Tackling the Cold Start Problem in Recommender Systems

9 minute read

As part of my machine learning internship at Wish (https://www.wish.com/), I'm tackling a common problem in recommender systems called the "cold start problem". Cold start happens when new users or new items arrive in e-commerce platforms. Classic recommender systems like collaborative filtering assumes that each user or item has some ratings so that we can infer ratings of similar users/items even if those ratings are unavailable. However, for new users/items, this becomes hard because we have no browse, click or purchase data for them. As a result, we cannot "fill in the blank" using typical matrix factorization techniques.

Fortunately, researchers have proposed various ways to tackle this cold start problem. In this post, I would like to introduce loosely categorized set of papers I found interesting. Note that this is not a holistic survey and I have not implemented many of them, so I'm not sure if they really work in practice.

Approaches (TL;DR)

- Representative based: use subset of items and users that represents the population
- Content based: use side information such as text, social networks, etc.
- Bandit: consider the exploration vs exploitation tradeoffs in new items.
- Deep learning: recent methods that tries to solve some of the issues tackled above but using a black box.

Representative based

If we do not have enough information about users and items, we can rely more on those who "represent" the set of items and users. That's the philosophy behind representative based methods.

Representatives can be users whose linear combinations of preferences accurately approximate other users'. For example, a famous representative based method, Representative Based Matrix Factorization (RBMF)[1] is an extension of MF methods with an additional constraint that m items should be represented by a linear combination of k items, as can be seen from the objective function below:

$$\min_{\mathbf{U},\mathbf{V}} \quad ||\mathbf{R} - \mathbf{UV}||_F^2 + \alpha||\mathbf{V}||_F^2$$

$$s.t. \quad \mathbf{U} \in \mathbb{R}^{n \times k} \text{ and } \mathbf{U} \subset_{n,k} \mathbf{R},$$

$$\mathbf{V} \in \mathbb{R}^{k \times m},$$

Here, we have the reconstruction error similar to standard MF methods, with this additional constraint. When a new user joins the platform, we can ask the new users to rate these k items, and use that to infer the ratings of other $m - k$ items. This way, with a small additional cost on users of rating some items, we can improve the recommendations for new users.

There have been improvements on RBMF proposed, where we can interview only a subset of users instead of all the new users to decrease the burden on the new users[2].

Advantages

- More interpretability, because new users can be expressed in terms of few representative items.
- If you're using MF methods already, this can be a simple extension to handle cold start.

Disadvantages

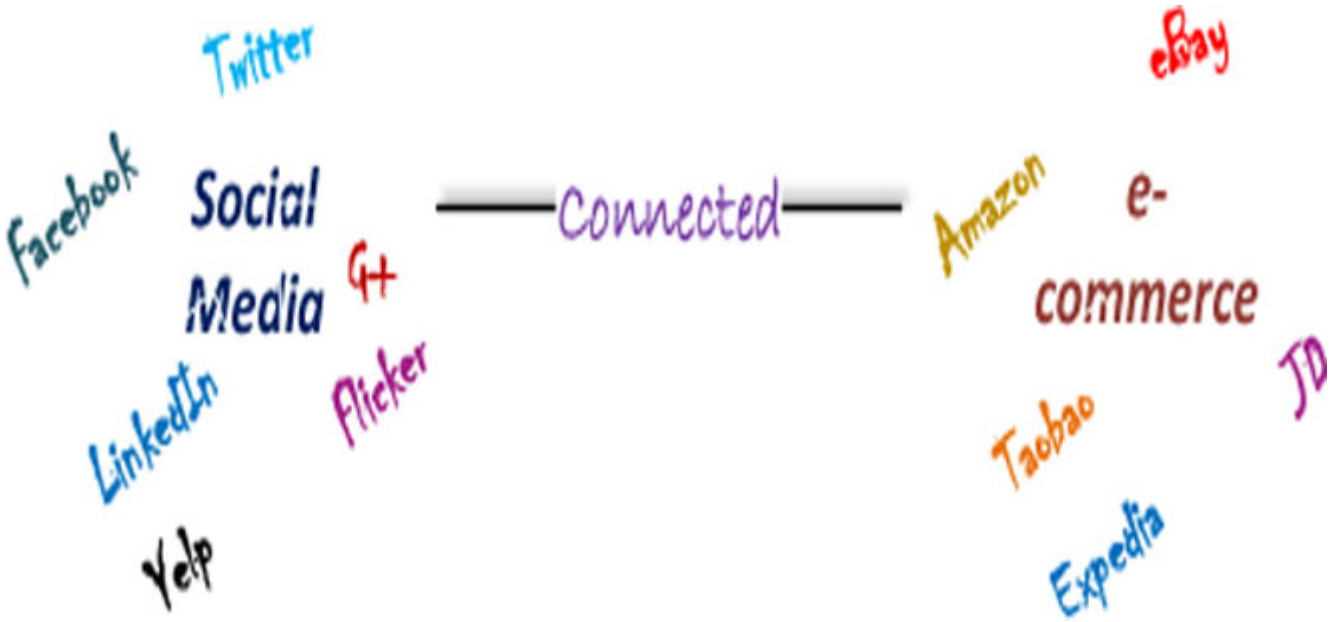
- Need to change UI and front end logic to ask the users to rate the representative items.

Content Based

There are many side information that has been underutilized in recommendation systems. For example, citing one of the content based papers[3] directly:

> *In recent years, the boundaries between e-commerce and social networking have become increasingly blurred. Many e-commerce websites support the mechanism of social login where users can sign on the websites using their social network identities such as their Facebook or Twitter accounts. Users can also post their newly purchased products on microblogs with links to the e-commerce product web pages.*

As other examples, for recommending research papers on journal archives, we not only have the ratings but also the actual text of the paper available. We can incorporate these information as additional features for users to alleviate information scarcity for new users. For textual information, we can for example use LDA to obtain topic vectors for user posts. Even for social networks, there have been proposed a variety of graph embedding methods, which can represent graph nodes in vector spaces[45].



How can we handle these contents in CF methods? Classic CF methods are essentially matrix completion with reconstruction error as its objective. Thus, it is hard to utilize these side information. Researchers have proposed hybrid methods that combines matrix reconstruction objective and content based objectives. These approaches are not cold start specific, so I'll leave it to the readers to explore some literatures in depth [67].

