*instinctools

# Recommendation Systems for Streaming Services

*unleashing the power of binge-provoking algorithms*

2021

## Contents

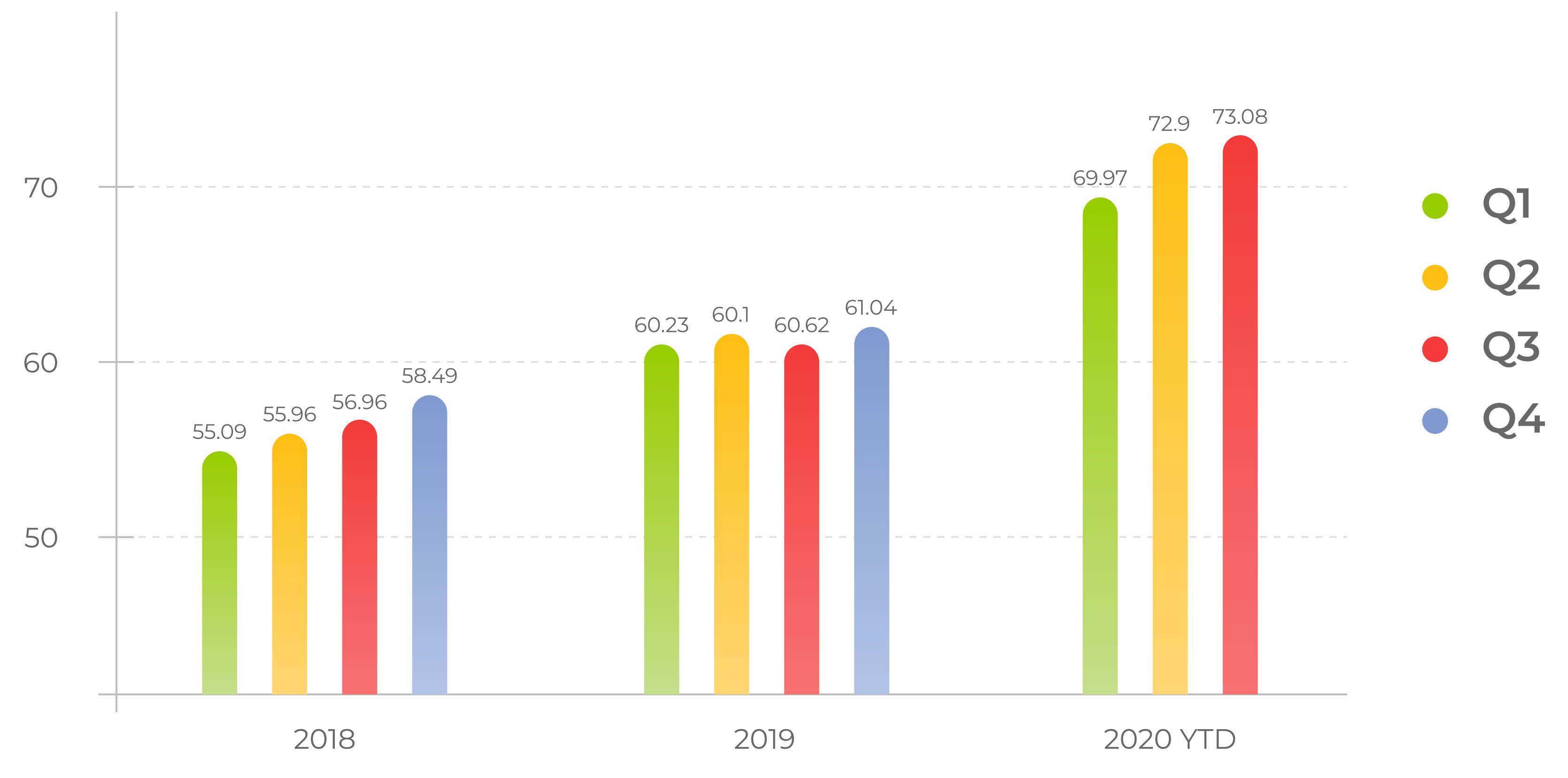
## Adding value to the content

In recent years, streaming services such as Netflix, YouTube, Amazon, Hulu, Disney+, among others, have revolutionized the viewing habits of millions of people all over the globe.

Consumers started shifting away from the traditional paid TV services to something called ***video on demand***. According to a Deloitte survey from April 2021, eighty-two percent of U.S. consumers subscribe to at least one paid streaming video service, while the average subscriber has four paid video streaming services.

Although the COVID-19 outbreak is sometimes considered one of the main explanations why streaming services are thriving, this is not exactly the case. By taking a look at the subscriber count percentage change in 2018, 2019, and 2020, it becomes clear that streaming services exhibited growth before and during the pandemic. In essence, COVID-19 has just catalyzed what was already happening in the SVOD (Subscription Video on Demand) market.

