Patent Categorization and Analysis using NLP

This report documents the methodology and results of my final capstone project for Udacity Data Scientist Nanodegree. I have been working in the Intellectual Property domain for 12 years now. One of the common tasks that has to be performed in the domain is patent portfolio evaluation. First step towards evaluating a patent portfolio is to categorize the patents into various buckets, then perform analytics on the categorized portfolio. The process can take anywhere between a week to a month depending on portfolio and team size. I believe, utilizing my knowledge gained in the Data Science nanodegree, I can substantially reduce the effort by 60 - 70% and also increase the quality of a patent categorization process.

Project Definition

Project Overview

A patent is a form of intellectual property that gives its owner the legal right to exclude others from making, using, or selling an invention for a limited number of years in exchange for publishing a public disclosure of the invention. A set of all patents owned by an individual or company forms a patent portfolio. In order to ascertain the monetary value of the portfolio, we need to get an idea which products or markets areas the patent portfolio covers. Currently most Intellectual Property (IP) attorneys and consultants perform this task manually - going through hundreds and thousands of assets to complete categorization.

There are some Machine Learning based solutions available (e.g. Innography) that automatically categorizes assets. However, the whole process is like a black box, thus most IP attorneys and consultants don't feel comfortable using those solutions.

Problem Statement

I would like to achieve the following through this project:

1. Provide a patent categorization solution using NLP.
2. Explain the NLP based categorization solution by engaging and incorporating user input throughout the process, so it doesn't feel like a black box.
3. Portfolio comparison.

Metrics

As we will see further in the report, I will be using an unsupervised clustering model. The evaluation of unsupervised learning is difficult as there is no goal model to compare with. We can use some metrics to compare two clustering algorithms and to determine the number of clusters to form.

For comparing two clustering algorithms, I will be using - Silhouette index
For determining the number of clusters to form in a dataset - I will be using rate of change in distortion.

Data Collection

I collected the data for the project from the following sources:

1. Google Patents - Link
2. Google Patents Research Data - Link
3. Patents View Bulk Download - Link
4. Patent Grant Bibliographic Text - Link
5. Patent Application Bibliographic Text - Link

To keep the dataset manageable, I downloaded data for only three assignees - Facebook, Google, and Microsoft - for patents filed in or after 2015 in the United States. This came to a total of 21,295 patents (Facebook: 3,332; Google: 7,130; Microsoft:10,833).

Analysis

Data Exploration

The bibliographic information about a patent that can be used to categorize a portfolio are

- CPC classification codes,
- Title
- Abstract

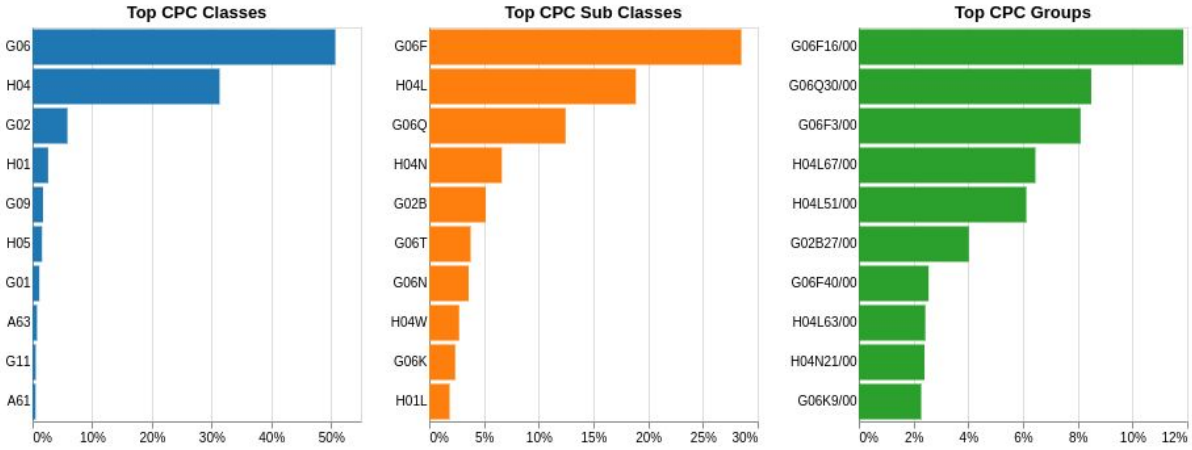
The Cooperative Patent Classification (CPC) is a patent classification system, which has been jointly developed by the European Patent Office and the United States Patent and Trademark Office. It is a hierarchical categorization system and has over 250,000 categories! Patents are assigned at least one classification code indicating the subject to which the invention relates to. Additional classification may be further assigned to cover more details of the contents.

The problem with using CPC classification for portfolio categorization is:

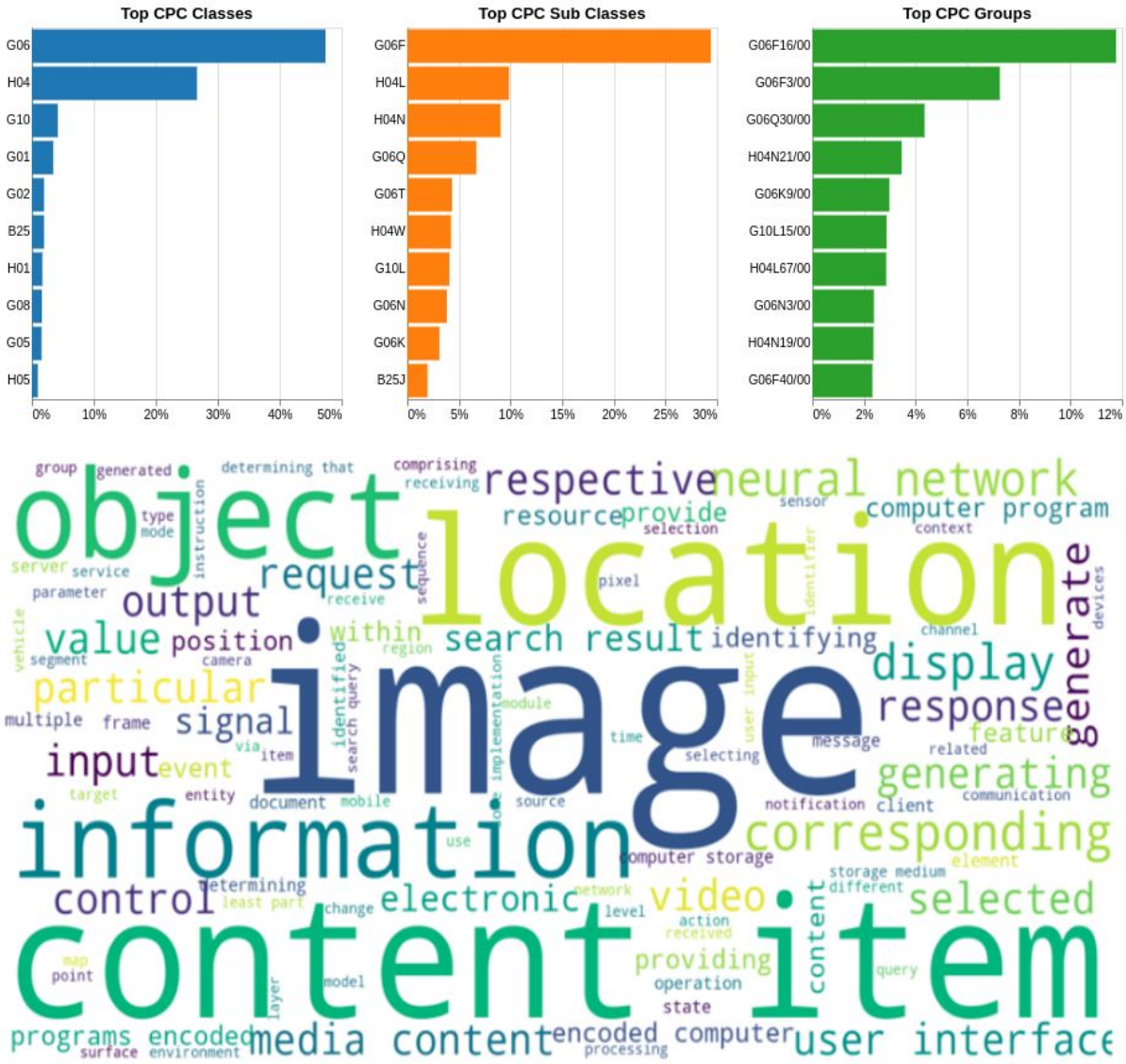
a. If we use CPC codes higher up in the hierarchy - most patents of a portfolio will be covered by two or three codes.
b. If we use CPC codes lower in the hierarchy, we will end up with too many categories.