Advantages

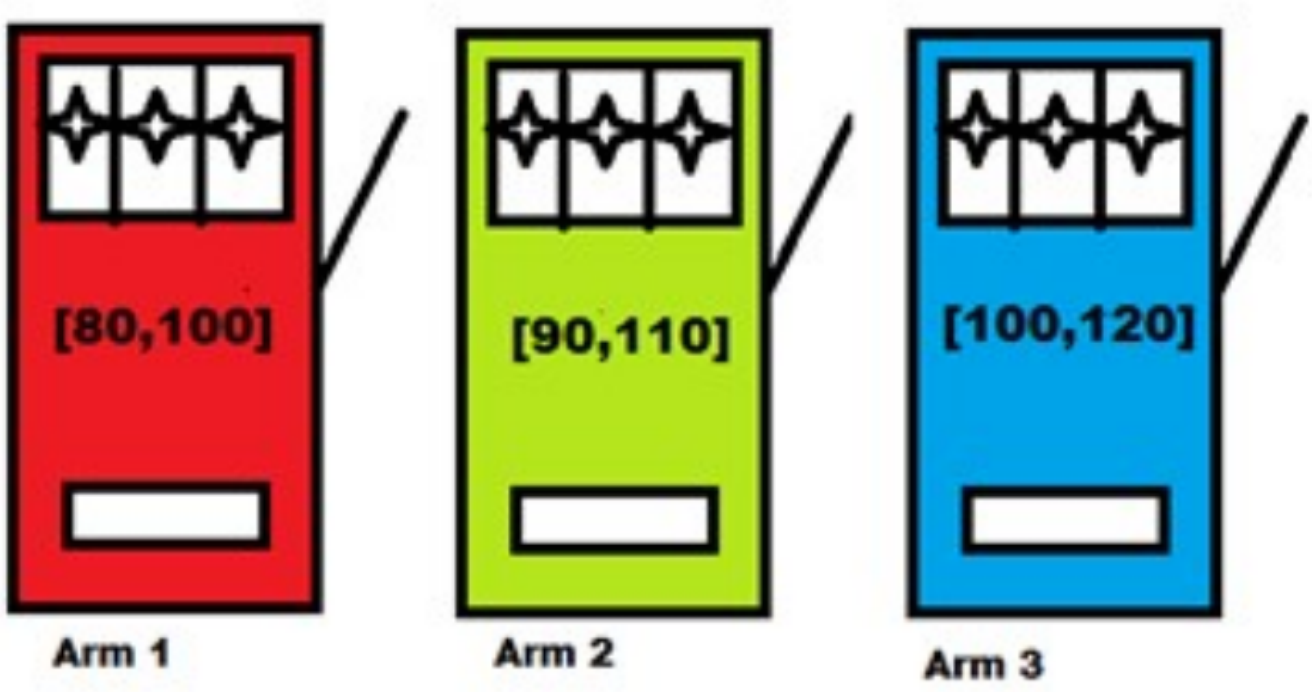
- Can incorporate a wide range of content information.
- Hybrid methods are extensions of MF.

Disadvantages

- Objective functions get cluttered. This is solved with deep methods, with a different downside.
- Lots of feature engineering? Depends on whether you have ready to plug-in user/item features in your db.

Bandit

Cold start can be reframed and solved as a bandit problem. What is a bandit problem? A classic example is the following: you are in a casino and you see k slot machines in front of you. Each slot machine has its own distribution of reward, but you do not have access to that information. Your goal is to maximize your return from playing these slot machines.



Why is this relevant? You can draw an analogy here. Think of an e-commerce problem where you have k new items arriving on your platform everyday. These k items can be thought of as slot machines with different returns (e.g. revenue, profit). But since they are new items, you do not have access to how much users will buy them. Then, recommending a subset of these items is like choosing a subset of slot machines to draw.

In bandit problems, we very much care about the **exploration vs exploitation problem**. If we found a new item that sells very well, we would like to show it to users more (exploitation). But at the same time, you would also want to show others which have not been shown as much, because they can be even more popular then the items you have shown already (exploration).

Solutions to the Bandit Problem

The bandit problem is well studied and there are many ways to solve this problem:

- ϵ greedy[8]: This is the simplest approach. Show an item that's discovered to be most popular (e.g. highest average sales / clicks / views among users so far) with probability ϵ. Show a random item with $1-\epsilon$. ϵ controls whether to focus more on exploitation vs exploration.

- Upper Confidence Bound (UCB)[8]: Select an item that maximizes $\hat{\mu_j} + \sqrt{\frac{2 \ln t}{t_j}}$. μ_j is the average sales / clicks / views of item j so far. t is the number of times you've shown the new items. t_j is the number of times item j was shown. Clearly, μ_j is the criteria for exploitation (e.g. favor items with more average sales) and $\sqrt{\frac{2 \ln t}{t_j}}$ for exploration (e.g. favor items with less exposure so far). This weird $\log$ and $\boxed{\text{\sqrt}}$ comes from the confidence interval of $E[\mu_{i,j}]$, which you can read more about in this paper[8].

- Thompson Sampling (TS)[9]: Think of this as a Bayesian version of the approaches above. You define some prior (e.g. gaussian) with parameters, obtain some observations to calculate the likelihood, and then update the prior with the posterior. We can then choose an action that maximizes the reward based on the posterior distribution of the parameters.

Contextual Bandit

Contextual bandit problem is a generalization of the bandit problem that resembles real world cases more closely. The only difference now is that, in addition to the action (e.g. which items to show) and reward (e.g. sales), you now have contexts (e.g. item discription, pictures, the number of times it was clicked). By utilizing these contexts, we can have a better estimate of the reword. For example, in UCB above, we can add to the reward function a linear combination of the context features. In TS, we can have the likelihood function depend not only on the rewards and actions but also on the contexts.

Advantages

- Easy to implement: The simple bandit algorithms are quite easy to implement, especially compared to sophisticated machine learning methods. So bandit methods are definitely a good baseline to compare to.

- Nice theoretical guarantees: As ML practitioners, you might care little about theoretical guarantees, but thanks to its simplicity, a lot of theoretical analysis has been done on the average and the worst case reward. So you can be rest assured that the algorithms won't be a disaster.

Disadvantages

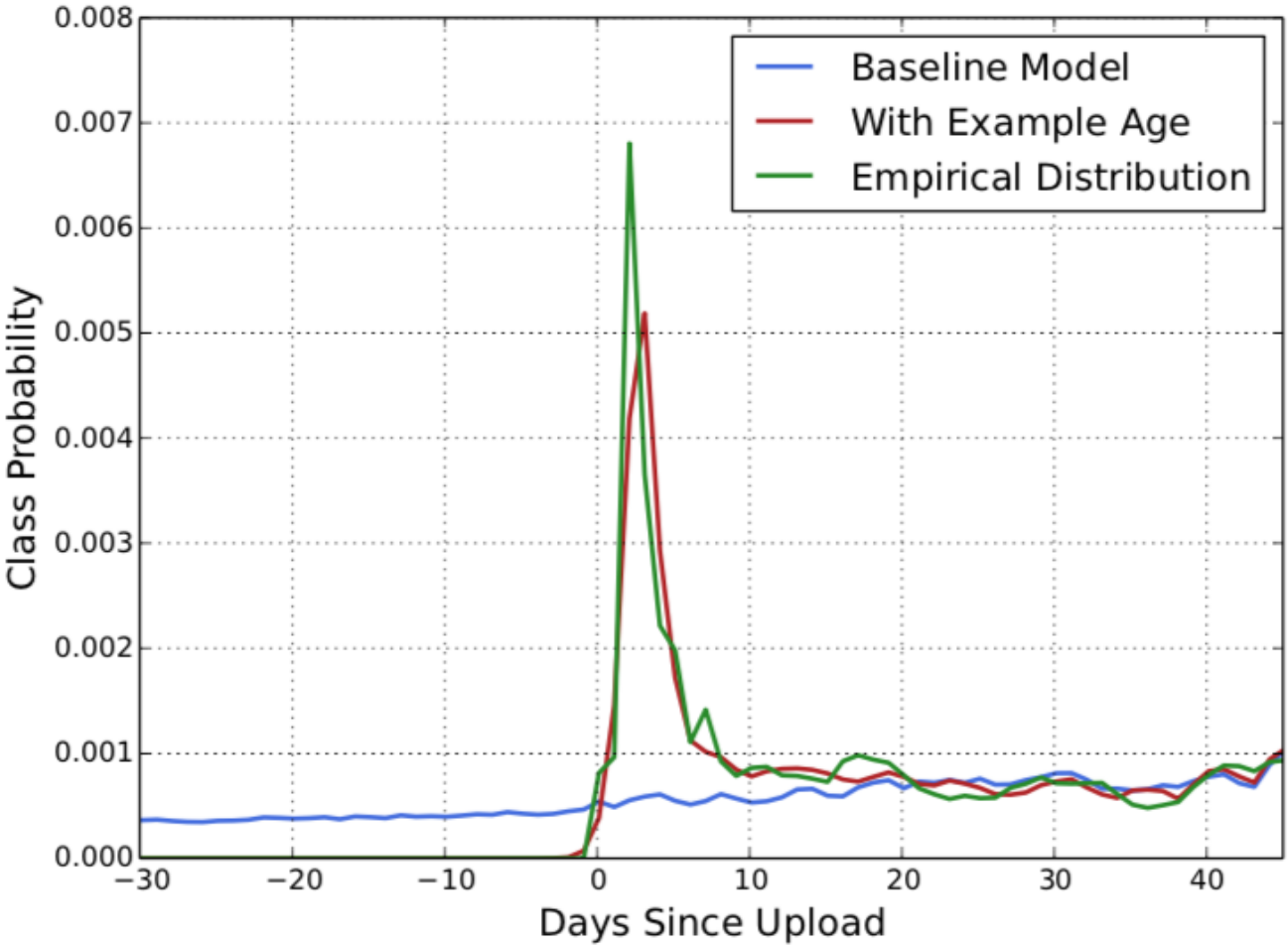
- Too simple?: Maybe, maybe not. Bandit might just be good enough.