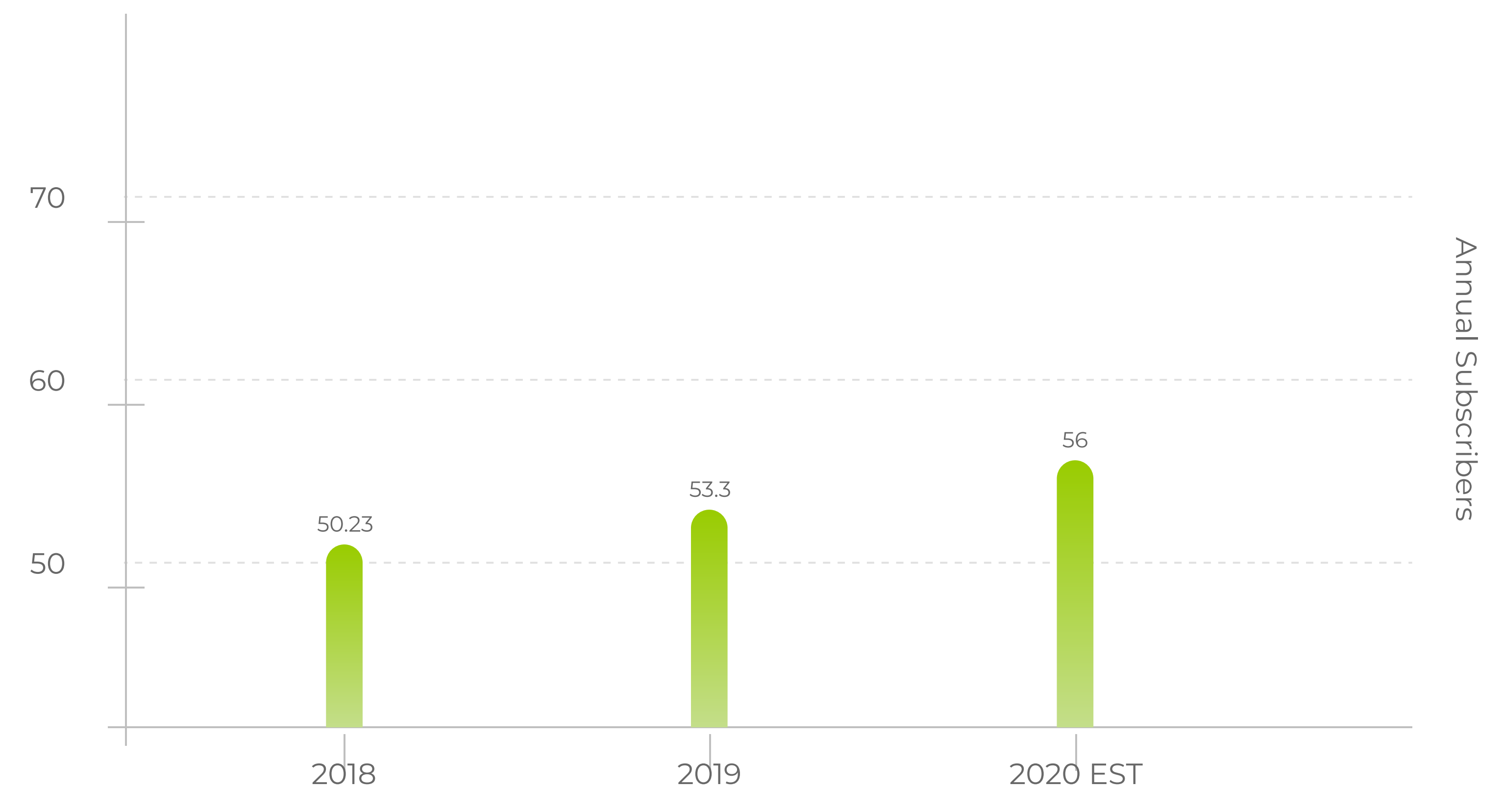
**Netflix Subscribers (in millions)**



Annual Percent Change **2019 - 4.36%**

Annual Percent Change **2020 - 19.72%**

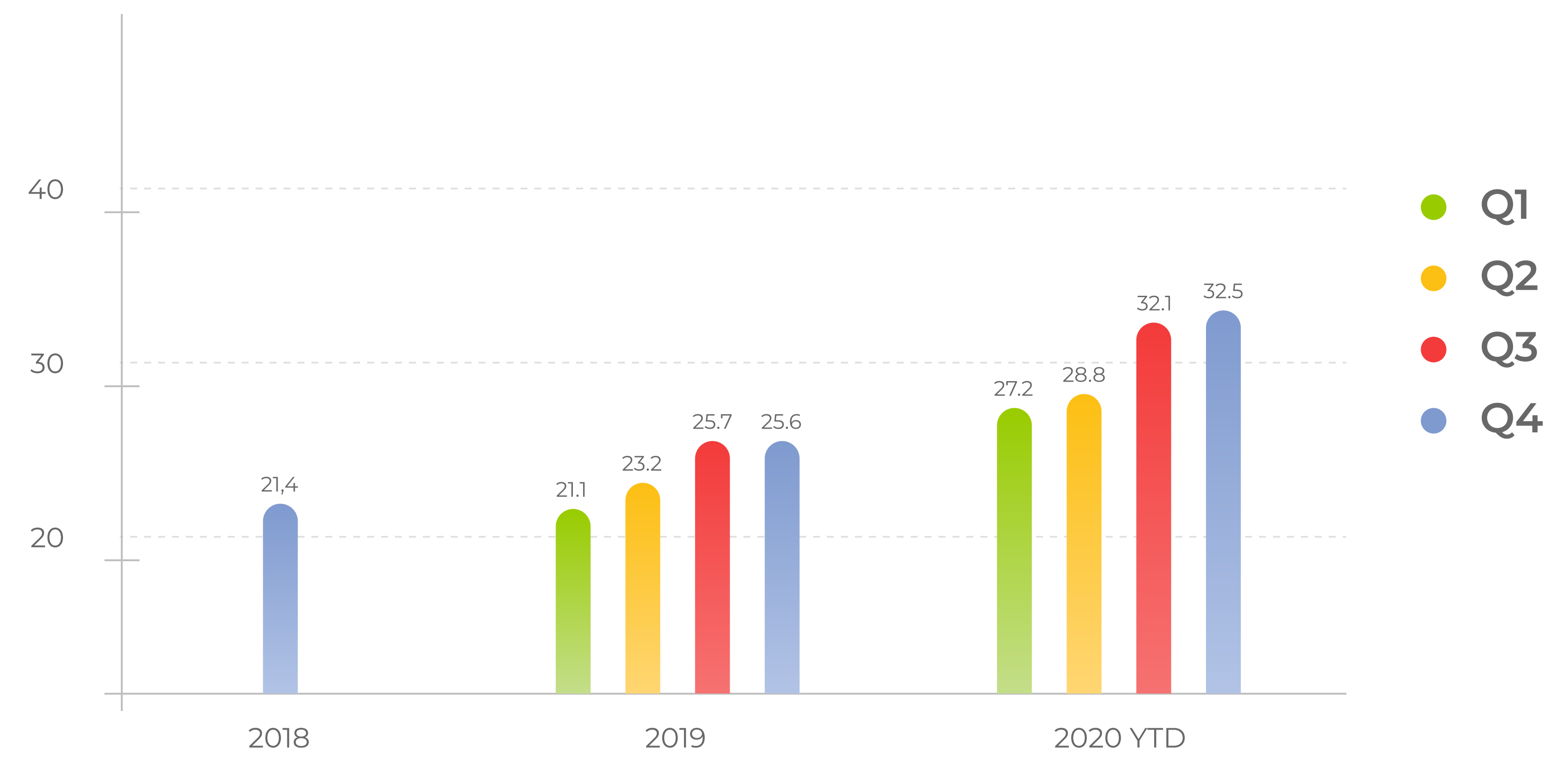
**Prime Video Subscribers (in millions)**



Annual Percent Change **2019 - 6.11%**

Annual Percent Change **2020 - 5.07%**

**Hulu Subscribers (in millions)**



Annual Percent Change **2019 - 19.63%**

Annual Percent Change **2020 - 26.95%**

In other words, the world-scale lockdown was the reason people became bored and explored streaming platforms for binge-worthy content, but it definitely wasn't the reason for them to stay. So, what was it then?

While linear broadcast and cable TV offer whatever is playing on at the moment, streaming TV allows its audience ***to choose what to watch, when to watch, and where to watch.***

However, what at first seems like an absolute, undeniable advantage over less flexible traditional competitors, is overwhelming for the majority of human beings. The dramatic explosion of choice that streaming services are renowned for has paradoxically become an anxiety-provoking problem rather than a brilliant solution. Customers pay a flat monthly rate to check out as much content as they want. The problem with this business model is that new members can usually think of a dozen movies they want to see but after that, they are not sure about what to watch next and their requests slow.

It used to be that if you wanted to rent a movie, you had to rely on a flesh-and-blood judgement — yours, or that of someone you trusted. You might read newspaper reviews or consult your friends, or, if you were lucky enough, there would be one of those cinema enthusiasts in your local video store who could size you up at a glance and suggest something suitable.

But with the rise of the Internet television, all that has changed drastically. It would be next to impossible for viewers to make sense of their choice without being navigated through terabytes of content provided by streaming platforms.

