IMO: Sentiment Analysis Visualization on Mobile App Reviews

Jeffrey Xiao and Flora He

Abstract

In this project, we created an interactive visualization given the app data that would allow a market research team to better navigate, visualize, and draw comparisons between applications for mobile devices. In addition, we aim to use advanced tools such as sentiment analysis to construct graphs over time that would better illustrate how an app is performing at a certain time due to what factors. It would also have the ability to display two different apps side-by-side and comparison by the digital distribution platform.

Introduction

In an era where convenience is king, 77% of US adults own a smartphone, which on average has enough computational power to rival that of a midrange desktop from just a few years in the past. Because of this, the mobile app market has grown larger than ever, with 2.1 million apps available on Google Play and 2 million apps on the Apple App Store in 2018. With so much competition, development teams often have trouble ensuring that their apps stand out in today's oversaturated market. The app stores themselves have layouts more tailored to the consumer than the developer, with navigation designed to lead a potential customer to try an application with little information about it beyond a description, rating, pictures, and a couple of reviews. There are plenty of tools available for professional market analysis, but almost all of them feature plain tables and line graphs with relatively bare-bones data.

Background and related work

Thus far, there has been little work done in the field of custom visualizations of app store data. Moreover, most of the similar projects do not provide much direct comparison between applications. Many datasets are limited to one source, free apps only, or out-of-date data. In our visualization, we dynamically query both Android and

iOS app store data and allow the user to compare app performances in multiple categories.

One of the key points in the visualization involves using a sentiment analysis process on the reviews of a mobile application. Some of these papers discusses the method of collecting, parsing, and analyzing the data. In general, “word bags” are used to define the attitudes of the reviews, which is to say each individual word is given a sentiment score based on how it might be interpreted. Many sentiment analysis algorithms are not suited to this task because they exploit indirect indicators of sentiment could instead reflect genre or topic and tend to be more general, causing some fidelity to be lost.

There are a myriad of data visualization tools which focus on to making data accessible and interactively, often in such a way that represents the complex and nuanced human nature behind the raw numbers. In our project, we will aim to borrow some of the philosophy behind their designs as well as some of their visualization techniques.

Data

Our data is taken from the free version of the 42matters third party service, specifically their API. This gives us the advantage of querying dynamically, ensuring that our market data is up to date all of the time. For this visualization, we use data from both the Apple App Store and the Google Play Store. 42matters provides information such as genres, ratings, links to the app icons, a description, number of updates, and other data which has been parsed and formatted neatly. All of this data can be found from the app store itself, but 42matters provides additional Sentiment Analysis for the app’s reviews. Each review may be sorted into a category depending on keywords and phrases found in the reviews, such as “Update” and “too many ads.” These categories are then given a positive sentiment score and a negative sentiment score which sum to 1.0; for example, a category might be 0.6 positive and 0.4 negative.

Design

Design overview

“Overview first, zoom and filter, details on demand.” - Schneiderman's Mantra

- Overview market analysis in different sources (Macro level)

First, we should give the user a broad awareness of the entire data space. The overview usage is not limited to initial reconnaissance; we allow the user to interleave the use of overviews and details views by switching back and forth between them many times.

- Explore per app analysis (Meso level)

At the beginning of the exploration process, we provide a few filter sections to guide the user in choosing where to drill down to inspect in more detail.

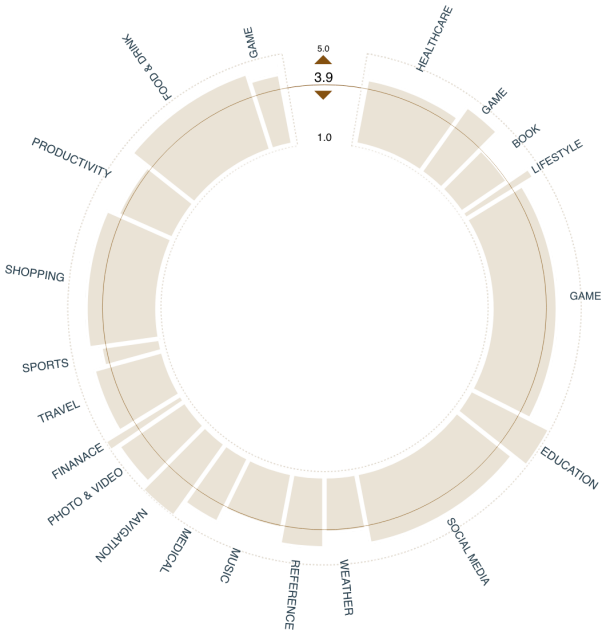
- Compare sentiment analysis in a time period (Micro level)