Deep Learning

I'm treating deep learning as one big category for tackling the cold start problem, but note that the ways deep learning is used in these papers are very diverse. There are many other recent methods that use deep learning. You might be able to find one that fits your context here[10].

A simple trick used in Deep Learning based Youtube Recommendation

Deep Neural Networks for YouTube Recommendations[11] is not specifically about cold start. Nonetheless, they recognize the cold start problem and proposes a simple hack to deal with it. Namely, among the many features on video watches, search tokens, geographic and biographic information, they included **Days Since Upload**. Training a deep network with this additional feature, they observed that their deep net learns that "fresh" videos are more important.



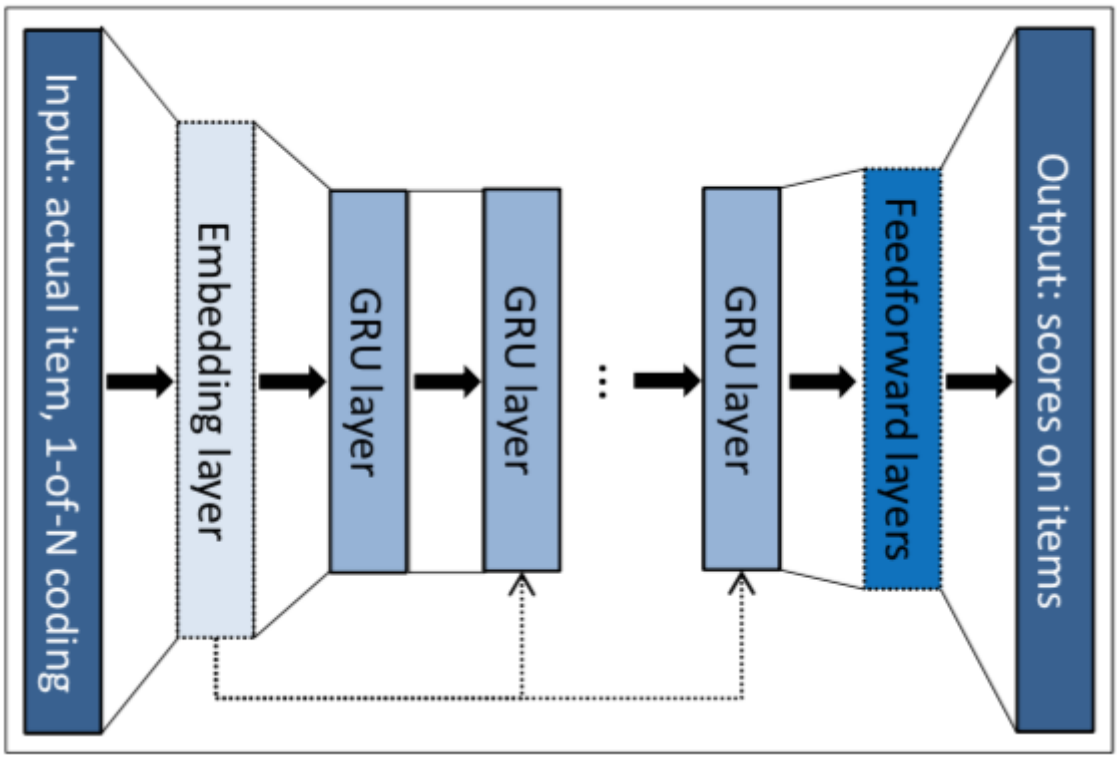
DropoutNet[12]:

The basic idea is simple yet powerful. Their approach is that, in training deep learning based recommendation systems (e.g. collaborative filtering with many layers), we can make it robust against the cold items by randomly dropping ratings of items and users. The key here is that, as

opposed to standard dropout in neural net training, they drop the features, not the nodes. By doing so, they can make the neural net depend less on certain ratings, and be more generalizable to items/users with less ratings. The strength of their approach is that it can be used with any neural net based recommender systems, and also that the approach works for cold start in both users and items.

Session-based RNN[13]:

This approach tries to utilize each session of users by feeding it into an RNN. Specifically, they trained a variant of Gated Recurrent Unit (GRU), where the input is the current state of the session and the output is the item of the next event in the session. This networks is useful e.g. in smaller e-commerce sites where only a few user sessions are available.



Advantages

- If you can find a model for a specific domain that fits with your purpose nicely, you might boost the recommendation performance significantly.

Disadvantages

- Takes time to implement and tune the model.
- Deploying deep models can take more time, depending on your current stack.
- No guarantee on how well it will do.

Conclusion

Assuming you have some recommender system in place in your product, you want to find the solution to cold start that satisfies the following:

1. Suitable for your domain.
2. Can built on top of an existing system, or can be implemented and improved quickly.

For example, you can start from bandit and then move on to deep learning methods. In an industry environment, a model that might look too simple is always better. It is more maintainable, interpretable, and can be improved quickly.

I hope you found some good hints on how to get started on your cold start solution!

1. Wisdom of the Better Few: Cold Start Recommendation via Representative based Rating Elicitation (https://dl.acm.org/citation.cfm?id=2043943) ↩
2. Local Representative-Based Matrix Factorization for Cold-Start Recommendation (https://dl.acm.org/citation.cfm?id=3108148) ↩
3. Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information (https://ieeexplore.ieee.org/document/7355341/) ↩
4. Deepwalk (https://github.com/phanein/deepwalk) ↩
5. Node2Vec (https://snap.stanford.edu/node2vec/) ↩
6. SVDFeature: A Toolkit for Feature-based Collaborative Filtering ↩
7. Personalized recommendation via cross-domain triadic factorization (https://www2013.w3c.br/proceedings/p595.pdf) ↩
8. Finite-time analysis of the multiarmed bandit problem (https://homes.di.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf) ↩ ↩[2] ↩[3]
9. Thompson Sampling for Contextual Bandits with Linear Payoffs (https://proceedings.mlr.press/v28/agrawal13.pdf) ↩
10. RecommenderSystem-Paper (https://github.com/daicoolb/RecommenderSystem-Paper) ↩
11. Deep Neural Networks for YouTube Recommendations (https://ai.google/research/pubs/pub45530) ↩
12. DropoutNet: Addressing Cold Start in Recommender Systems ↩
13. Session-based Recommendations with Recurrent Neural Networks (https://arxiv.org/abs/1511.06939) ↩

Updated: June 05, 2018

LEAVE A COMMENT

PUBLISHED ON SEPTEMBER 26, 2021
IN DEVELOPERS CORNER
(HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Cold-Start Problem in Recommender Systems and its Mitigation Techniques

The recommender systems face a problem in recommending items to users in case there is very little data available related to the user or item. This is called the cold-start problem.