Consumer research held by Netflix suggests that a typical Netflix member loses interest after 60 to 90 seconds of choosing, having reviewed 10 to 20 titles. Users either find something of interest or the risk of them abandoning the service increases substantially.

The large platform needs a recommendation engine to automate the search process for users. Netflix claims that more than 80% of the TV shows people watch on their service are discovered through the recommendation system. Essentially, behind every “you might also like” recommendation is an algorithm built on the collected data. Streaming platforms make recommendations based on the metadata about the movies (such as the genre, categories, and more), the user’s viewing history, and the viewing history of other members with similar tastes.

One may argue that a recommendation engine is capable of making accurate predictions as it’s less personalized than a store clerk, who knows a lot about you — your age and profession, what sort of things you enjoy, and can even read your current mood. (Are you feeling miserable? Maybe it’s not the day for Lars von Trier’s depression trilogy.) That’s true, a recommendation system knows very little about who you are — only what you’ve watched and whether you rated it highly or not. But the computer has numbers on its side. It may know only a little bit about you, but it also knows a little bit about a huge number of other people. This lets it detect patterns we often cannot see on our own.

## How does a recommendation system work?

What is behind the recommendations provided by streaming services? How does a platform know which video the user would like to watch? The answer is Machine Learning (ML), a subset of artificial intelligence, which helps the system "learn" without human assistance. It gives the platform the ability to automate millions of decisions based on content metadata and user activities.

Most of the credit for the streaming TV breakthrough goes to recommendation engines, which, in their turn, would never exist without ML techniques. Let's talk numbers. In 2009, Netflix paid one ***million*** dollars to create a system of accurate recommendations. In 2015, the company claimed that the combined effect of personalization and recommendations save them one ***billion*** dollars per year — a return on investment (ROI) of 1,000%.

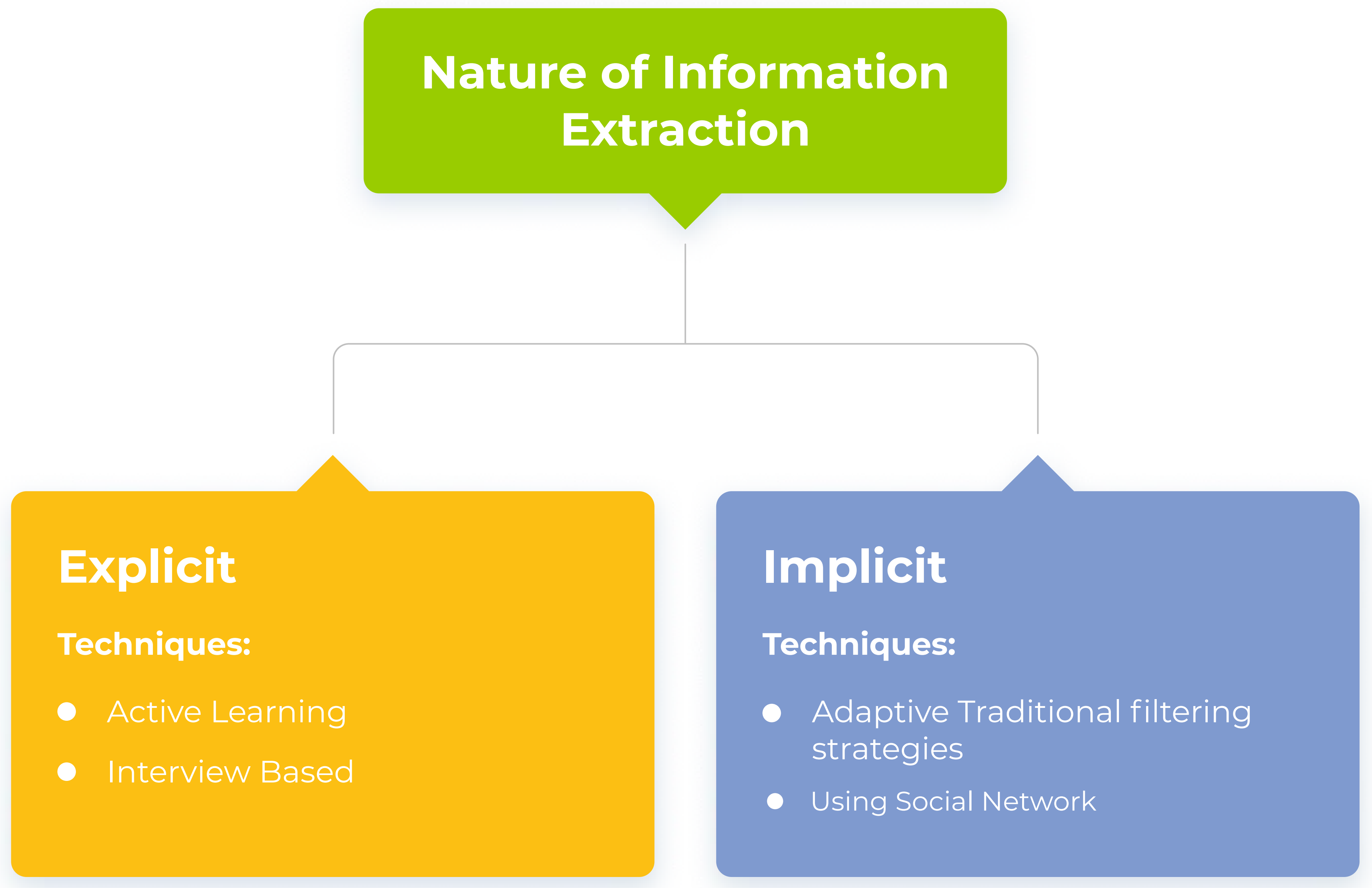
Machine learning algorithms are designed to work diligently to provide users with everything they want, and some things they don't even know they want yet by means of recommendations.

### How the information is collected

Machine Learning algorithms can't make perfectly calibrated recommendations for users to stay without enough information being fed into them. This information can be collected either **explicitly** or **implicitly.**

Explicit information is something that is obtained from direct interaction with users. They are either asked to answer a couple of questions akin to 'What film genre do you prefer?' or 'Who's your favorite actor?' or rate some given items — by giving a thumbs up to "Squid Game" the user literally entitles the system to decide what he/she will binge-watch next. It hardly needs saying that when the system collects information explicitly it can acquire more relevant information as it controls what it asks. The problem is that users are not that eager to take part in queries or even rank the content because of the time and effort required.

Implicit solutions are built on the system's ability to extract and analyze information about its users' preferences without their involvement. You don't need to openly say that you liked "The Queen's Gambit," the system figures it out behaviorally because you finished it in a couple of days. Most of the truly useful data is implicit. If used properly, it allows to avoid on-the-surface recommendations, making users constantly expand their list of shows to watch and become less prone to leave. However, finding this information and being able to analyze it in an appropriate way is exactly the kind of challenge that recommender engines are designed to meet.



# Types of recommendations

In a very broad sense, there are two categories of recommendations: non-personalized or popularity recommendations, which simply recommend what's popular among all users, and personalized ones, making suggestions based on the information about the content or the user's viewing habits. Personalization systems can be further broken down into three main types: content-based, collaborative filtering, and hybrid.

