## Problem Statement:

Build a recommender system to find similar podcasts on iTunes. Currently, users have to sift through data on iTunes to attempt to find similar podcasts.

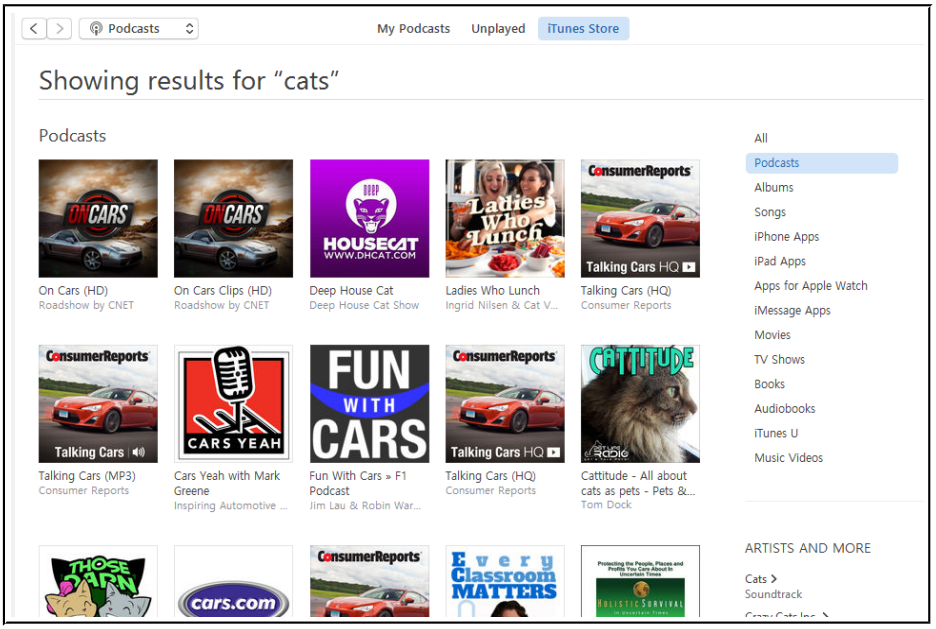
## Understanding the Problem:

Before I can attempt to solve the problem, I need to experience it (I need to identify the pain points).

- **Options for finding a podcast**
  - can browse through the featured selections from iTunes' editors
  - can explore the episodes and podcasts that are topping the iTune's charts
  - can search for a podcast by title or subject
- **Options for finding a podcast similar to the one you are listening to**
  - can view related tab in iTunes, shows you two lists:
    - ***Listeners also subscribed to***
    - ***Top podcasts in the current podcast category***

Now, three problems are apparent.

1. "Listeners also subscribed to list" doesn't guarantee a similar podcast, only podcasts also listened to by other listeners of the podcast. It may or may not give a podcast that is similar (meaning it may give a podcast that others just like, but are not related by genre, bias, style, etc.)
2. "A top podcast in a category" may not really be very similar (i.e. you may be listening to a podcast about financial markets, but a top podcast in the business genre may be a podcast about managing a team)
3. The itunes podcast search is terrible. Example of a search result for 'cats' is below. (I would expect podcasts about cat lovers or pet owners, instead…)




These three problems combined lead to iTunes having no real **similar podcast tool**. Below, is an outline of how to create such a tool.

