# Script

<Title Slide>

You know, I was at the bar that serves a lot of craft beers… The thing is, There is A LOT of different craft beers.

You have your lagers, pilsners, stouts, or ales. And they branch out too! You have your pales, your reds, golden, blonds, browns, IPA’s and so on.

Was it made with barely, wheat, or rye? Where did the malt come from, The US, Germany, the UK, Canada, Argentina? Where was it brewed? Because the water quality of the region comes into play

Was a fruit involved? Or a vegetable? Or some other plant or spice?

Don’t forget about the the color index, Alcohol by volume, bitterness, or hopiness.

**There are a lot of choices** and ways to make that choice. And unless you are a brewer, half of those things I just said you probably don’t even care about. As a beer drinker, the only thing that really matters, **is if you like it or not.**

**-pause**

You’re probably wondering why I’m talking about beer when the screen behind me is about movies…

The answer is simple… I just came from the bar <beat>

And to drive the point that recommender engines solve the problem of choice fatigue. By filtering options so only the best choices remain… and to show that their **applicability** goes beyond movies or beer.

But yeah, Let’s talk about movies!

The idea was to build an app that choose a movie for you to watch that you would like.

And that app exists and it’s called Netflix.

<Slide 2: netflix>

So when you open Netflix, you are still bombarded with choice. You have a list titled “suggestions for you.” You could select a genre and it throws even more lists of movies it thinks you would like.

<slide 2.5: pandora>

So lets take the Pandora approach. You tell the app what you are in the mood for watching. And It will choose for you. So we will reduce the choices to “pick a genre” and we’ll find something you’ll like.

<Slide 3: the hybrid engine>

The recommendation engine we’ve built is a hybrid of content and collaborative based filtering.

Let’s start with content-based filtering.

Our dataset has over 10,000 movies and each movie has a set of a 1000 attributes.

Lets have a look at what that looks like

<slide 4: Action movies wordcloud>

Here is a wordcloud of attributes that are commonly associated with Action movies

<slide 5: comedy wordcloud>

Now here’s comedy

<slide 6: thriller>

Let’s check out thrillers

<slide 7: user profiling>

When a user likes a movie. They are actually like a set of attributes. And conversely the same with disliking.

So when finding a movie in a genre that a user would like. Just find the movie with the most attributes that the user has already liked.

<slide 8: evaluation & performance>

We trained our models on the watch history of individual users minus the latest one to see if we could predict what they would rated it.

On a 5 star ratings scheme. Our fastest model was able to predict at about +/- 1 star across users.

Our most accurate model was able to predict between .8 – 1 stars with better accuracy for users who have rated more things. The only downside of it taking about 3x longer. Which sounds like a lot, but it took a grand total of 3 seconds to sort through our most crowded genre to find a movie for our most active user. Could easily be hidden in a buffer or with a cute graphic saying “finding the perfect movie for you…” because we love you and you are our customer.

If I go to my local bar and the bartender who knows me asked what’ll be having. If I reply “surprise me!” and they take 3 seconds to think about it. I’d be happy cause I knew they actually took the time to think about it.

<slide 9: strengths >

* **Users can get started more quickly.** Content-based filtering avoids the cold-start problem that often bedevils collaborative-filtering techniques. While the system still needs some initial inputs from users to start making recommendations, the quality of those early recommendations is likely to be much higher than with a system that only becomes robust after millions of data points have been added and correlated.
* **New items can be recommended immediately.** Related to the cold-start problem content-based recommenders don’t require other users to interact with an object before it starts recommending it.
* **It’s technically easier to implement.** The data science behind a content-based system is relatively straightforward. The computational cost isn’t high either and can be updated in real-time.

<Slide 10: Weaknesses>

* **Lack of novelty and diversity.** It’s also important for a recommendation engine to come up with results that are novel (that is, stuff the user wasn’t expecting) and diverse (that is, stuff that represents a broad selection of their interests).
* **Attributes is hard work.** Someone had to watch the movie and assign all 100 attributes. In the case of Pandora. It takes a musical expert 20-30 minutes to evaluate 400 attributes for a single song. If this is all they do for an 8 hour day, they can complete 16-24 songs. It’s still gonna take 100 days to go through all of Jay Z’s albums!
* **Attributes may be incorrectly or inconsistently applied.** Content-based recommendations is only as good as the experts that are assigning these attributes.

This is why we made a hybrid engine. Where content filtering is weak. Collaborative filtering is strong. Collaborative filtering specializes in novelty and diversity.

Attributes, get em out of here, it doesn’t need them.

**<**slide 11: collaborative filtering>

I’m going to do a quick demonstration of how collaborative filtering works

Now in the case of our dataset. Add the other 130,000 users and 10,000 movies… and do what we just did for every non-rated movie for every user.

<slide 12: Evaluation and Performance>

So our Collaborative Filtering model was evaluated using 5 cross fold validation.

That is, I have 20 Million ratings. I took out 4 million at a time to see if the remaining 16 million could predict them… And did that 5 times.

It went pretty well, +/- .7

Now for performance… And here’s a picture of Catching Fire because you’ll need a lot of processing power and memory otherwise your server will catch on fire! It took a VERY POWERFUL computer 30 minutes to train the model.

If I want to my favorite bartender and said “surprise me!” and they took 30 minutes to get me a drink… I’d find a new bar.

So real-time updates are not feasible and must be done periodically, like once a day or something. But near- real time updates can be simulated by storing the predicted recommendations in its own database and using the content-filtering to make sure the predictions are still relevant.

---

<slide 13: Strength and Weaknesses>

I’ve already talked about some of the strengths, so let’s talk about the weaknesses.

* **Complexity and expense.**
* **The “cold start” problem.**
* YMMV