Data Visualization

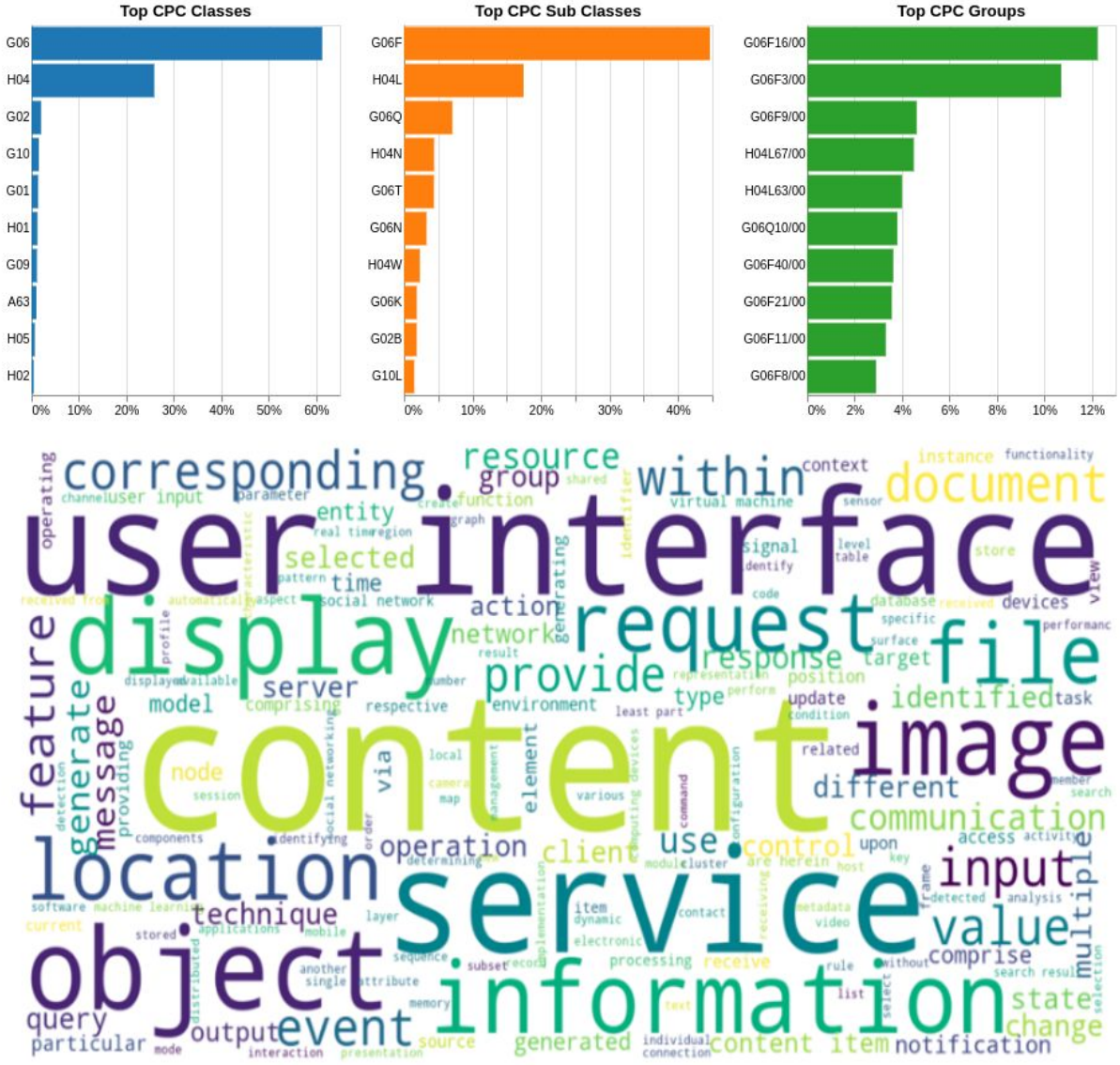
Before we start the categorization process, let's look at some visualizations.
Figures below show the top CPC classes, sub-classes, and groups, along with a word cloud for based on title and abstract text of patents in three portfolios - Facebook, Google, Microsoft.