## Popularity recommendations

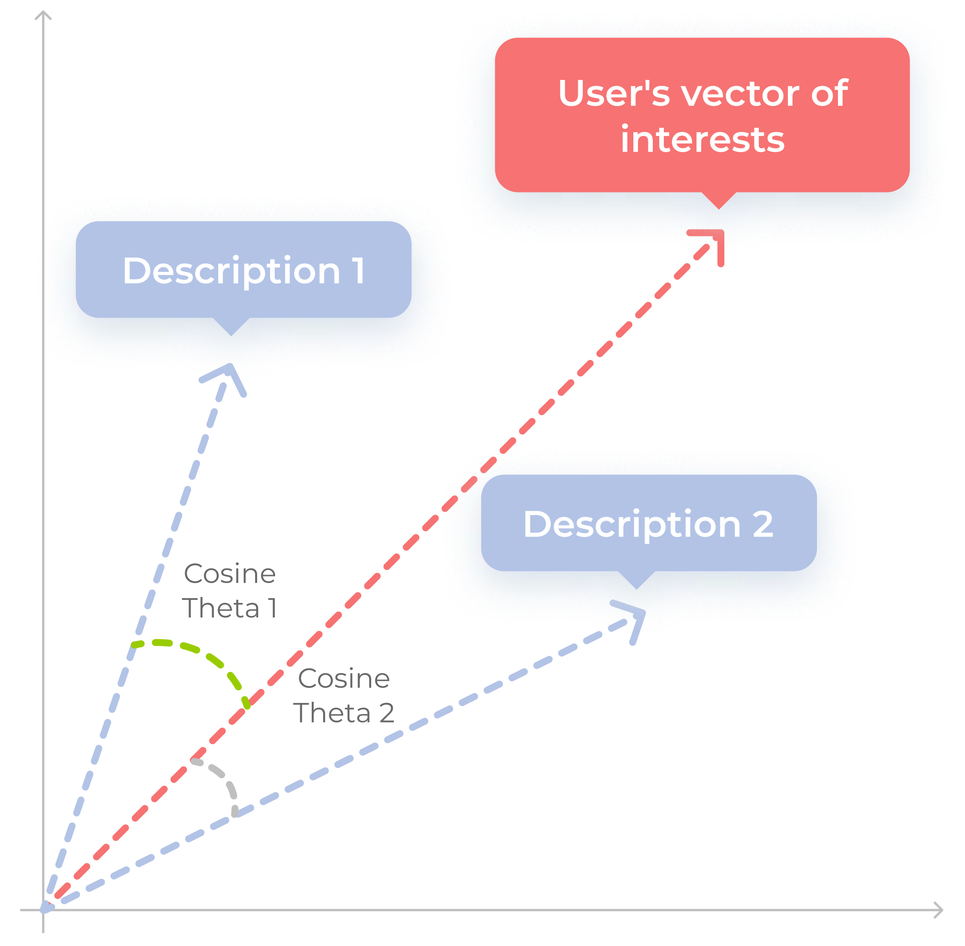
Recommending popular items might seem tempting as it's the path of least resistance. Within this simple strategy, the recommender can suggest items that have been "on trend" in the last few hours or days, or recommend the most popular items in each category that the user explores. The absolute advantage of this recommendation type lies in its simplicity, which is embodied in less computational usage and trouble-free updates. Unfortunately, that's exactly where the benefits end and flaws start to unfold. 'If everyone likes it so will you' principle is too shaky to rely on. Poor accuracy and lack of personalization leads to user dissatisfaction, which is the last thing the subscriber should feel being a click distance away from canceling their subscription. In this case, the risks of going too small, without bothering about complex solutions, are much higher than the risks of going too big.

## Content-based recommendations

In response to the lack of accuracy, new and more precise personal recommendations began to be developed. Personal recommendations use the maximum information about users based on what they have already watched.

In a **content-based approach**, the 'content' is recommended to the user according to their previous watching experiences, defined by personal ratings. The more a film fits the user's preferences the higher the user's potential interest is estimated. The compulsory requirement here is that all the items need to have descriptions such as genre, loglines, cast, etc.

These attributes are represented by vectors in a word space (Vector-Space model), typical of text documents. Each element of such a vector is an attribute, which qualifies the user's potential interests. Similarly, a film is a vector in the same space. While the user is interacting with the system, i.e. streaming certain films/series/shows, the descriptions of all the watched content come together into a single vector of the user's interests. There's only one thing left — finding an item, the description of which is located closely to the user's vector of interests.



The idea seems quite basic... in theory. But when it comes to the nuts and bolts of running the algorithm, it turns out that not all the analyzed elements are equally important. For example, articles or conjunctions may repeat themselves in almost all the content descriptions, but this doesn't make this content any similar ***in sense.*** Adding little sense to the content but being widely used, these parts of speech should not be estimated in the same way as meaningful components. Otherwise, it'll cause confusion in the algorithm.

The problem can be addressed with a well-known system for text analysis, called TF-IDF (time frequency - inverse document frequency), which gives more weight to relevant terms instead of insignificant ones. Thus, before the overlapping elements are defined, they have to be weighed in the first place. Rare attributes should have a larger weight than simply popular ones. If content recommendations were built on the popularity of all the words indiscriminately, the advice on what to watch could hardly be accurate, as it would be based on perpetual ***'the-s'*** and ***'in-s'*** rather than something more worthwhile such as 'heartfelt' or 'dangerous'.

$$W_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

**TF-IDF**

**Term x within document y**

$tf_{x,y}$ = frequency of x in y

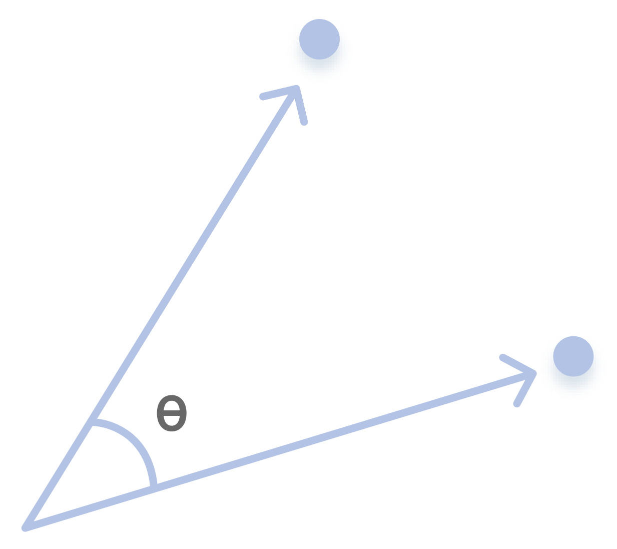
$df_x$ = number of documents containing x

$N$ = total number of documents

To be honest, the approach is not all that groundbreaking. Content-based filtering almost entirely replicates query-document matching systems used by search engines. The difference between them lies only in the form of the search query. In a recommendation system, it's a vector that describes the user's preferences, while in a search engine, these are keywords related to the requested document. Ever since search engines started to utilize personalized search functions, the difference between the two has faded away even more.