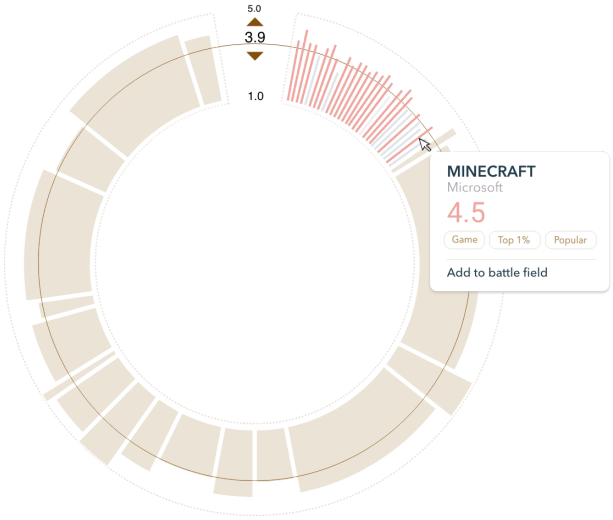
On the side of the page, we create a Battlefield for detailed comparison on demand. In this section, a line chart with more detailed rate change overtime are displayed.

Target users

- Developers who want to gain insight into meaningful app data trends
- Market research teams interested in sentiment analysis
- Project managers who monitor the app performance over time
- Random users interested in the mobile app market

Because of the widely varying audience, we have to strike a balance between a visualization that is pleasant to interact with and view as well as one that is easy to decipher and draw meaningful insights from.





Interactions

A single static view can show only one aspect of our dataset, therefore, we designed an interactively changing display that supports multiple queries, visual feedback, and responsiveness.

Selecting filters and changed them into labels on the top frame;

Using the slider in the middle, the user can be sorting the app above a certain value;

Hovering on the bar to read a list of attributes of that year;

Showing the chosen apps on the sidebar;

Detailed comparison when open the side slides;

Choosing a period of time to compare the performance of two selected apps.

Layout

"Out of sight, out of mind" - John Heywood

The "out of sight, out of mind" mentality about missing information: users tend to forget to take into account elements that have been filtered out, even when their absence is the result of quite recent actions. The disjoint interaction (scrolling) in the vertical layout has a more detrimental effect on the experience than the compromised readability in a horizontal layout. Based on this fact, we chose the horizontal layout in the final design.

Eyes Beat Memory

Using our eyes to switch between different views that are visible simultaneously has a much lower cognitive load than consulting our memory to compare a current view with what was seen before. Arranging the overviews and detail, we designed a sidebar that shows the selected apps and floating filter sections on the right since aggregation can be somewhat safer from a cognitive point of view.

Color

"Get It Right in Black and White" - Maureen Stone

This quote from Maureen Stone is sums up one of the most important design guidelines for effective use of color. Simply put, we must ensure the most crucial aspects of visual representation are legible even if the image is transformed from full color to black and white. The color scheme we utilized past this challenge.

We chose mono-color on the different app category and emphasized the central sentiment network diagram. The two representative colors, orange, and purple, are the strongest comparison color set, which distance 180-degree on the color wheel.

Code

Due to the multifaceted nature of this project, there are many different components working in tandem. I will be discussing the source code based linearly in sections, based on how the code is rendered. The three main components are the dynamic selection and loading of data sets, the outermost radial bar chart, and the inner chord diagram of review sentiment analysis.

Because the app store shifts rapidly, we made the decision to obtain datasets dynamically using a third-party service. 42matters provides an API that allows an end user to obtain data from Android and iOS applications. For our purposes, we obtain the top charts from the Google Play Store and Apple App Store. The code before the function “loadNewData” renders buttons that allows the user to switch between these two data sets. When one is clicked, it turns a shade of light brown to indicate that it is the currently selected data set. The loadNewData function takes the given source and uses it as a json object to begin generating the other graphs. The data is sorted differently between iOS and Android, but the gist is that the list of applications is sorted first by category, where each application is put into a different array of the same primary genre; then, after each app is put into a genre array, they arrays themselves are sorted by reviews from lowest to highest. Once this is completed, our data is ready to be visualized.