By Vijaysinh Lendave(https://analyticsindiamag.com/author/vijaysinh-lendaveanalyticsindiamag-com/)

The recommender systems (https://analyticsindiamag.com/learn-about-recommender-systems-with-these-8-resources/) face a problem in recommending items to users in case there is very little data available related to the user or item. This is called the cold-start problem. Here in this article, we will discuss the cold-start problems faced by the recommender system with their causes and approaches to overcome this issue. Major points that we will discuss in this article are listed below.

Table of Contents

- Hybrid Approach

4. Deep Learning-Based Mitigation Approaches
    - Dropout-Net
    - Session-Based RNN

Let's start with understanding what actually the cold-start means.

Cold-Start Problems and its Types

Recommender systems are a sort of information filtering technology that aims to offer information items that are likely to be of interest to the user. The cold start problem occurs when the system is unable to form any relation between users and items for which it has insufficient data. There are two types of cold-start problems: –

1. **User cold-start problems:** When there is almost no information available about the user, the user cold-start problem arises.
2. **Product cold-start problems:** When there is almost no information about the product, the product cold-start problem arises.

Reasons for Cold-Start Problem

Commonly there are three main reasons for which the system fails to give recommendations namely Systematic Bootstrapping, Low Interaction (Few instances available), and the entry of new users.

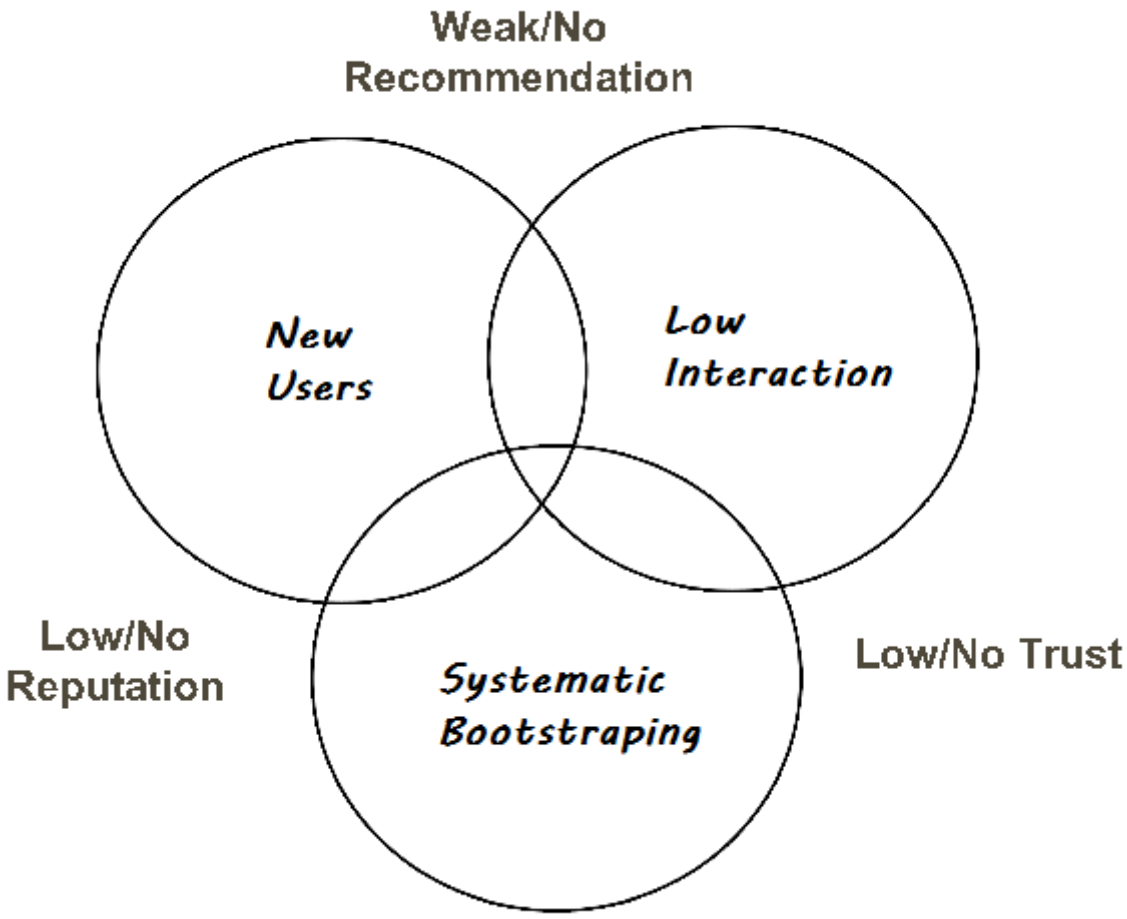


Image Source (https://www.researchgate.net/figure/Beyond-the-cold-start-problem-new-items-may-fail-in-trust-if-new-users-with-low_fig1_263575914)

Systematic Bootstrapping

Systemic bootstrapping refers to the starting of the system when there is almost no information on which the recommender may rely. As all assets and users are new, this example demonstrates the drawbacks of both the minimal interaction and the New user cases. As a result, several of the approaches established to cope with those two scenarios are unsuitable for system bootstrapping.

Low Interaction

When things are added to the catalogue, the item cold-start problem occurs when they have no or very few interactions. This is particularly problematic for collaborative filtering algorithms, which generate recommendations based on the item's interactions. A pure collaborative algorithm cannot recommend an item if there are no interactions available.

If only a few interactions are available, a collaborative algorithm can recommend it, but the quality of those recommendations will be poor. This raises a new issue, one that is not tied to new products, but rather to unpopular items.

New User

The new user case occurs when a new user enrols in the system and the recommender is required to offer recommendations for a set length of time without depending on the user's previous interactions, as none have been recorded yet.

This issue is especially important when the recommender is part of the service provided to users, because a user who receives poor-quality recommendations may opt to left the system before giving enough interaction to allow the recommender to understand his or her interests.

When dealing with new users, the primary technique is to ask them to offer certain preferences in order to create an initial user profile. You may have seen such interaction on Netflix.

Mitigation Techniques

Many solutions to minimize the cold-start problem have been created due to a large number of recommender algorithms available, as well as system type and features. The basic strategy is to use hybrid recommenders to offset the drawbacks of one category or model by combining them with another.

All three types of cold-start problems we have discussed so far have a lack of user engagements in common, and there are some similarities in the techniques available to solve them. Let's see them one by one;

Representative Approach

If we lack sufficient knowledge of users and objects, we can rely on those who represent the set of items and users. That is the underlying idea of representative-based approaches. Representatives can be users whose linear combinations of preferences closely resemble those of other users.

When a new user joins the site, we may ask them to rate a few contents, and then use that information to deduce the ratings of other products. In this manner, we can enhance suggestions for new users by charging consumers a modest fee to rate select goods.

Feature Mapping

The matrix factorization technique may be used to solve feature mapping. The basic concept is that a matrix factorization model depicts user-item interactions as the product of two rectangular matrices whose content is learnt using known interactions via machine learning. Each user will be assigned a row in the first matrix, and each item will be assigned a column in the second matrix; this set is called a latent factor. Latent factors are rows or columns that are linked with a certain individual or object.