But how does it work in practice? The math for the similarity between two vectors is measured by cosine similarity, and it looks a little bit like this:

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



When a new rating is added, the vector of interests updates incrementally involving only the elements that have been changed. During the recalculation, it makes sense to give a new rating a little bit more weight since the preferences can change over time.

## Collaborative filtering recommendations

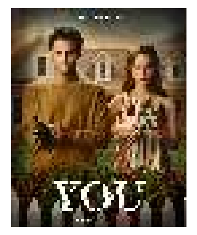
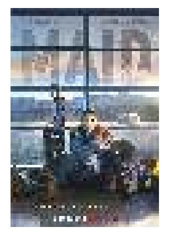
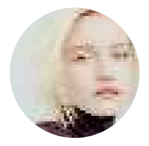
The main idea behind a **collaborative filtering approach** is that similar users have similar tastes. This type of recommendation system makes predictions of what might interest a person based on the preferences of many other similar users. Generally, it assumes that person X is similar to person Y. So, if person X likes "The Crown", and person Y likes "The Crown" and "Queen's Gambit", then person X might like "Queen's Gambit" as well.
Such recommendations are generated due to the "collaboration" of users.

Depending on their focus, collaborative filtering techniques are classified into **user-based filtering** and **item-based filtering.**

### User-based filtering

Imagine the system needs to recommend a movie to Joe. Suppose that Joe has seen the same, give or take, films as Barack and both of them rated these films ***(explicitly or implicitly)*** almost identically. That being so, if Joe hasn't seen "The Becoming", but Barack has and liked it — as a matter of fact, how could he not — the system will offer this documentary to Joe, reasonably assuming that he'll like it too. User-based filtering algorithms look for users with similar consumption patterns as the ones of the active user and give him/her content that these similar users found interesting.

The classic implementation of this technique is based on the k-nearest neighbors algorithm, which suggests that similar things exist in close proximity. For each user look-alike "k" neighbors are being found so that the information about this user can be complemented with the data known from his/her neighbors.
In other words, we create a User-Item Matrix, determined by similar users, to predict the ratings for the items the active user hasn't seen.

| |  |  | | |
|---|---|---|---|---|
|  | 5 | 1 | 3 | 5 |
| | ? | ? | ? | 2 |
| | 4 | ? | 3 | ? |

Although being favorable in many aspects such as accuracy, context independence, and easy implementation, the k-nearest neighbors algorithm has a serious drawback — there is too much data to process. Within this method, all the pairwise distances between vectors (users) have to be measured and there might be millions of users.

Does it mean that you need to give up on the algorithm? Not at all. To a certain degree, this issue can be tackled with high-performance hardware. On the other hand, making some changes to the algorithm seems like a wiser option as it does not imply spending a fortune to maintain adequate functioning of the system. This is something that can be done in the first place: the distance between the vectors should be renewed in batches (for example, once a day) and it's better to update the matrix incrementally, not recalculate it entirely.

## Item-based filtering

Back to ***Joe***. Suppose, he really enjoyed watching "Suits." To recommend him something that would catch his eye, we found people who also rated "Suits" high and noticed that a lot of them also liked "House of Cards." The correlation was high enough to assume that Joe might like it too. This method of collaborative filtering is called item-based filtering. Item's recommendation rating for a user is calculated depending on the items' ratings by other users.

The difference between User-based and Item-based methods is that, in case of the latter, we directly pre-calculate the similarity between the co-rated items, bypassing k-neighborhood search.

Moreover, there are several different mathematical formulas to calculate the similarity between two items.

- Cosine-Based Similarity
- Correlation-Based Similarity
- Adjusted Cosine Similarity
- 1-Jaccard distance

Each one uses different approaches to calculate the prediction ratings.

- Weighted Sum
- Regression

| USER-BASED FILTERING | |
|---|---|
| **Pros** | **Cons** |
| Diverse, non-trivial recommendations | Data sparsity |
| Context independence | Cold start |
| - | Scalability |

| ITEM-BASED FILTERING | |
|---|---|
| **Pros** | **Cons** |
| High prediction accuracy | Lack of bold predictions |
| Better performance | - |
| Stability | - |

## Avoiding the confusion between a content-based system and item-based filtering

In simple terms, the item-based collaboration deals with the other user actions on the piece of content you are streaming. For example, you're watching "The Crown" and the system starts recommending you "The Downton Abbey" because many people who watched "The Crown" have also watched "The Downton Abbey".

What's for the content-based filtering, it matches the predefined attributes of the content (genre, cast, description, etc.) so that similar movies or shows will be recommended.

## Content-based systems or collaborative filtering? Hybrid!

All the systems above have their advantages and drawbacks, and the solutions these systems offer are based on different inputs.

For example, collaborative recommenders depend on an overlap in ratings across users and have difficulty when the space of ratings is sparse, for example, few users have rated the same items. Collaborative recommenders work best for a user who fits into a niche with many neighbors of a similar taste. The technique does not work well for so-called "gray sheep," who fall on a border between existing groups of users. Meanwhile, content-based techniques have the problem of being limited by the features that are explicitly associated with the objects that they recommend. For example, content-based movie recommendations can only be based on written materials about a movie: actors' names, plot summaries, etc., because the movie itself is blurred to the system, which puts this model at the mercy of the descriptive data available. Collaborative systems rely only on user ratings and can recommend items without any descriptive data.

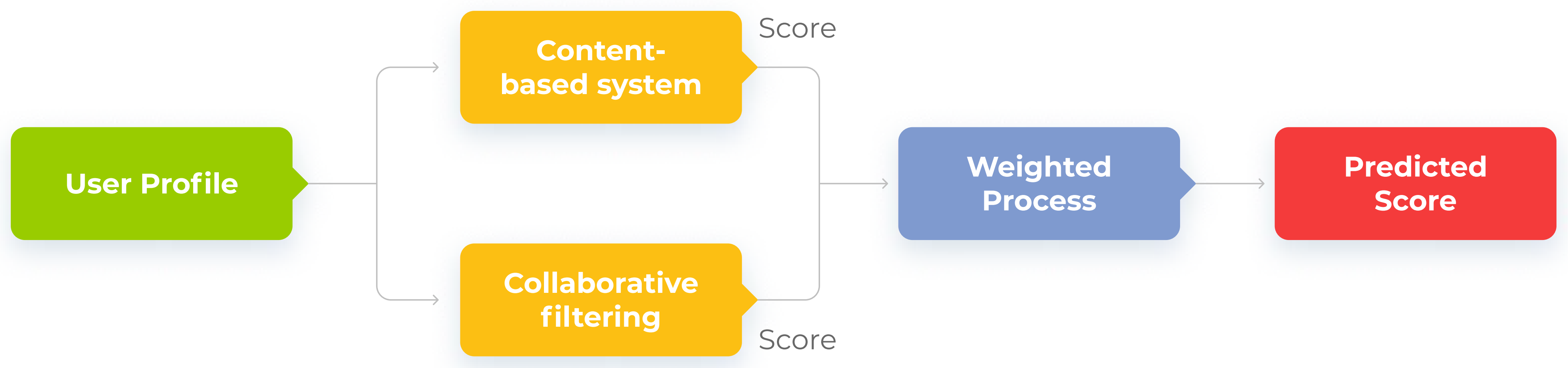
With content-based recommendations, the ability to positively surprise the user with a compelling option becomes a long shot. To get over the threshold of obvious, yawn-provoking suggestions and not undermine their accuracy, the best possible solution is to integrate different prediction methods. And just like that (finger snap), the quality of recommendations can skyrocket.