Top CPC codes and Word Cloud for Facebook



Top CPC codes and Word Cloud for Google



Top CPC codes and Word Cloud for Microsoft

Methodology

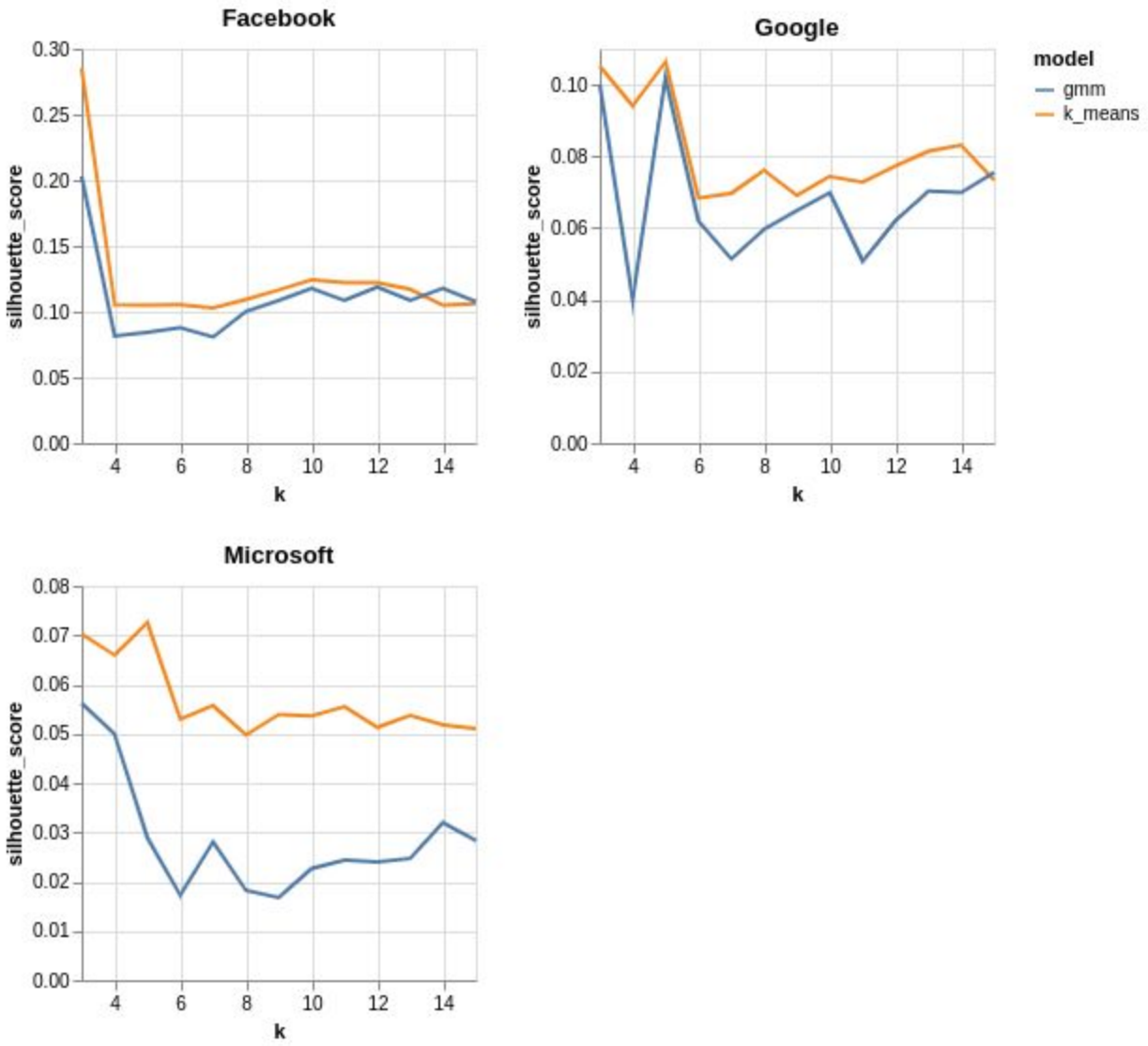
Data Processing

Following steps were performed in attached Jupyter notebooks to prepare data for modeling:

1. Bibliographic data collected for different assignees from Google Patents was collated into one CSV file
2. Word embeddings data collected from Google Patent Research data was collated into one CSV file.
3. CPC data collectected from Google Patent Research data was collated and added to the Bibliographic data.
4. The XML data downloaded from Patent Grant Bibliographic Text and Patent Application Bibliographic Text was processed and abstract for all patents in Bibliographic CSV file was extracted and saved as CSV file.

KMeans or Gaussian Mixture Model?

I compared the Silhouette score between KMeans and the Gaussian Mixture Model.
The silhouette score is a measure of how similar a data point is to its own cluster (cohesion) compared to other clusters (separation). The silhouette score ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.
Based on the comparison I decided to use the k-Means model.



Comparison between silhouette score for k-Means and GMM

Grid Search for KMeans

I performed a grid search for different values of k-Means input parameters - init, n_init, max_iter, and tol.

init: Method for initialization

> **k-means++**: selects initial cluster centers for k-mean clustering in a smart way to speed up convergence.
>
> **random**: choose n_clusters observations (rows) at random from data for the initial centroids.

n_init: Number of time the k-means algorithm will be run with different centroid seeds.

max_iter: Maximum number of iterations of the k-means algorithm for a single run.

tol: Relative tolerance with regards to Frobenius norm of the difference in the cluster centers of two consecutive iterations to declare convergence.

As observed, changing these parameters didn't have any significant impact on distortion.

					k_4	k_5	k_6	k_7	k_8	k_9	k_10	k_11
init	algorithm	tol	max_iter	n_init								
k-means++	auto	0.0001	200	5	903.245774	868.260063	837.470118	808.365126	785.323616	763.831909	744.297658	731.236331
				10	903.245774	868.260063	837.470118	808.360297	785.323616	763.831909	744.297658	731.236331
			300	5	903.245774	868.260063	837.470118	808.365126	785.323616	763.831909	744.297658	731.236331
				10	903.245774	868.260063	837.470118	808.360297	785.323616	763.831909	744.297658	731.236331
		0.0010	200	5	903.246375	868.265745	837.474442	808.365126	785.324569	763.831909	744.297658	731.236331
				10	903.246375	868.265745	837.474442	808.361524	785.324569	763.831909	744.297658	731.236331
			300	5	903.246375	868.265745	837.474442	808.365126	785.324569	763.831909	744.297658	731.236331
				10	903.246375	868.265745	837.474442	808.361524	785.324569	763.831909	744.297658	731.236331
	full	0.0001	200	5	903.245774	868.260063	837.470118	808.365126	785.323616	763.831909	744.297658	731.236331
				10	903.245774	868.260063	837.470118	808.360297	785.323616	763.831909	744.297658	731.236331
			300	5	903.245774	868.260063	837.470118	808.365126	785.323616	763.831909	744.297658	731.236331
				10	903.245774	868.260063	837.470118	808.360297	785.323616	763.831909	744.297658	731.236331
		0.0010	200	5	903.246375	868.265745	837.474442	808.365126	785.324569	763.831909	744.297658	731.236331
				10	903.246375	868.265745	837.474442	808.361524	785.324569	763.831909	744.297658	731.236331
			300	5	903.246375	868.265745	837.474442	808.365126	785.324569	763.831909	744.297658	731.236331
				10	903.246375	868.265745	837.474442	808.361524	785.324569	763.831909	744.297658	731.236331
random	auto	0.0001	200	5	903.246579	868.258639	837.499106	809.633035	785.581773	763.773095	745.031547	731.005648
				10	903.246579	868.258639	837.476413	809.633035	785.581773	763.773095	744.961100	731.005648
			300	5	903.246579	868.258639	837.499106	809.633035	785.581773	763.773095	745.031547	731.005648
				10	903.246579	868.258639	837.476413	809.633035	785.581773	763.773095	744.961100	731.005648
		0.0010	200	5	903.251390	868.264052	837.499920	809.633035	785.581773	763.779350	745.031547	731.005648
				10	903.251390	868.264052	837.476413	809.633035	785.581773	763.779350	744.961100	731.005648
			300	5	903.251390	868.264052	837.499920	809.633035	785.581773	763.779350	745.031547	731.005648
				10	903.251390	868.264052	837.476413	809.633035	785.581773	763.779350	744.961100	731.005648
	full	0.0001	200	5	903.246579	868.258639	837.499106	809.633035	785.581773	763.773095	745.031547	731.005648
				10	903.246579	868.258639	837.476413	809.633035	785.581773	763.773095	744.961100	731.005648
			300	5	903.246579	868.258639	837.499106	809.633035	785.581773	763.773095	745.031547	731.005648
				10	903.246579	868.258639	837.476413	809.633035	785.581773	763.773095	744.961100	731.005648
		0.0010	200	5	903.251390	868.264052	837.499920	809.633035	785.581773	763.779350	745.031547	731.005648
				10	903.251390	868.264052	837.476413	809.633035	785.581773	763.779350	744.961100	731.005648
			300	5	903.251390	868.264052	837.499920	809.633035	785.581773	763.779350	745.031547	731.005648
				10	903.251390	868.264052	837.476413	809.633035	785.581773	763.779350	744.961100	731.005648

