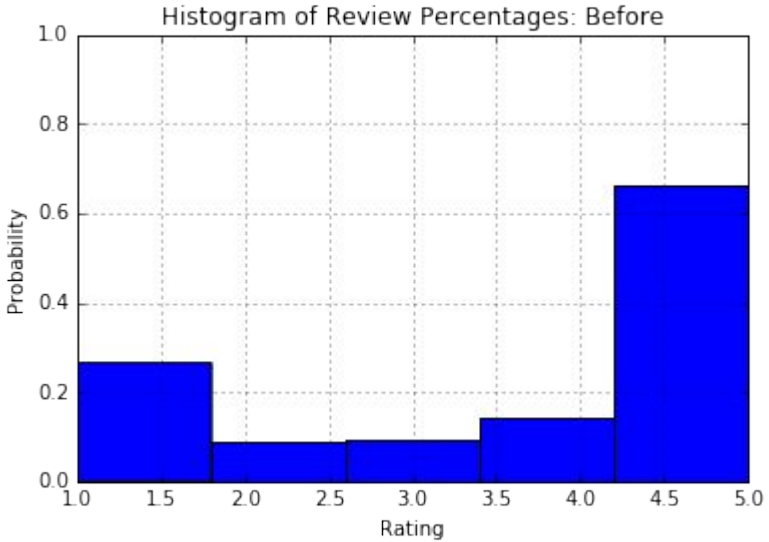
- 131/135 clusters had at least one 1 star review in them.
    - This is a good indication that some 1 star reviews contain words that should not pertain to the most negative rating possible.

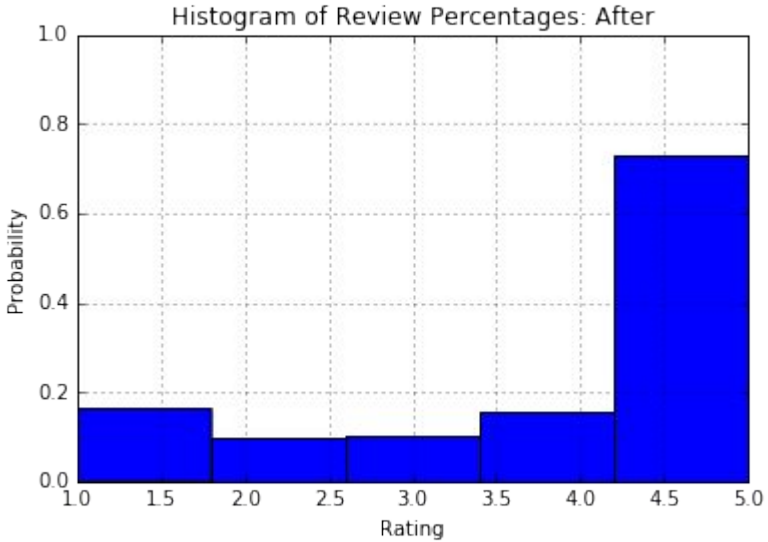
Top 1 Star Review Words

- Fix
  - Analysis: App is buggy
- Update
  - Analysis: App is buggy or the change may have created an unexpected experience for the user
- Needs
  - Analysis: The app does not fulfill the desires of the user
- Crashes
  - Analysis: App is buggy
- Slow
  - Analysis: App is too not performant enough for the standards of the user
- Tried
  - Analysis: App is either buggy or the UX is confusing

Value Analysis

- If the app doesn't crash, isn't buggy, and isn't slow, then there will be about 45% less 1-star reviews on the app.





Next Steps

- Development teams must balance speed of shipping with quality of delivery. In order to balance these, a recommended approach would be to ship very small, focused features often. For each small feature, there should be a lot of integration and unit testing on all supported devices. It is also recommended to make fixing old bugs targeting a reasonable percentage of users a top priority.

https://github.com/RobertErvin