When a new item is added, there are no related latent factors, and the lack of interactions prevents them from being learned, as was done with previous things. If each item is linked with certain attributes (for example, author, year, publication, actors), an embedding function may be defined that, given the item features, estimates the corresponding item latent factors. The embedding function may be built in a variety of ways, and it is trained using data from warm things that are already accessible. By the feature mapping, we can have a reasonable recommendation

Hybrid Approach

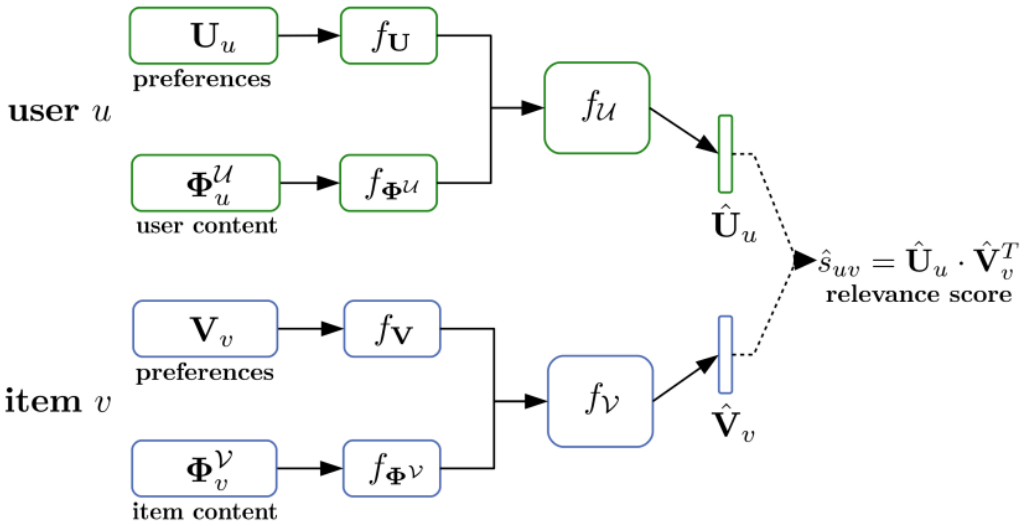
Another new technique that is related to feature weighting is the development of a hybrid content-based filtering recommender in which characteristics, either of the items or of the people, are weighted depending on the user's sense of significance.

Different criteria (such as the actors, director, region, and title) will be weighted differently in order to select a movie that the consumer would enjoy.

Deep Learning-Based Mitigation Approaches

Dropout-Net

The basic concept is simple but effective. Their idea is that by randomly deleting ratings of things and users when training deep learning-based recommendation systems (e.g., collaborative filtering with multiple layers), we may make it resilient against cold items.

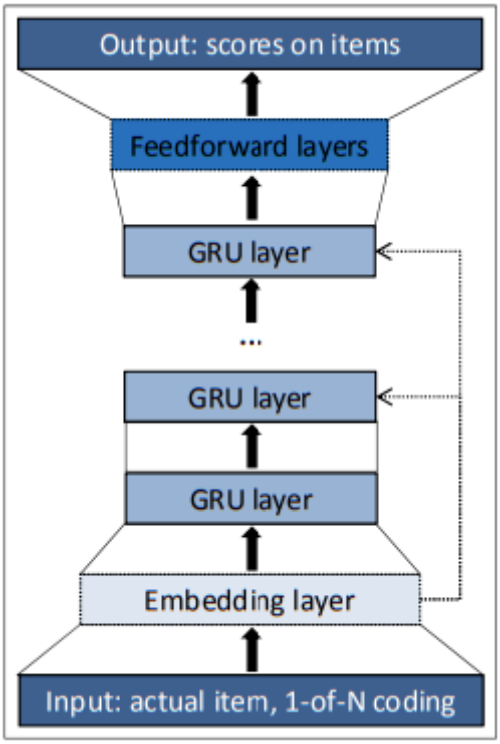


DropoutNet architecture diagram. For each user u, the preference $\mathbf{U}_u$ and content $\mathbf{\Phi}_u^{\mathcal{U}}$ inputs are first passed through the corresponding DNNs $f_{\mathbf{U}}$ and $f_{\mathbf{\Phi}^{\mathcal{U}}}$. Top layer activations are then concatenated together and passed to the fine-tuning network $f_{\mathcal{U}}$ which outputs the latent representation $\hat{\mathbf{U}}_u$. Items are handled in a similar fashion with $f_{\mathbf{V}}$, $f_{\mathbf{\Phi}^{\mathcal{V}}}$ and $f_{\mathcal{V}}$ to produce $\hat{\mathbf{V}}_v$. All components are optimized jointly with back-propagation and then kept fixed during inference. Retrieval is done in the new latent space using $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ that replace the original representations $\mathbf{U}$ and $\mathbf{V}$.

The above information is taken from the original research paper of Volkovs et. al (https://proceedings.neurips.cc/paper/2017/file/dbd22ba3bd0df8f385bdac3e9f8be207-Paper.pdf). The essential point here is that, unlike normal dropouts in neural network training, they drop the features rather than the nodes. By doing so, they may reduce the neural network's reliance on specific ratings and make it more generalizable to items/users with fewer ratings. Their approach's strength is that it can be utilized with any neural network-based recommender system, and it also works for a cold start.

Session-Based RNN

This method attempts to make use of each user session by feeding it into an RNN. They trained a version of the Gated Recurrent Unit (GRU) with the input being the current state of the session and the output being the item of the next event in the session.



The architecture of Session-Based RNN (https://arxiv.org/pdf/1511.06939.pdf)

The GRU layer(s) form the network's core, and between the last layer and the output, additional feed-forward layers can be added. The projected preference of the items, i.e. the likelihood of being the next item in the session for each item, is the output. When using several GRU layers, the preceding layer's hidden state becomes the input for the next. Optionally, the input can be linked to GRU layers located farther down the network. Take a look at the entire architecture, which illustrates the representation of a single event inside a timeline of occurrences.

This network is helpful, for example, in smaller e-commerce sites with a limited number of user sessions.

Conclusion

Through this post, we could understand the cold-start problem in the recommender system and its major causes. Mitigation techniques discussed here can help you to overcome this issue out of which hybrid approach can make your system robust because it considers many aspects of the product.

References

- Cold-Start (https://en.wikipedia.org/wiki/Cold_start_(recommender_systems)#cite_note-Koren09-25)
- Tackling the Cold Start Problem in Recommender Systems (https://kojinoshiba.com/recsys-cold-start/)
- Dropout-Net (https://proceedings.neurips.cc/paper/2017/file/dbd22ba3bd0df8f385bdac3e9f8be207-Paper.pdf)
- Session-Based RNN (https://arxiv.org/pdf/1511.06939.pdf)

More Great AIM Stories

All Major NVIDIA Announcements At Computex 2021 (https://analyticsindiamag.com/all-major-nvidia-announcements-at-computex-2021/)

DataOps Goes Mainstream As Atlan Lands Big (https://analyticsindiamag.com/dataops-goes-mainstream-as-atlan-lands-big/)

Stanford University Professor Maneesh Agrawala On Video Editing Tools, Deep Fakes & More (https://analyticsindiamag.com/stanford-university-professor-maneesh-agrawala-on-video-editing-tools-deep-fakes-more/)

Data Mesh: Moving Away From Monolithic & Centralised Data Lakes (https://analyticsindiamag.com/data-mesh-moving-away-from-monolithic-centralised-data-lakes/)

A Beginner's Guide To Intel oneAPI (https://analyticsindiamag.com/a-beginners-guide-to-intel-oneapi/)

An AI Tool To Assess Severity Of COVID-19 Cases (https://analyticsindiamag.com/an-ai-tool-to-assess-severity-of-covid-19-cases/)

(https://analyticsindiamag.com/author/vijaysinh-lendaveanalyticsindiamag-com/)

Vijaysinh is an enthusiast in machine learning and deep learning. He is skilled in ML algorithms, data manipulation, handling and visualization, model building.