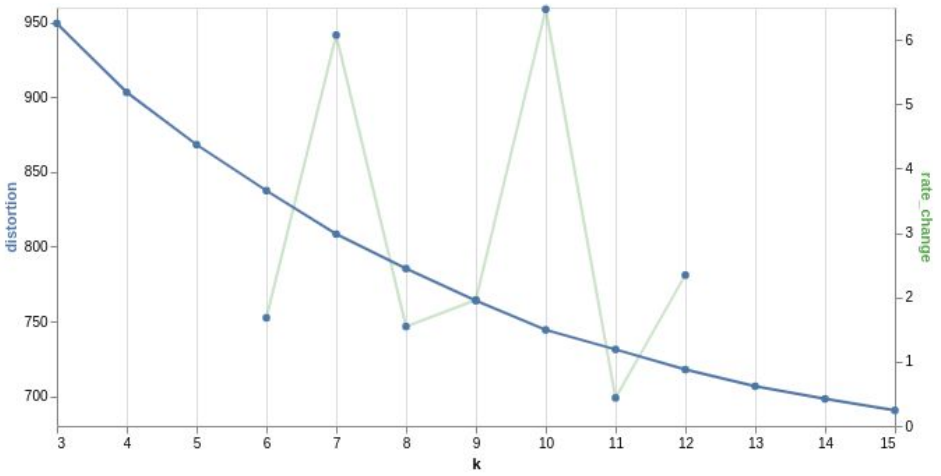
Distortion for different k (number of clusters) for different values of init, algorithm, tol, max_iter, n_init

Modelling

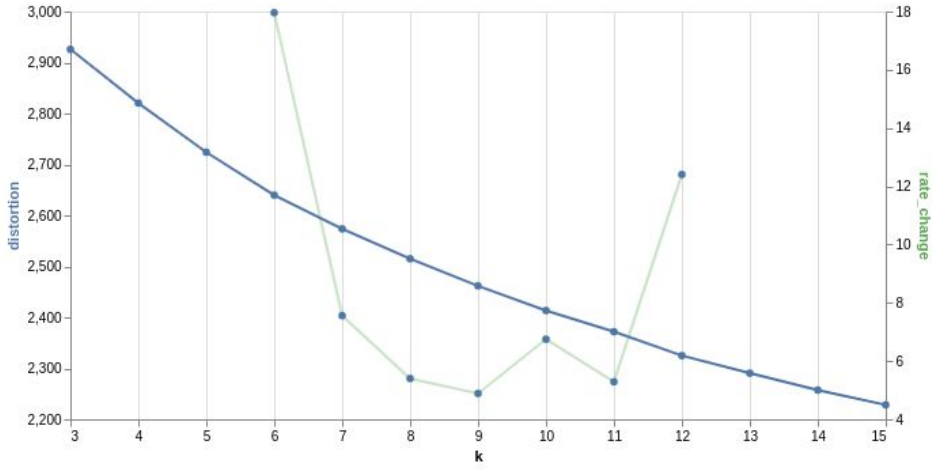
Going forward, I will include analysis for only Facebook and Google portfolio for simplicity.

Step 1: Determine number of cluster to form

Rate of change in distortion is an indicator which can be used to determine the number of clusters to categorize into.