## Creating a Similar Podcast Tool:

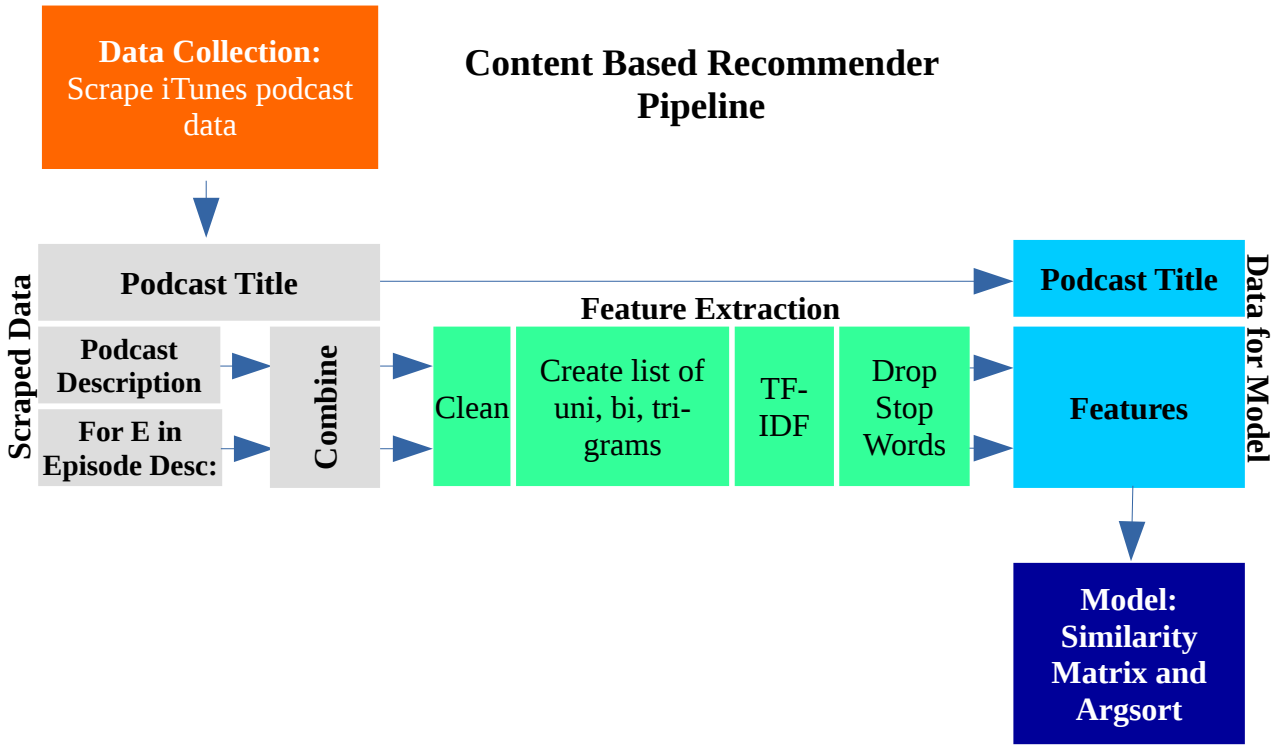
To create a tool, I have to first understand the data available.

**Data:**

- **iTunes Podcast Webpage Data:** podcast title, author, podcast description, category, language, customer rating and number of ratings, link to podcast website, customer reviews (appears to be only three through website, more in iTunes app), other podcasts listeners have subscribed to, podcast episode names, podcast episode descriptions, release date of episode
- **Podcast websites,** may or may not have one
- **Podcast Twitter accounts**, some podcasts have a twitter account soley for the podcast, others may just have a twitter account for themselves

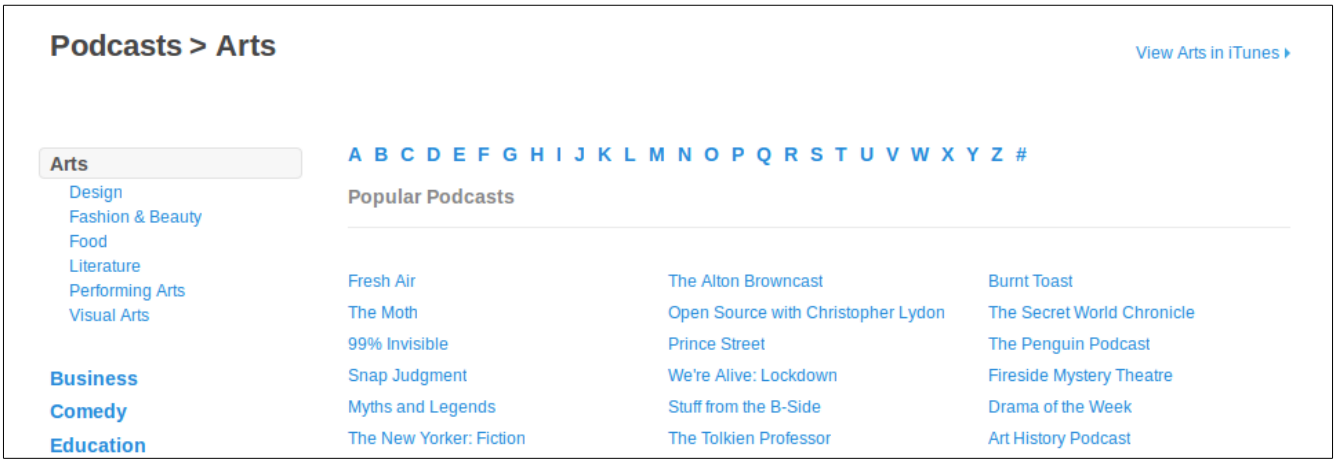
After seeing the data available, we reach our first decision on building a **similar podcast recommender**. The goal is to build a minimal viable product first, then add improvements if needed. With that said, we still want to think about additions and improvements early on, so we can develop a pipeline that makes these additions easier later. The mvp in this case is a **content based recommendation engine**. For this tool, we will only use a podcast's description, title, and epsiodes' descriptions to build a **similar podcast tool**.