Hybrid recommender systems combine two or more recommendation techniques to gain better performance with fewer drawbacks of any individual solution. There are several combination types offered within hybrid engines.

## Weighted

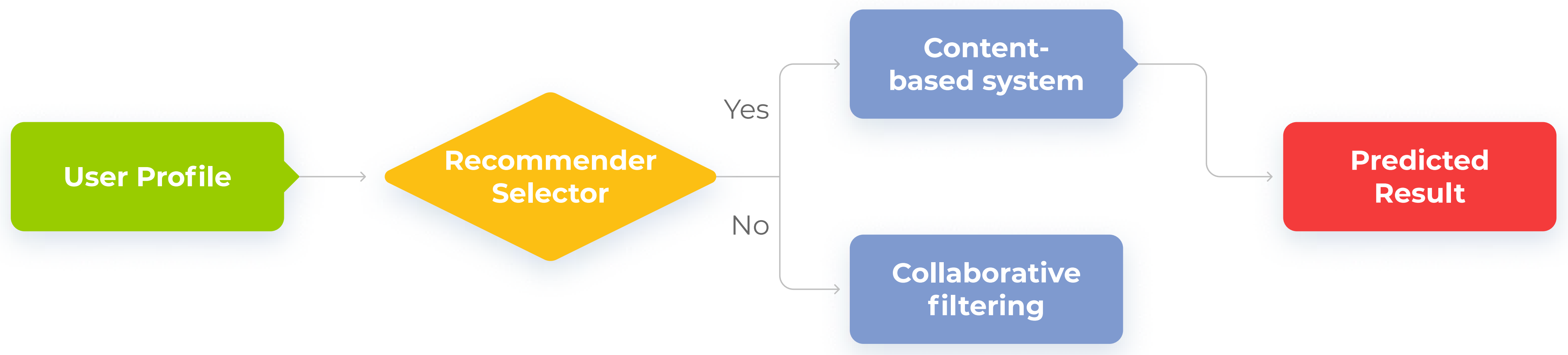
With a weighted approach, the score of a recommended item is calculated based on the results of the recommendation models currently available in the system.

For example, the system can consist of content-based and collaborative filtering models, initially giving equal weight to each (50%/50%), but gradually adjusting the weighting as predictions about user ratings are confirmed or disconfirmed.



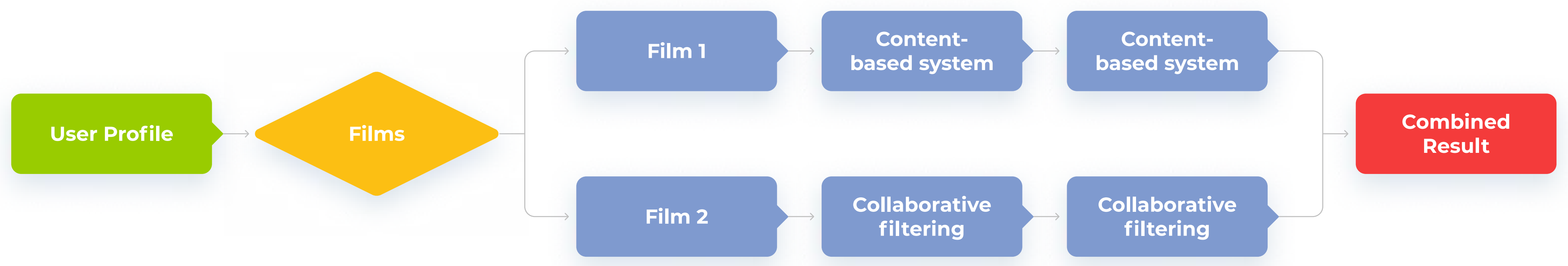
## Switching

The system switches between recommendation models, giving priority to the one which is the most confident in its suggestions.



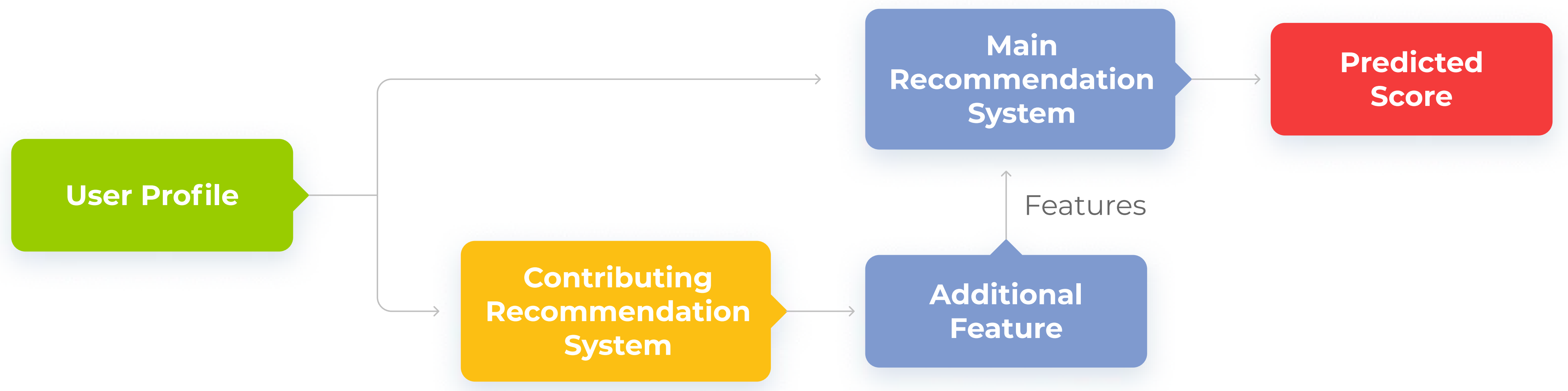
## Mixed

A mixed recommendation system presents a large number of suggestions by producing them simultaneously within different models. It perfectly exemplifies the point that two heads are better than one. A content-based component uses **textual descriptions** of films and shows, which helps to avoid a cold start problem of recommending content to users who are new to the system; whereas collaborative filtering gathers information about **users** to sprinkle the magic dust of serendipity on suggestions.