Distortion and rate of change of distortion for Facebook portfolio

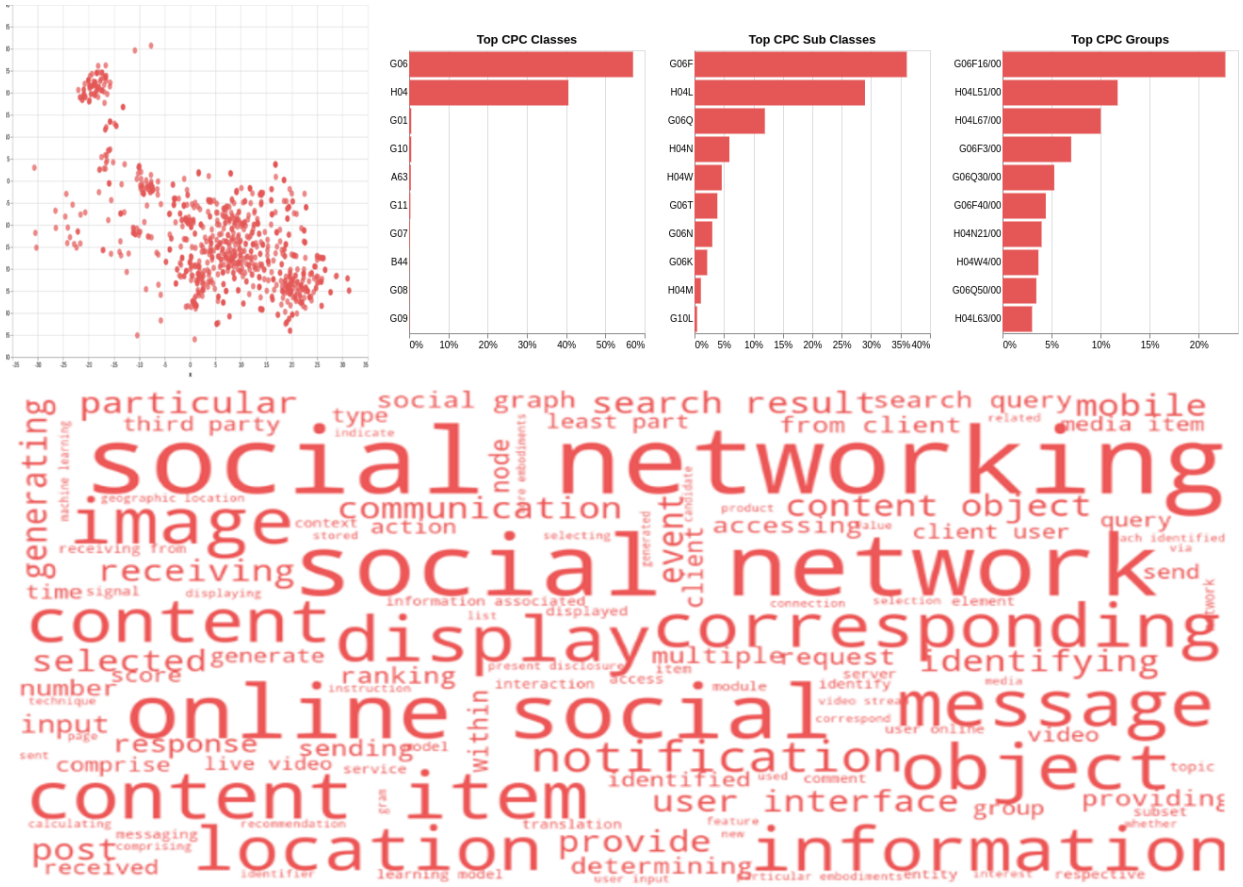


Distortion and rate of change of distortion for Google portfolio

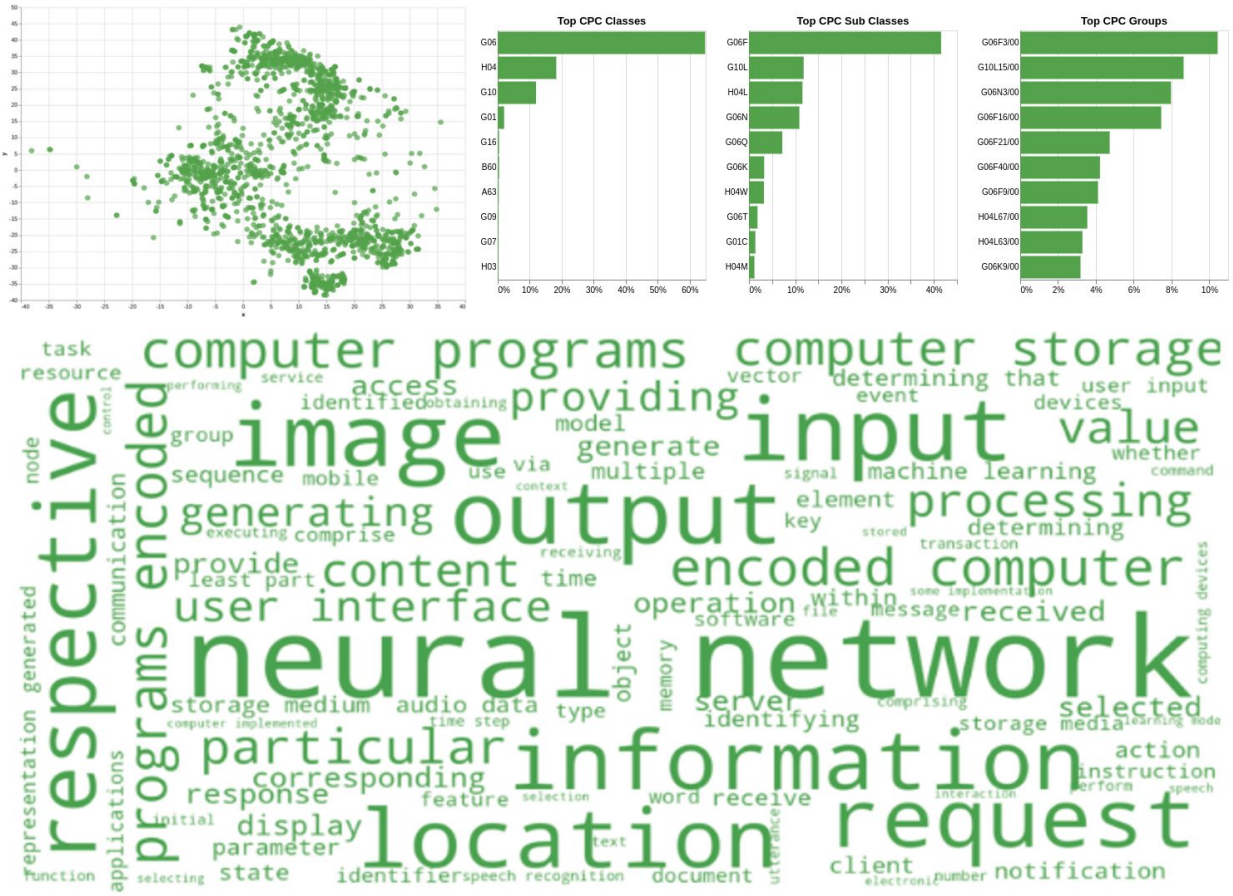
Looking at the rate of change of distortion - it makes sense to cluster Facebook portfolio in 7 clusters and Google portfolio in 6 clusters.

Step 2: Categorize

Categorize the portfolio into n clusters as determined from the previous section. For each cluster I also showed top CPC classes, sub classes and groups, and word cloud. This will help in deciding the name for a cluster. To keep the report concise,, I am showing only one cluster for Facebook and one for Google.



One of the Facebook clusters along with top CPC codes and Word Cloud for the cluster



One of the Google clusters along with top CPC codes and Word Cloud for the cluster

After looking at the top CPC codes and word clouds for the clusters, I decided with the following names for the clusters.

Facebook:

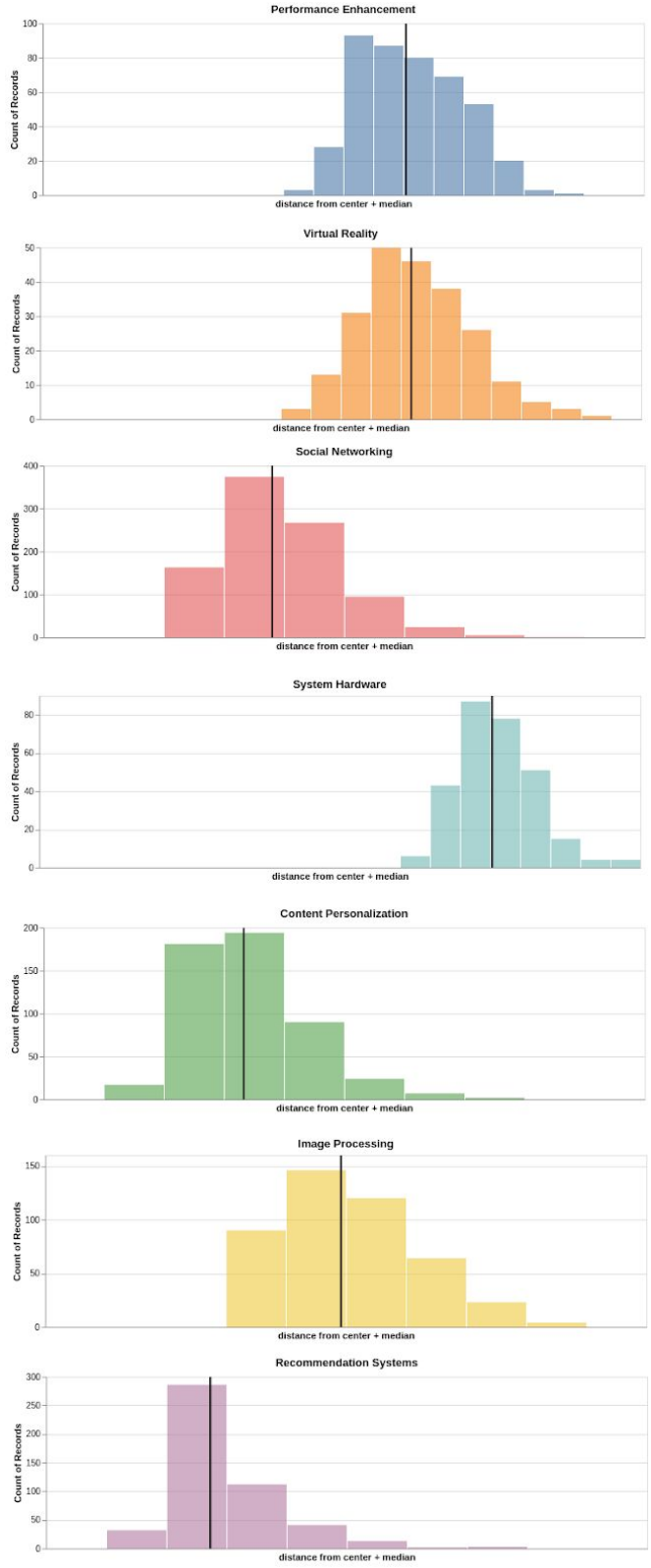
1. Content Personalization
2. Performance Enhancement
3. System Hardware
4. Social Networking
5. Image Processing
6. Recommendation Systems
7. Virtual Reality

Google:

1. Autonomous Driving
2. Home Improvement
3. Search and Content Personalization
4. Image Processing
5. Wireless Communications
6. Machine Learning