The radial bar chart displays each app as a bar with the lower bound being 2 stars and the upper bar being 5 stars. The verticality of the bar is given by the variable *y* while the position along the circle is given by *x*, mapped by the number of applications in the visualization. To actually create this bar chart, we use the *d3.arc()* function. When initializing the bar chart, the *startArc* variable is used. This initializes all of the bars at the *innerRadius* position so we can apply a sequential *d3.transition()* with a delay between each bar's start of transition, creating a smooth animation where the bars pop up one by one around the circle in a wave-like fashion. The final position is determined by the *arc* variable, which creates a gap between given by the *isGap* data value, separating bars in different categories. Each different category is also given a label, which fade in sequentially along with the bars. If the user wishes only to view apps over a certain threshold, a filter circle (given by the variables *ratingCircle, ratingLabel, and ratingRhombus*) is created using *d3.drag()*, which changes the fill of bars below a certain threshold to grey instead of pink. Hovering the mouse over a bar will display a tooltip that displays the app name, its publisher, its rating, and its primary category. It also triggers another transition with grows that bar out, showing the user which bar they have selected. In addition, it calls the *hoverArc* and *otherArc* variables, which remove the gap between and increases the opacity of all apps' bars except those belonging to the category that the user is hovering over. Clicking on this bar allows the user to select that app to display in the center, calling the *update* function to render these center elements.

At the center of the visualization, a labeled chord diagram displays categories which 42matters has grouped a review under. Each topic has a positive and negative score attributed to it, based on whether the reviews making up that topic were perceived to be positive or negative; these values sum to 1.0. The topic also has a percentage contribution to the overall sentiment analysis score based on how many reviews involved that topic, though the contribution scores do not necessarily sum to 1.0 and we must account for that in the range of our chord. The *mapNextValues* function performs the angle calculations for each topic, given by *prevAnglePos* and *prevAngleNeg* for positive sentiment and negative sentiment angle values, respectively. An additional value is added in the return array of *mapNextValues* because for some reason, a bug with the code causes d3 to sometimes skip over the first data value. Further analysis

should be done in determining whether it is the d3 library or our code which is causing this issue. Once this data is mapped, the positiveChordArc and negativeChordArc are drawn. The angle of each segment is given by the previous angle calculation for the start angle and the current angle calculation for the end angle. These values are inverted about the y-axis though because due to the nature of the diagram, this orientation causes the ribbons in the chord diagram to cross over each other less and makes it easier to distinguish which ribbon belongs to which category. *d3.ribbon()* is a function that generates ribbons based on a source and destination both using the same start and end angle function; however, since our chord diagram is slightly different from the norm because the source and target destinations have different calculations for the start and end angles, we had to specify a value *beginningOrEng* to determine which angle calculation to use. For our source, denoted by *"beginning,"* we simply return a start and end angle that has few differences with a linear and equidistant distribution. For our target, our start and end angles are the same as those used to construct the chord arc. At the source, we also place circles with either a orange or purple fill depending on whether that topic contributed more towards the positive or negative reviews with a label of that topic name in the same color. Hovering over these will cause the other of the topics' ribbon and arc to fade such that hovered topic's sentiment contributions can be seen. It also shows a label of which topic has been selected and its positive and negative contributions to the sentiment score.

Future Evaluation

After deploying the visualization, we are going to conduct usability tests to our target users. The user should be able to navigate through visualization as quickly as they could through a mobile application distribution platform with the same if not more grouping and filtering features. They should also be able to gain insight beyond that of the application distribution platform.

IMO, as an exploratory system, is inappropriate to define success on just one side. We intended to measure the expressiveness of the system on two aspects: User performance and User experience.

Learnability, Effectiveness, and Insightfulness are three metrics we selected to evaluate the usability of the visualization. And for the subjective User experience, we choose SUS scores, Satisfaction, and Future use willingness as key measurements.

Reflections