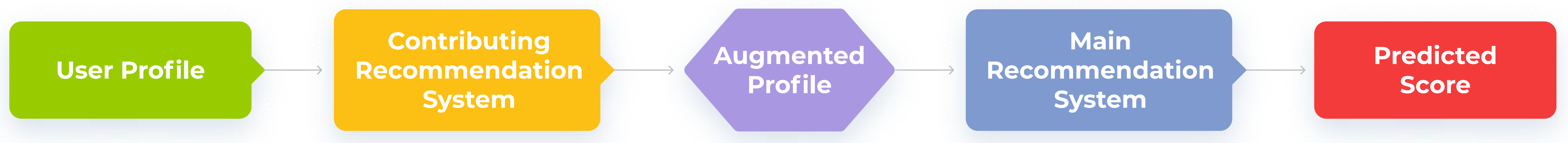
## Feature combination

Another way to reap the benefits of the "recommendation merger" is to throw together features from different recommendation data sources into a single algorithm. In this case, there's a model which is treated as the primary one, and the other model just injects some features into it for a little added precision.


User Profile
Main Recommendation System
Predicted Score
Features
Contributing Recommendation System
Additional Feature

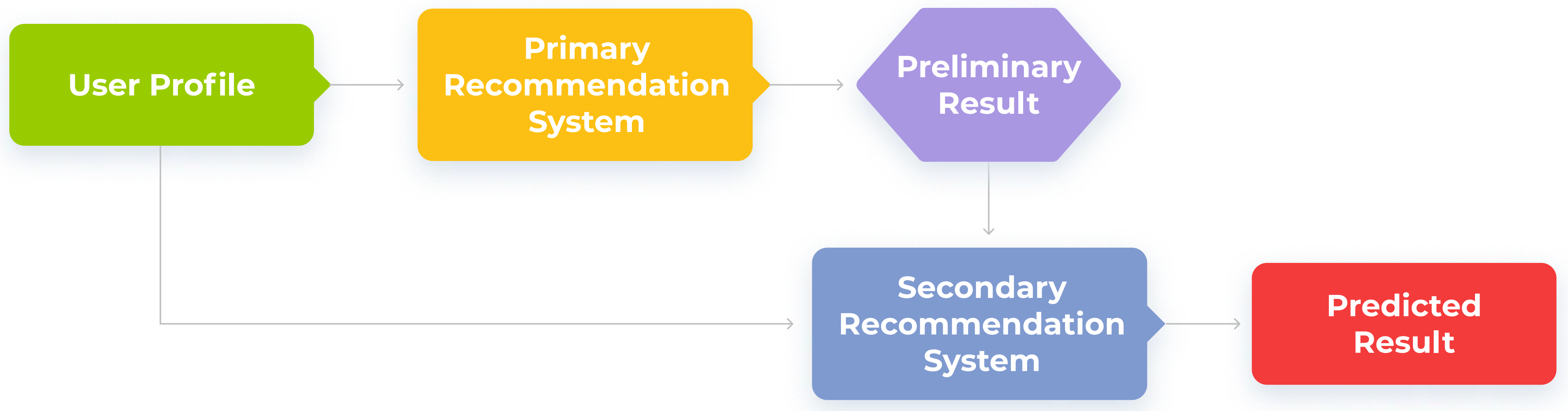
### Feature augmentation

Augmentation offers a way to improve the performance of a core system without modifying it. The point is that a contributing technique is employed to produce a rating or a classification of the user/item profile, which then flows into the main system to make final predictions.


User Profile
Contributing Recommendation System
Augmented Profile
Main Recommendation System
Predicted Score

### Cascade

First things first — the principle of cascade hybrid states. Unlike in the previous methods, in this one, preliminary results are generated by a model of primary importance and then are refined by a secondary model. Such an approach allows to avoid applying a lower-priority technique to users or items that are already well-differentiated. At the same time, this technique really comes in handy in the case of equal scoring and missing data issues.


User Profile
Primary Recommendation System
Preliminary Result
Secondary Recommendation System
Predicted Result

## Key challenges in making recommendations

Although current recommender systems do a great job predicting users' preferences, they are still far from being perfect and can be improved significantly if some challenges, revolving around recommendation algorithms, are overcome.

### Cold start problem

As soon as the system has learned to make recommendations for users and content that have been there for a while, the problem of recommendations for new content or users, aka a "cold start" problem, comes up. This highlights two questions: how to advise a film or series that nobody has seen yet and what to show a user who's just signed up.

While the solution to the first problem is quite obvious — to build an initial recommendation based on content metadata (e.g. the genre, cast, keywords, etc.); the second one is not that easy to tackle. The basic answer is to provide newcomers with popular items. The flip side is that recommendations work best when they offer low-frequency items that match with the user's preferences. There's no need to bring up Titanic, if a user intended to watch it, they have already done it. The realm of the user's interests might not be limited to the top-best-trends-ever list; and making the audience bored, when canceling a subscription has become as easy as clicking a button, is the last thing any streaming service wants to do. A good way to warm the engine up a bit is to get some explicit information from the user by giving them a simple questionnaire to fill out or try a location-based approach that doesn't require a history of user ratings but generates predictions relying on the ratings of the people who are geographically close to the user.

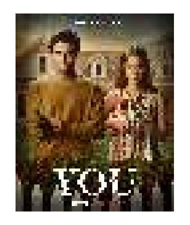
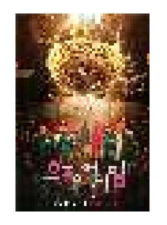
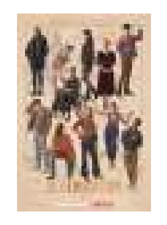
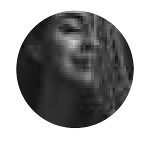
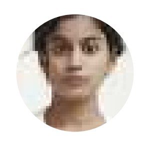
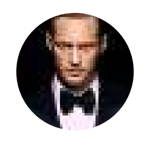
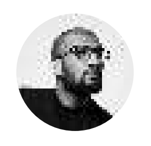
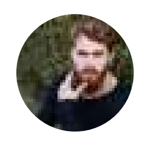
## Infrastructure problems

Infrastructure problems originate mostly from the necessity to update the models all the time. The biggest mistake people make in regard to machine learning is thinking that the models are just like any other type of software. Once a model is built and goes live, people assume it will continue working as normal. Indeed, there are models, for example, voice recognition, that can be retrained less frequently because their input doesn't change over time. For recommenders, if the predictions are offered by ***static*** machine learning models, they'll be less accurate, and less useful in a matter of days, which isn't acceptable by any means.

Taking into account that the recommender's suggestions are only as good as being up-to-minute, the issues related to server load, the speed of result generation, and recommendation updates won't fail to spring up. To crown it all, for a platform that has millions of users and terabytes of content, scalability becomes a major stumbling block. An adequate way to mitigate scalability issues is by using clustering techniques. There are two significant benefits of implementing them. Firstly, they diminish the sparsity of the data set. Secondly, they divide the data into smaller groups, which significantly increases prediction generation speeds.