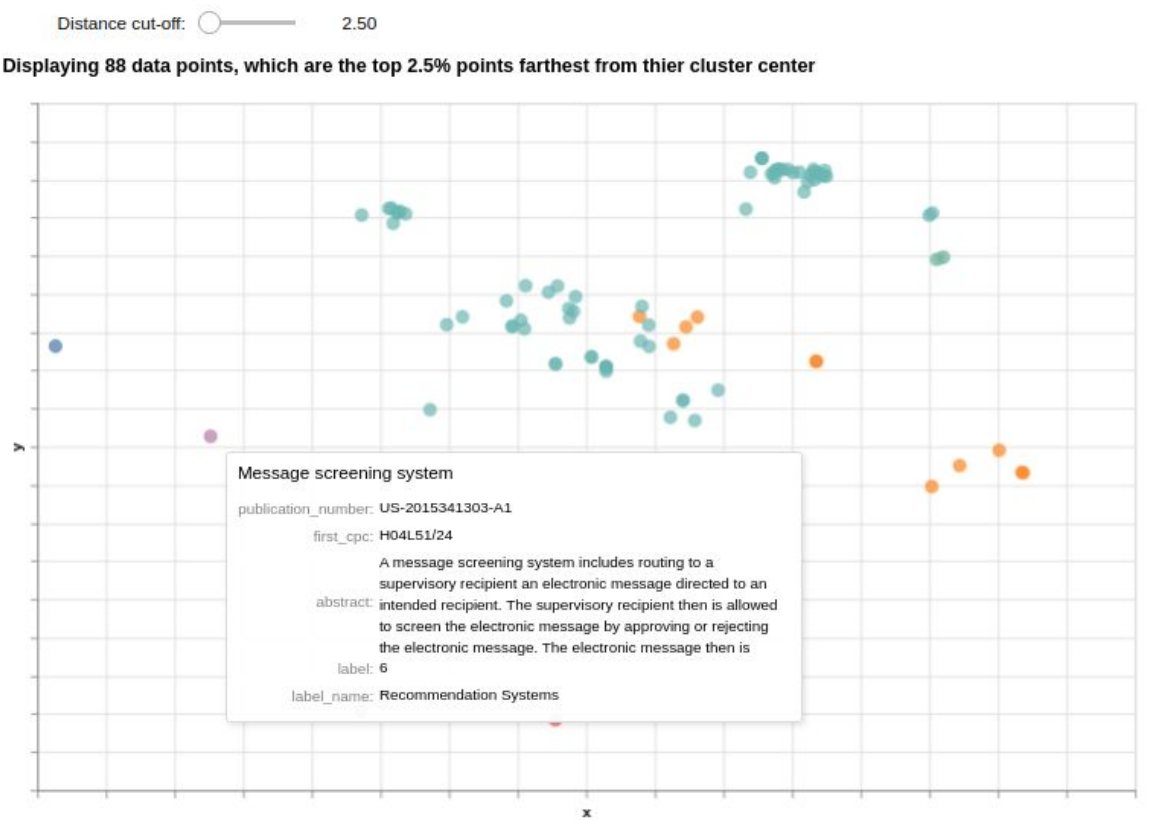
Step 3: Review

In this step, I show the distribution of distances of data points from the cluster center. This will help in ascertaining how tightly packed the clusters are. The closer the distances are zero (left), the tightly packed th clusters are. For Facebook, it appears the cluster for ‘System Hardware’ is not very tightly packed and would require more manual review to increase the quality of clustering.



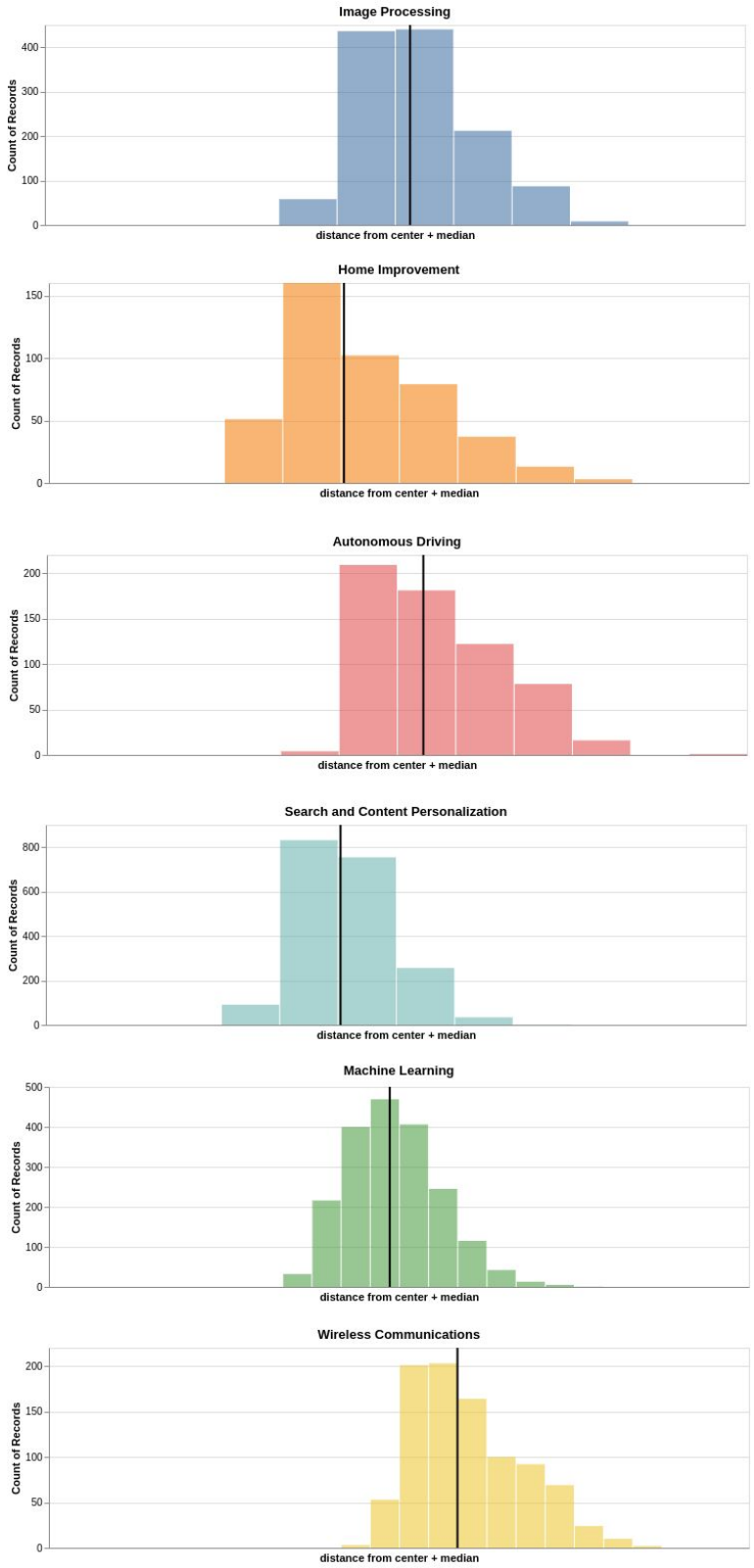
Distribution of distances from cluster center for all clusters of Facebook

I also added a slider to filter data points based on distance from the cluster center. For example in the image below, the slider is set at 2.5 showing 88 data points for Facebook that are 2.5% of the points farthest from cluster centers.