Our Upcoming Events

Conference, in-person (Bangalore)

MachineCon 2022

24th Jun

Register (https://machinecon.analyticsindiamag.com/tickets/)

Conference, Virtual

Deep Learning DevCon 2022

30th Jul

Register (https://dldc.adasci.org/get-the-tickets/)

Conference, in-person (Bangalore)

Cypher 2022

21-23rd Sep

Register (https://www.analyticsindiasummit.com/cypher-2022/register/)

3 Ways to Join our Community

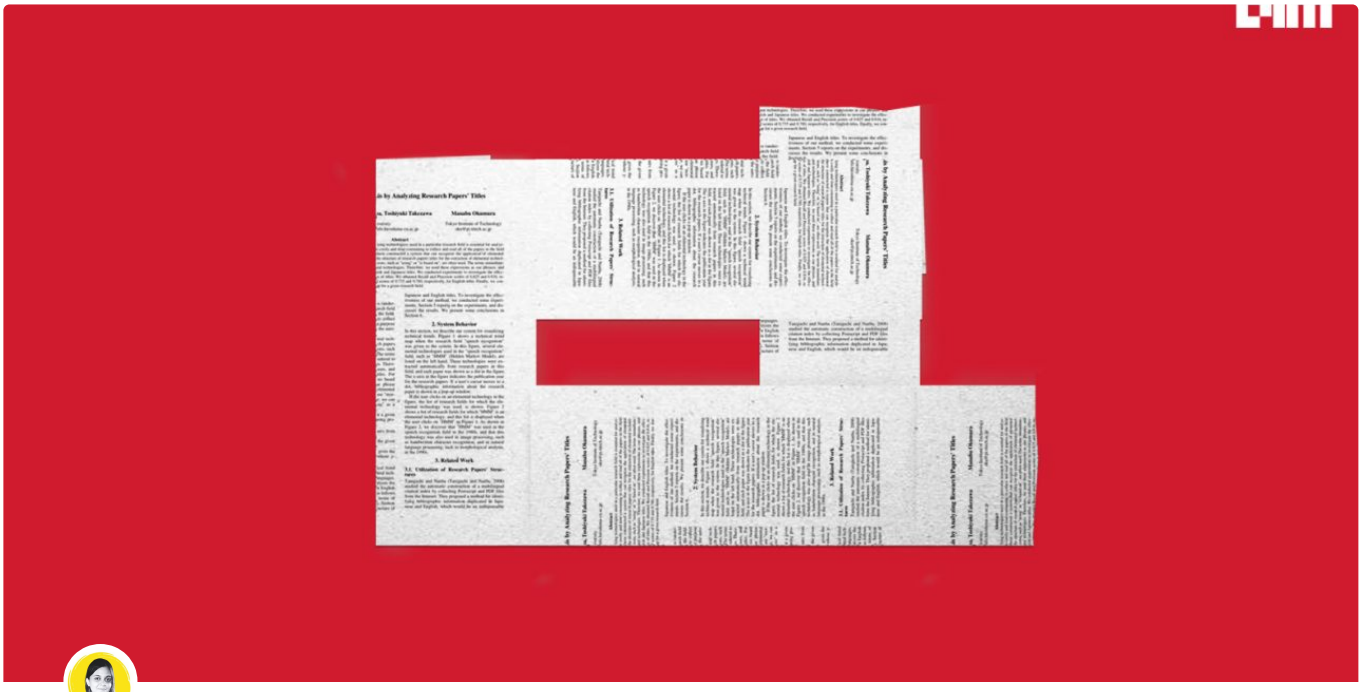
Discord Server

Stay Connected with a larger ecosystem of data science and ML Professionals

MORE FROM AIM

Alternative data analytics startup Synaptic raises USD 20 Mn in Series B (https://analyticsindiamag.com/alternative-data-analytics-startup-synaptic-raises-usd-20-mn-in-series-b/)

The funding is a significant step forward to harness the full potential of merging ML and analytics with alternative data to improve investing decisions.



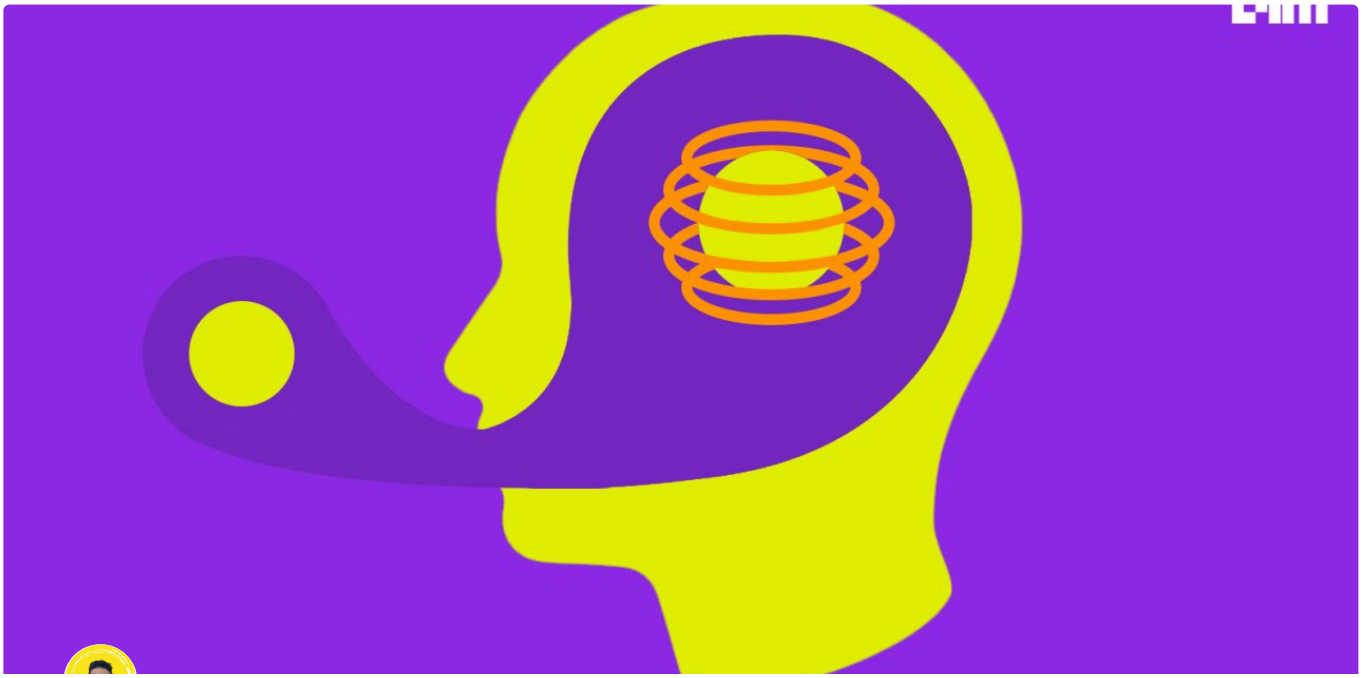
Inside ACL 2022 Test of Time Papers Awards (https://analyticsindiamag.com/inside-acl-2022-test-of-time

(https://analyticsindiamag.com/inside-acl-2022-test-of-time-papers-awards/)

The ACL Test-of-Time Paper Award recognises up to four papers for their long-lasting impact in the field of Natural Language Processing and Computational Linguistics.

Marathon: An unusual marketing strategy for TCS (https://analyticsindiamag.com/marathon-an-unusual-marketing-strategy-for-tcs/)

TCS' association with marathons began in 2008 when they became the junior sponsors for the Mumbai Marathon.