## Data sparsity

Think of a recommender system that offers thousands of films and shows to millions of users: if you stored the data about user-product interaction in a matrix, it would be a huge amount of data consisting of lots of zeros. In other words, the value of certain data points would be unknown, since the active users only rate a small number of items. Lack of ratings often results in poor recommendation quality.

| |  | |  |  | | |
|---|---|---|---|---|---|---|
| | 4 | 0 | 3 | 0 | 5 | 0 |
|  | 0 | 2 | 0 | 0 | 4 | 1 |
| | 0 | 0 | 1 | 0 | 2 | 5 |
|  | 0 | 0 | 3 | 0 | 0 | 1 |
|  | 1 | 4 | 0 | 0 | 2 | 5 |
|  | 5 | 0 | 2 | 1 | 0 | 4 |
|  | 0 | 2 | 3 | 0 | 4 | 5 |

It's necessary to have an algorithm that can interact with sparsely represented data and, in many cases, this is not offered off-the-shelf. As most of the values are zeros for most pieces of content, they bring relatively little information and if you train your model on such data, then you end up with a huge number of parameters that are useless most of the time. Models with a big number of parameters are problematic on their own, because to estimate the parameters you need large amounts of data and the optimization algorithms that work well in such settings. So in cases where sparse data is common, like language data where the distribution of words is very unbalanced, we ***usually*** use some kind of dimensionality reduction. Apart from that, instead of changing the dimensionality of the data, some versions of machine learning models that are robust towards sparse data might be used. Also, to reduce the noise produced by sparse features, some of them can be removed from the model. For instance, rare words can be taken out of text mining models. However, sparse features that have important signals should stay.

## Privacy concerns

Allowing streaming services to track their users' watching habits seems like a clear trade-off: being under surveillance in exchange for a plethora of options to binge on. It would be nice if algorithms were more transparent, so users can understand why certain recommendations are made to them. Netflix does a good job explaining how its system works, but what about others?
People trust the information more if they know how it's been gathered. Not to imperil the user's loyalty, the service had better provide a lucid explanation for the recommendation received. As part of an explanation, there might be several things to unveil: the rating given to the film by similar users, an attribute the "match" is based on (genre, actor, etc.), the degree of the system's confidence in the provided rating, and more. For example, "you might like ***'The Morning Show'*** because it stars Steve Carell" or 'users with tastes similar to yours rated this series 4.5 out of 5.' Not to overload the interface, some of this information can be put into a special section behind the 'Tell me more' button.
Privacy consciousness should become an integral characteristic of businesses capitalizing on Big Data. At the end of the day, in competition for users' attention, it's not the ones who skillfully manipulate personal data that will win, but those who are transparent about their solutions.

# System quality assessment

Recommendation system testing always creates many questions, mainly due to the ambiguity of the notion of "quality." Generally, in machine learning tasks, there are two main ways of testing:

- **offline retrospective testing**
- **online testing**

Both of them are widely used in the development of recommendation systems.

In offline testing, the main limitation we have to deal with is that we can assess the prediction accuracy only for the content the user has already rated.

The standard approach here is cross-validation with the ***leave-one-out*** and ***leave-p-out methods.***

Leave-one-out model is learned on all the items rated by the user, except for one, and is tested on this one item. It's done for all the 'n' items, and based on the 'n' ratings of quality; the average is calculated. Leave-p-out is almost the same, but at each stage, the 'p' dots are excluded.

All the quality metrics can be roughly divided into three categories:

- Prediction accuracy — assessing the accuracy of the predicted rating,
- Decision support — assessing recommendation relevance,
- Rank accuracy — assessing the ranking quality of the given.

Unfortunately, there's no one-size-fits-all metric, so anyone, who's responsible for testing the recommendation engine should choose the criteria that will meet their particular needs.

There's one more thing that should come under close scrutiny during the quality assessment and that's the user's behavior. The problem is that describing the cognitive processes of the user while he/she is choosing a film is beyond the capacity of any existing metric. Users make their decisions under the influence of many factors.

Understanding the logic of the user at least partially is possible with the help of online testing.

The first and, probably, most obvious scenario of such testing is the event analysis of the website. We check what users are doing there, whether they pay attention to the recommendations, which features of the system are popular and which are not. To get a grasp of which algorithm works best or just try out a new promising idea, A/B testing can be done. Specifically, our A/B tests randomly assign different members to different experiences that we refer to as cells. For example, each cell in an A/B test could map to a different video similars algorithm, one of which reflects the default (often called "production") algorithm to serve as the control cell in the experiment — other cells in the test are the test cells. We then let the members in each cell interact with the product over a certain period of time. Finally, we analyze the resulting data to answer several questions about member behavior from a statistical perspective, including:

- Are the members finding the part of the product that was changed relative to the control more useful? For example, are they finding more videos to watch from the video similars algorithm than in the control?
- Are the members in a test cell streaming more than in the control cell? For example, is the median or other percentile of hours streamed per member for the duration of the test higher in a test cell than in the control?
- Are the members in a test cell retaining their subscription more than members in the control?

A/B test results can act as the most important source of information for making product decisions.

The second testing scenario requires feedback from the subscribers in the form of queries or voting. In general, these are common questions about using the service, for example, "Which is more important: relevance or diversity?", "Which list seems more attractive to you?", etc. An unbeatable advantage of this scenario is that it provides direct answers to the questions.

Such testing might be quite complicated but it's crucial for fine-tuning the systems. It gives the platform the opportunity to know its audiences better and be more flexible at adjusting to their needs. Recommendation models make absolutely no sense without being improved by integrating new parameters that work for the user: time, location, day of the week, etc.

## Recommendation engines: the secret ingredient to the streaming services’ success recipe

The growth that streaming services have been experiencing for the past few years is impressive. But one thing is to capture users’ attention with an unprecedentedly wide choice, and a whole other — to retain them. Recommendation systems, at least well-thought-out ones, comprise various algorithms, which are constantly changing according to the new data available, to predict what users might want to watch next. Eventually, it all comes down not to the content really, but to the personalization that data science provides, opening the door to enhanced experiences and truly the best options out there. The business value of skillfully tailored recommendations is as clear as day — if users find something engaging to watch in a matter of a few seconds, they won’t abandon the service for another entertainment alternative.

According to the Netflix report, the price paid for neglecting the implementation of a recommender engine might be as high as the frustrated user leaving the service after one to one and a half minutes of a futile search. Meanwhile, a well-thought-out system of recommendations can save a lot of money. For Netflix, it’s no less than one billion dollars annually.

The race of streaming service providers to get to know their subscribers best is gaining speed and there is no reason to assume it’s going to slow down any time soon.

According to IBC, global spending on artificial intelligence (AI) is forecast to double over the next four years, growing from $50.1 billion in 2020 to more than $110 billion in 2024. One of the leading drivers for AI adoption is delivering a better customer experience, which also spreads on recommendation systems.

Companies would not spend so much money, if they weren’t sure about the result they plan to obtain, which is expressed in better designed marketing campaigns, higher conversion and retention rates, and increased revenue.

And all that becomes possible thanks to the power of knowing more, epitomized in recommendation engines.

# Recommendation Systems for Streaming Services

*unleashing the power of binge-provoking algorithms*

*instinctools

* instinctools has been a reliable technology partner in various industries for over 20 years. We advise our customers on the path of digital transformation and turn their visions into reality. As an international team with headquarters in Germany fueled by more than 350 experts and ambitious growth plans we focus on Software Development, Cloud Computing and Business intelligence. We build on our strong Partnerships with the likes of Google, Microsoft and Bechtle to be able to create state-of-the-art solutions and solve our customers' challenges, building relationships that last.

Learn more: ***www.instinctools.com***