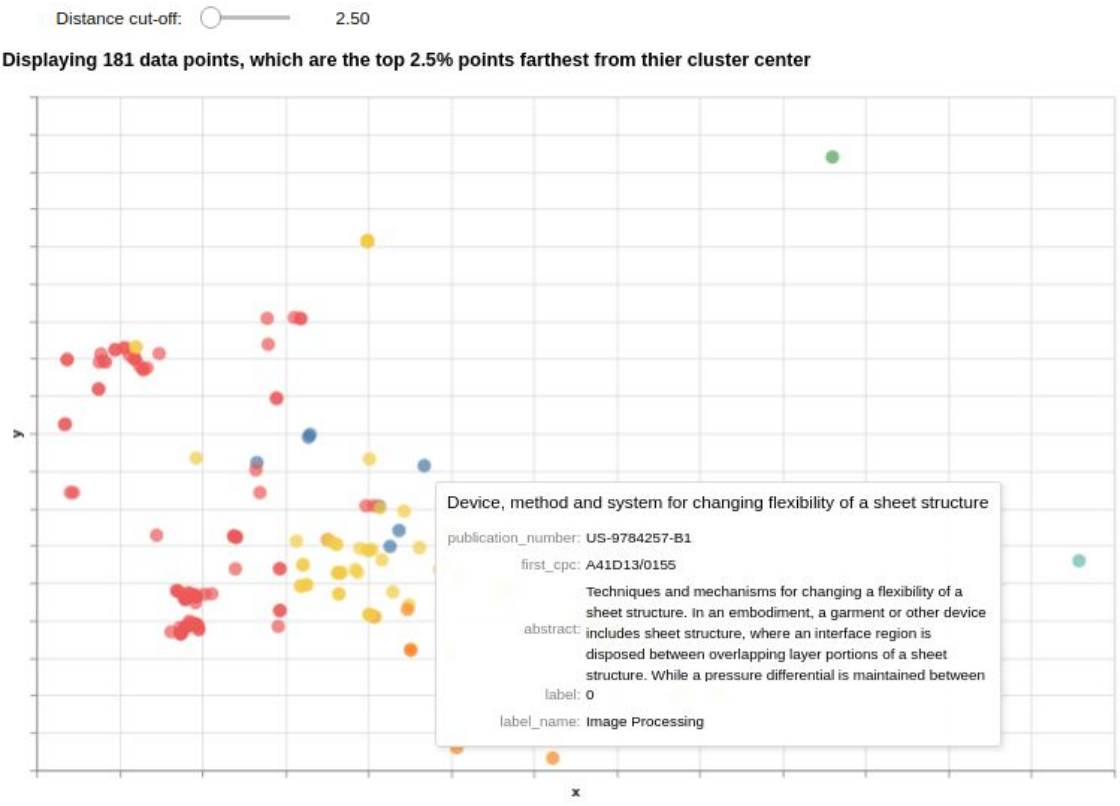


2.5% of the Facebook patents that are farthest from cluster centers

Similarly, for Google Wireless Communications is one cluster that needs to be reviewed further.



Distribution of distances from cluster center for all clusters of Google



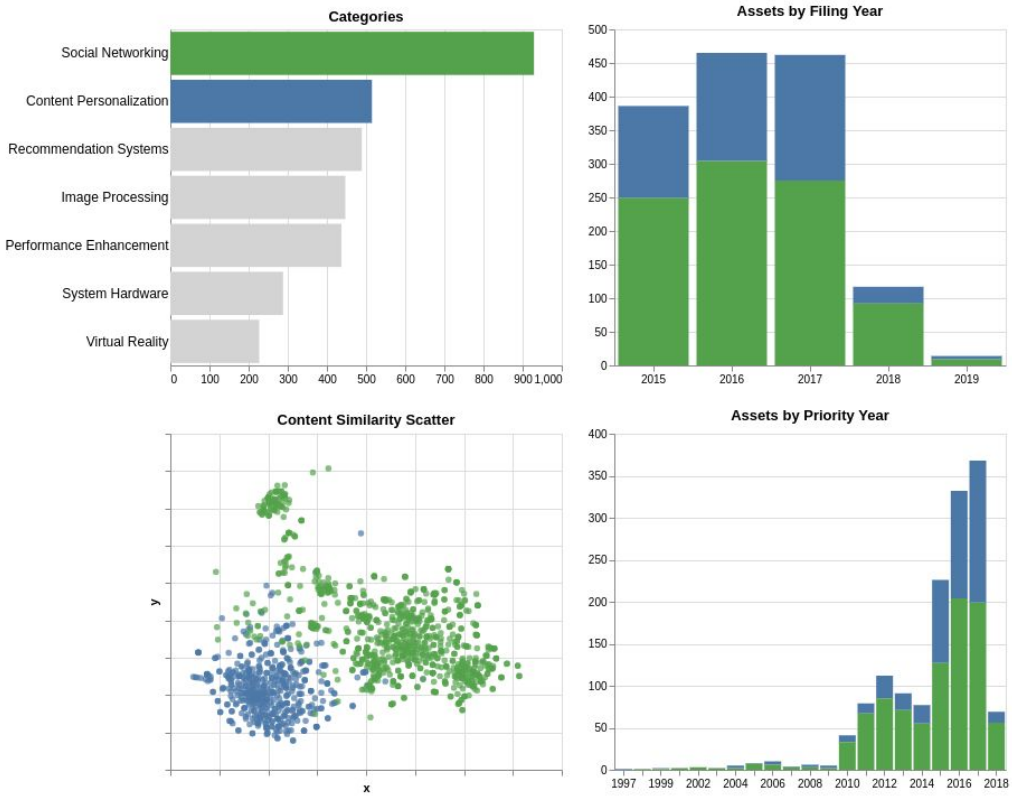
2.5% of the Google patents that are farthest from cluster centers

As can be seen, users can hover over a data point to see more information which would help them in deciding if the category is fine or needs to be changed. Further, users can Ctrl + click on the point to open the patent in Google patents.
After reviewing the patents users can manually change categories for some patents, if required.

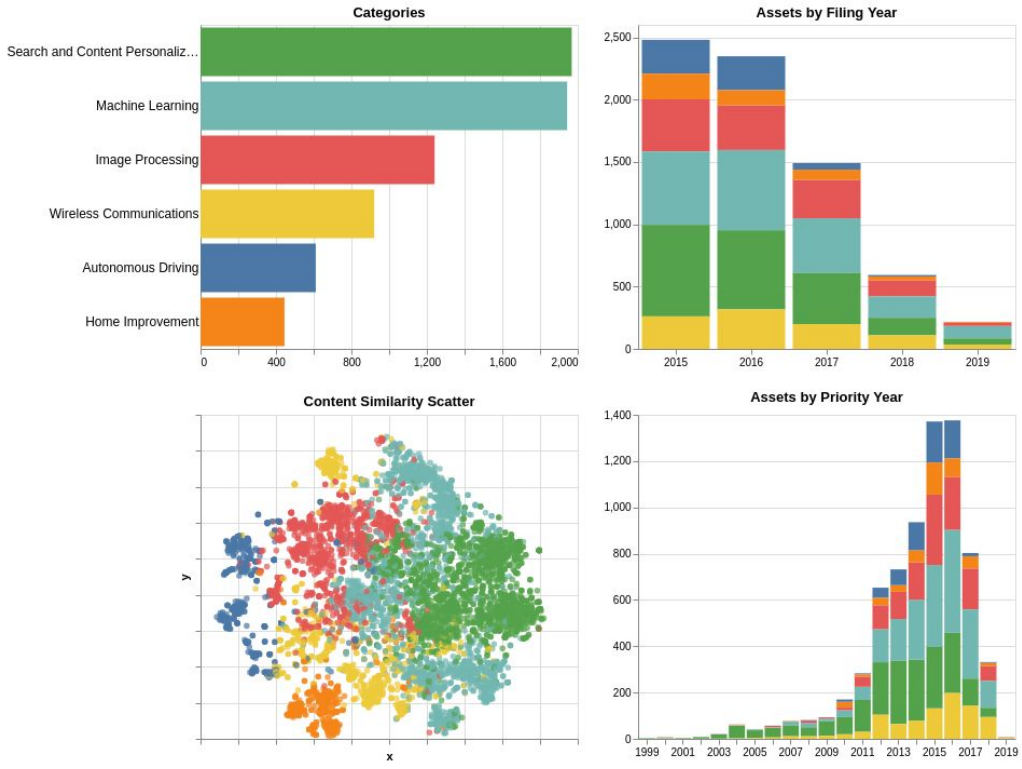
Step 4: Basic Analysis

Once we have categories finalized, we can present the information about the portfolio in a visual form. In the charts below - clicking on one or more bars on the left, filters the data in the other three charts, thus allowing a user to get a more granular understanding of the portfolio.