NLP gets a quantum boost (https://analyticsindiamag.com/nlp-gets-a-quantum-boost/)

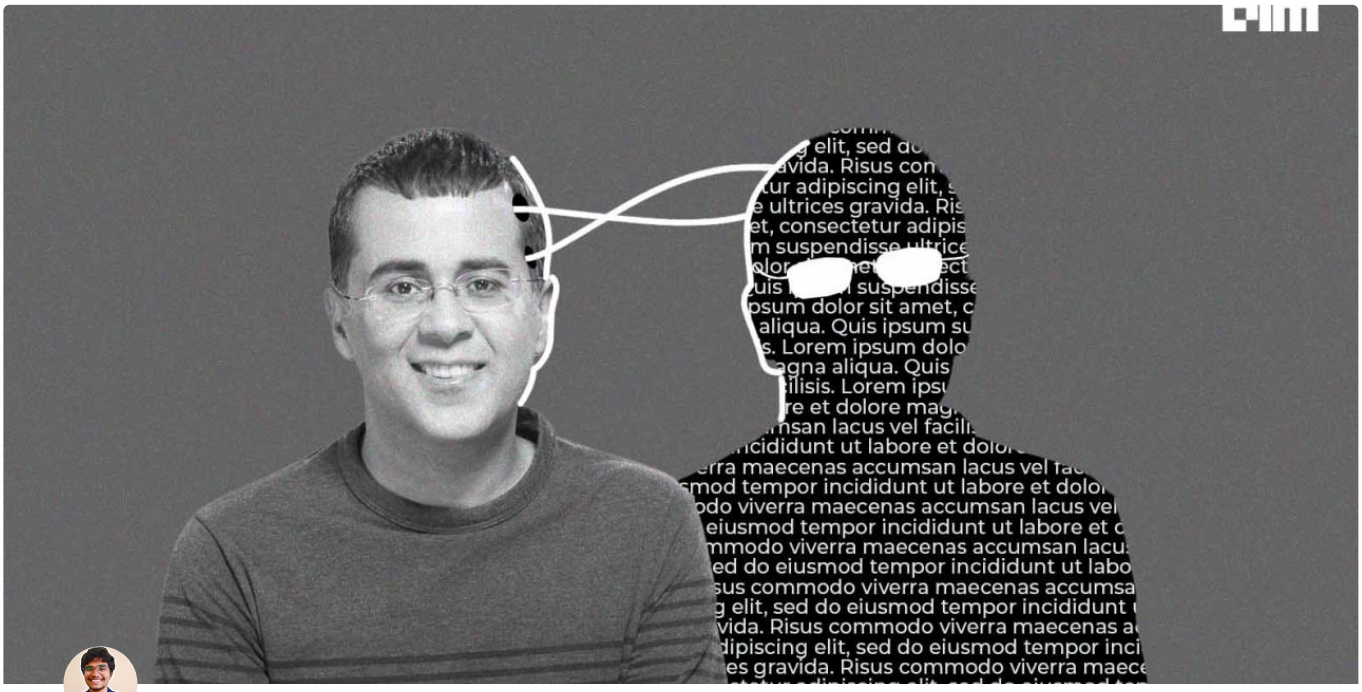
QSANN is effective and scalable on larger data sets and can be deployed on near-term quantum devices.

IIT Roorkee partners with Deloitte to bridge AI skill gap in India (https://analyticsindiamag.com/iit-roorkee-partners-with-deloitte-to-bridge-ai-skill-gap-in-india/)

This partnership has the potential to strengthen the AI roadmap of India.

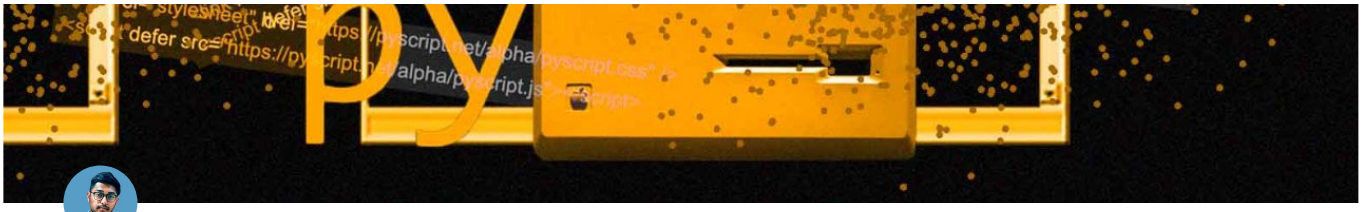
Key announcements from NVIDIA at ISC 2022 (https://analyticsindiamag.com/key-announcements-from-nvidia-at-isc-2022/)

Venado will be the first system in the US to feature a mix of Grace CPU Superchip nodes and Grace Hopper Superchip nodes.



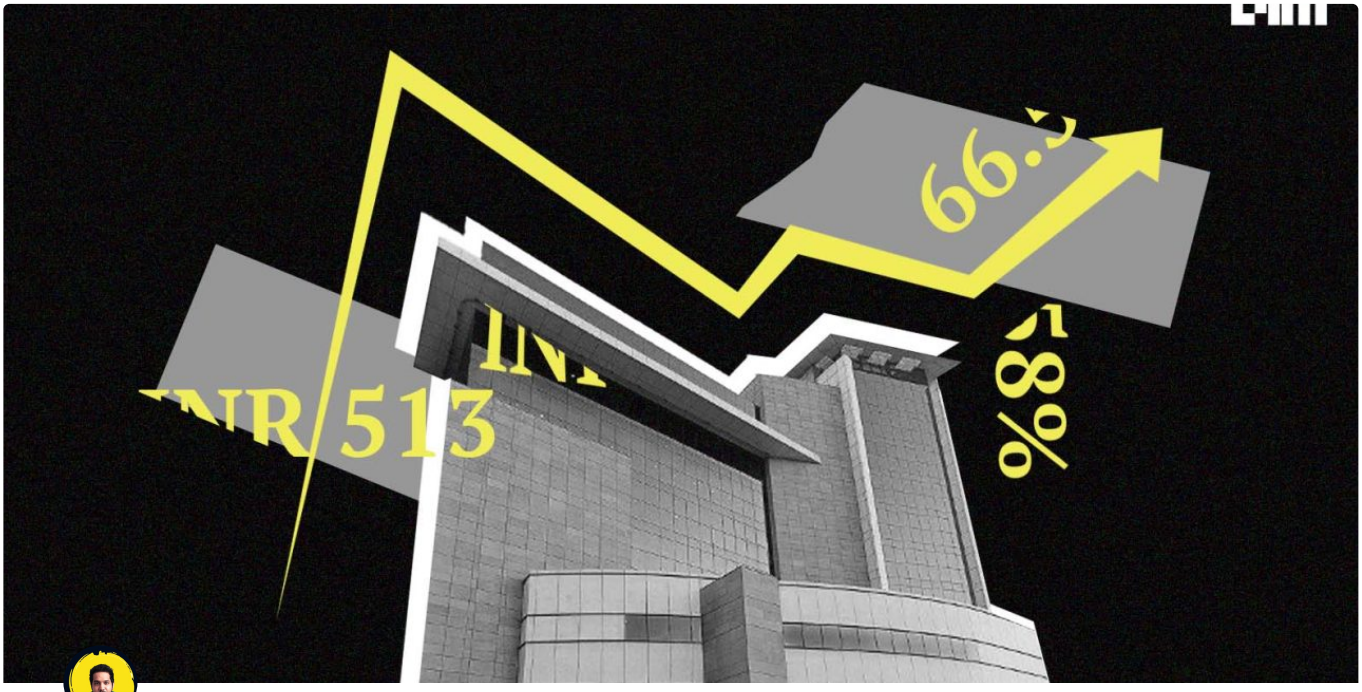
How to build an AI model that writes like Chetan Bhagat (https://analyticsindiamag.com/how-to-build-an-ai-model-that-writes-like-chetan-bhagat/)

I used TextBlob for sentiment analysis.