## Content Based Recommender Pipeline:



## Data Collection:

**Tools/Libraries:** Selenium, SQL




**Process:**

1. The Podcast section of the iTunes website can be navigated by genre. And within each genre, you can view all podcasts by alphabetical and numerical order (see picture above). Because of this, we can collect all podcast's web page data.
    1. The process to collect the data is to build a python script that leverages selenium in a headless browser.
    2. The web automation tool would loop through all genres, letters/numbers, sub pages for letters, and visit every podcast's iTune webpage.
    3. From every web page, we scrape: podcast title, author, podcast description, category, language, customer rating and number of ratings, link to podcast website, customer reviews , other podcasts listeners have subscribed to, podcast episode names, podcast episode descriptions, release date of episode. The data that has multiple occurrences will be stored in a dictionary or array. *Example, a podcast with multiple episode descriptions will have a vector like: ['episode n description', 'episode n+1 description',...]*
2. Then each podcast's web page data will be added to a SQL database where the podcast's title will be the primary key. Note: there is the possibility that some podcasts may be cross listed, but the SQL data base will be created to only accept unique primary keys (so no duplications will be added).

Aside on data size: iTunes has 250,000 unique podcasts in more than 100 languages. Due to this tools construction, we do not need to worry about using any big data tools yet.

## Feature Extraction:

**Overview:** To extract features from text we will use TF-IDF (Term Frequency - Inverse Document Frequency) to parse the descriptions and episode descriptions . This will leave us with a feature space of TF-IDF scores on words and phrases (uni, bi, and tri-grams) that came from a podcast's description and its episode descriptions.

**Tools/Libraries:** SQL, pandas, regex, scikit learn (specifically: TfidfVectorizer)

**Process:**

1. Take a subset of the data collected (title (string), description (string), array of episode descriptions) and load it into pandas.
2. Loop through each episode description for every podcast and add each 'episode description' to 'description'.
3. Drop the 'episode description' column.
4. Clean the description column (drop everything that is not a letter and convert everything to lower case. )
5. Create uni, bi, tri-grams (note this is very easy with TfidfVectorizer, its just a parameter of the function)
6. Create a TF-IDF matrix, where every row refers to a unique podcast
7. Drop stop words (words like: a, the, he, etc) (This is also just a parameter in TfidfVectorizer)

## Model (Content-Based Recommendation Engine):

**Overview:** Once we have the features, we can calculate a matrix that holds the cosine similarity between all podcasts. This matrix can be stored. Then when you want to know the most similar podcasts to it, we just take the row associated with the podcast from the cosine similarity matrix and perform and argsort. The argsort will return a vector with the index locations of the similar podcasts in ascending order. Therefore, take the reverse order and use the indexes to get the podcast titles.

**Tools/Libraries:** pandas, scikit learn (specifically: linear_kernel)

**Process:**

1. Calculate a cosine similarirty matrix using linear_kernel function from scikit learn and store it
2. Write a function that gives you the most similar other podcasts by using the cosine similarity matrix

## Future Improvements:

The current recommender will give similar podcasts based on topics covered in the podcast (Mainly just similar in terms of content). It will not necessarily give similar podcasts of a particular flavor (i.e. tone, style, bias, etc). To improve upon the current or add a collaborative filter approach to the **similar podcast tool,** we can:

1. **Utilize the other data collected from the iTune's podcasts' websites.**
    1. Convert features into the appropriate format
    2. Perform nearest neighbor search over feature space, possibly changing the weights on certain features
2. **Add a collaborative filter approach to the recommender** (One way to execute without the twitter API is)
    1. Build a web automation script to loop through followers of a podcast and store the podcasts they follow.
    2. Create a matrix that holds users by podcasts data where you have a 1 if the user follows the podcast and zero otherwise
    3. Using that matrix, create a new feature that holds the sum of all the similar follows to this podcast. Example, if you are calculating the new feature for podcast A, you filter the 'user by podcast matrix' on a 1 in the column that represents podcast A. Then you sum over the rows (add each users vetcor) to get a vector that you can normalize by the number of users that followed that podcast A.
    4. Perform a nearest neighbor search with the old and new features, choosing which features you may want to weigh more.