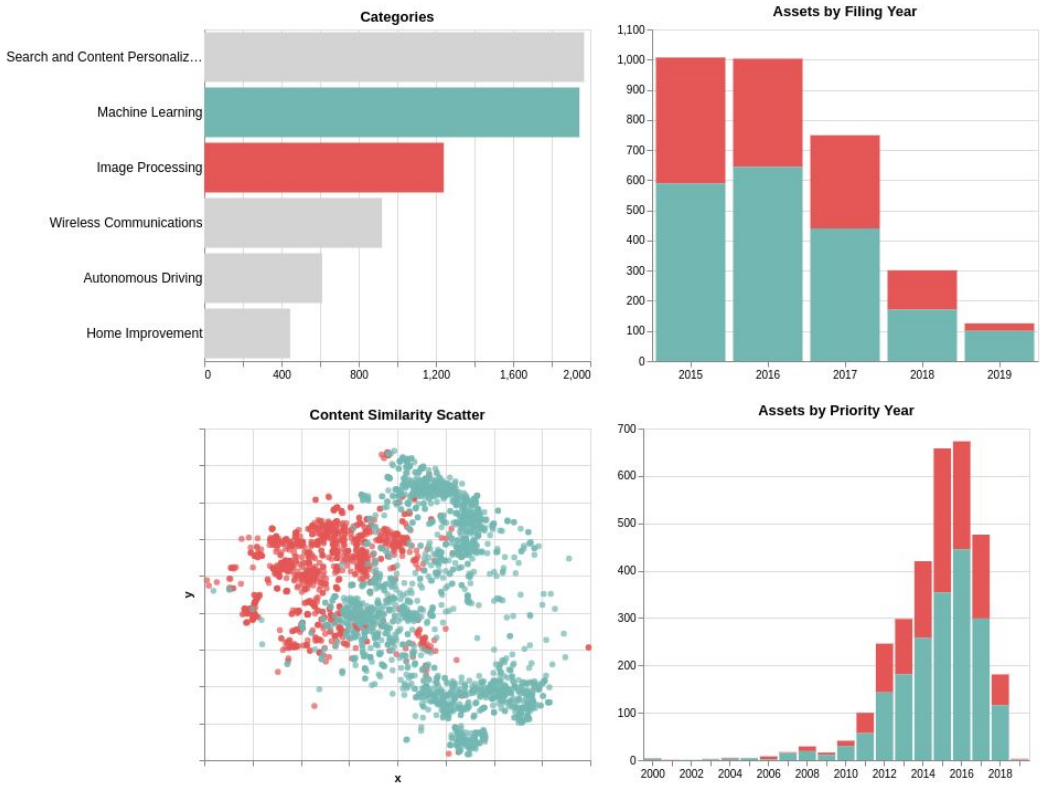
Snapshot of Facebook Portfolio after categorization



Snapshot of Facebook Portfolio after categorization with selection



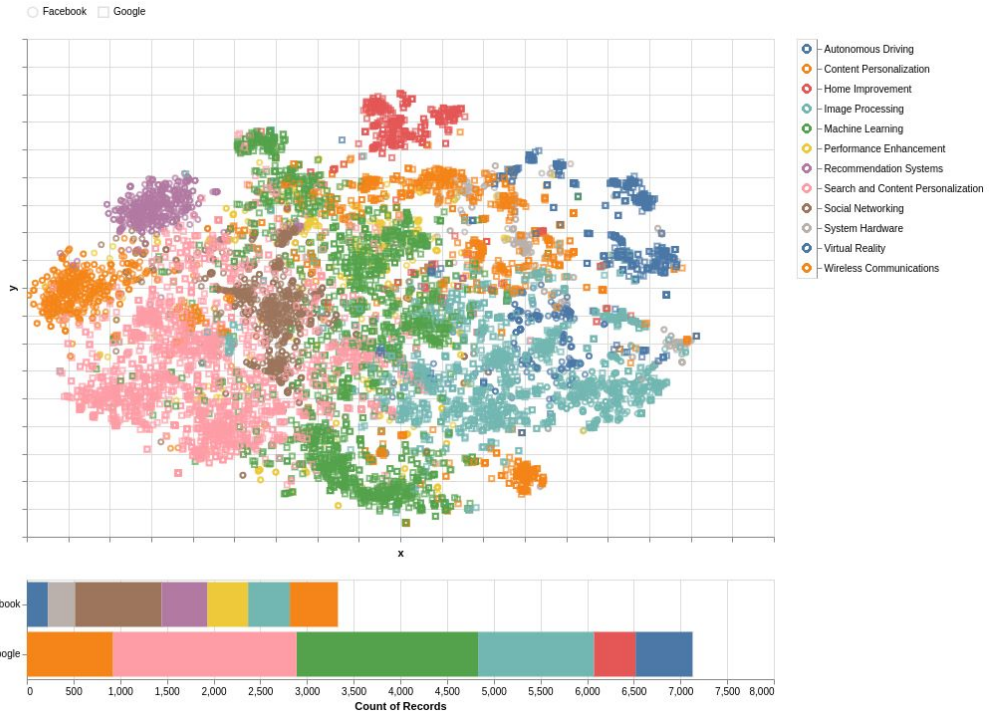
Snapshot of Google Portfolio after categorization



Snapshot of Google Portfolio after categorization with selection

Step 5: Advanced Analytics

Once we have gained some understanding on individual patent portfolios, we can compare two or more portfolios together. We can see how similar or dissimilar two categories in different portfolios are. Such a comparison would not be possible in traditional manual categorization of patents.



Visual Comparison of Facebook and Google patent portfolios

Visual Comparison of Facebook and Google patent portfolios with selection

Not only visually, we can get similarity between patents as numbers by taking dot product of the vectors representing the patents.

	patent_1	title_1	patent_2	title_2	similarity
440406	US-2019313087-A1	Pupil swim corrected lens for head mounted dis...	US-2019271844-A1	Lightguide optical combiner for head wearable ...	0.927011
439548	US-2019313087-A1	Pupil swim corrected lens for head mounted dis...	US-9851565-B1	Increasing effective eyebox size of an HMD	0.923403
440553	US-2019313087-A1	Pupil swim corrected lens for head mounted dis...	US-2020041798-A1	Head wearable display using powerless optical ...	0.920947
389083	US-10529117-B2	Systems and methods for rendering optical dist...	US-2018061119-A1	Quadrangulated layered depth images	0.918013
392806	US-2019318528-A1	Computer-Graphics Based on Hierarchical Ray Ca...	US-2018061119-A1	Quadrangulated layered depth images	0.917839
360817	US-2018239145-A1	Focus adjusting multiplanar head mounted display	US-10241329-B2	Varifocal aberration compensation for near-eye...	0.917539
391565	US-2019318530-A1	Systems and Methods for Reducing Rendering Lat...	US-2018061119-A1	Quadrangulated layered depth images	0.916321
386601	US-2019318529-A1	Systems and Methods for Rendering Foveated Eff...	US-2018061119-A1	Quadrangulated layered depth images	0.915904
180872	US-2017262054-A1	Focus adjusting headset	US-10241329-B2	Varifocal aberration compensation for near-eye...	0.911147
514008	US-2020081252-A1	Polarization-sensitive components in optical s...	US-9851565-B1	Increasing effective eyebox size of an HMD	0.910840

Top 10 most similar Facebook and Google patent in Image Processing category

References

- Clustering - Link
- In Depth: Gaussian Mixture Models - Link
- sklearn.cluster.KMeans - Link
- sklearn.manifold.TSNE - Link
- The General Ideas of Word Embeddings - Link
- Generating WordClouds in Python - Link
- Altair: Declarative Visualization in Python - Link
- Bindings, Selections, Conditions: Making Charts Interactive - Link
- Exploratory Data Visualisation with Altair - Link