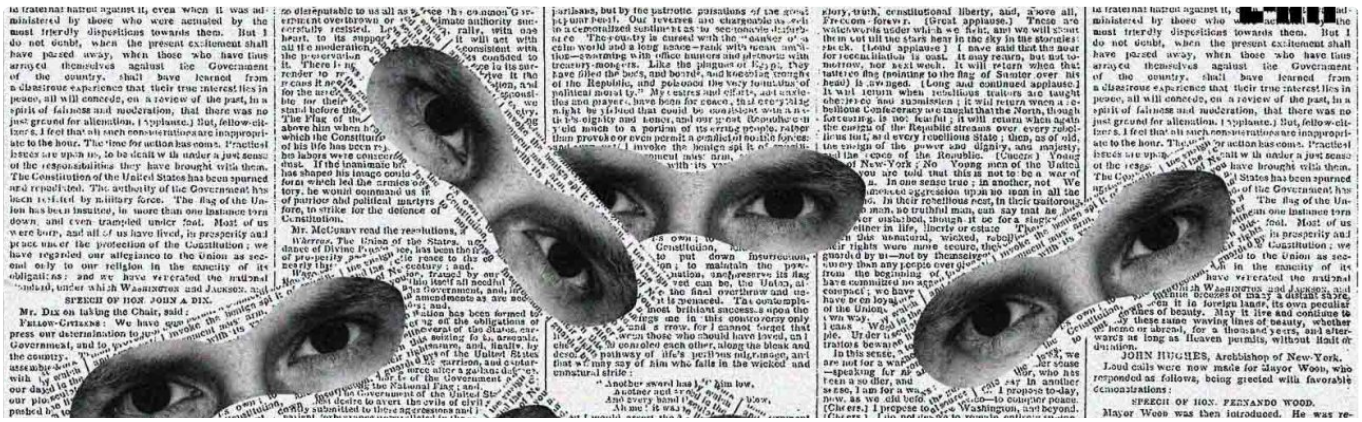
Run your first data analysis program in the browser with PyScript (https://analyticsindiamag.com/run-your-first-data-analysis-program-in-the-browser-with-pyscript/)

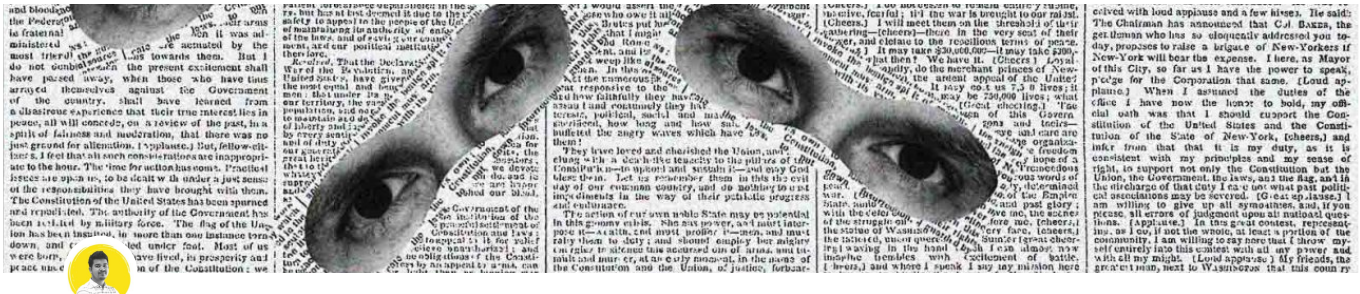
PyScript allows users to build Python applications on the web using an HTML interface.



Why Info Edge succeeds (https://analyticsindiamag.com/why-info-edge-succeeds/)

The numbers indicate that the IT hiring is going through the roof and is providing additional revenues for the platform.





‘I don’t really trust papers out of top AI labs anymore’ (https://analyticsindiamag.com/i-dont-really-trust-papers-out-of-top-ai-labs-anymore/)

Findings that can’t be replicated are intrinsically less reliable.

(https://analyticsindiamag.com)

(https://www.linkedin.com/company/analytics-india-magazine/)

About Us

Advertise

Weekly Newsletter

Write for us

Careers

Contact Us

RANKINGS & LISTS

Academic Rankings

Best Firms To Work For

PeMa Quadrant

OUR CONFERENCES

Cypher

The MachineCon

Machine Learning Developers Summit

The Rising

Data Engineering Summit

OUR BRANDS

MachineHack

AIM Recruits

AIM Leaders Council

expressanalytics
Innovative Data Solutions

Machine Learning Solutions For Cold Start Problem In Recommender Systems

AI in Marketing Reading Time 8 mins Express Analytics | 30 Jul 2021

Machine Learning Solutions For Cold Start Problem In Recommender Systems

Study the past to know the future – Confucius. While building product recommenders, the cold start is one of the biggest challenges that developers face, in addition to others such as data scarcity, scalability, and diversity. Know more about Machine Learning Solutions for Cold Start Problem in Recommender Systems.

Synonymous with the failure of a car engine to turn when the weather is cold, the cold start problem confounds recommenders across two categories – product and user. When a new product or service hits the market, it has no likes or dislikes, reviews, or clicks, thus making it difficult to recommend. It's the same when a new user signs up – since the concerned system has no information about them or their preferences, it can't recommend a product to them. It's like a clean slate, so the opportunity of studying the past to forecast the future simply does not exist.

Want to use our recommender system to improve conversions? Get in touch with our experts.

One of the most popular methods used by classic recommender systems is collaborative filtering (CF). The latter presumes that the user or the item has some ratings, from which it can then infer the ratings of similar users or items. But for

Recent Posts

Technology Advancements For Clienteling In Retail ...

Varun Madhok May 27, 2022

and users.

Table of Contents

So, collaborative filtering cannot be used for cold starts because of its very nature of recommending each item (products advertised on your site) based on user actions. A CF model throws up more and more items with an increasing degree of accuracy as a user continues to act on the website or app, over a period of time. But at the start line, i.e. cold start, since the user (or the product) is new there's no track record to fall back on.

In today's world of commercial artificial intelligence (AI), the cold start is also where the solution can now include a degree of automated data modeling.

In their research paper, "Treating Cold Start in Product Search by Priors", researchers Parth Gupta, Tommaso Dreossi, Jan Bakus, Yu-Hsiang Lin, Vamsi Salaka, explain the problem explicitly:

Learning to Rank (LTR) models rely on several features to rank documents for a given query. Many LTR features are based on users' interactions with documents such as impressions, clicks, and purchases. We call these features behavioral features. Ranking models are trained to optimize user engagement, and therefore, such behavioral features tend to be the most important training signals.

However, new and tail products that do not have user engagement lack behavioral features and hence are ranked as irrelevant, which in turn further excludes them from catching user engagement. It takes time for them to gather enough behavioral signals to show up at their fair ranking position. This leads to the causality dilemma: No behavioral data causes poor ranking which in turn results in new products having a reduced likelihood of accruing behavioral data.

This phenomenon is referred to as a cold start problem and poses serious concerns, from bad customer experience to lost revenue opportunities.

Over the years, recommender systems have used different solutions to "kick start" their recommendations from a cold start. The early day's recommenders drew on the methods and theories taken from other artificial intelligence (AI) fields for user profiling and preference discovery. But, of late, with the commercial deployment of AI, we have seen an increase in the success of AI apps.

Solutions For Cold Start Problem

How To Integrate Lightspeed Retail POS With CDP

Varun Madhok May 19, 2022

Point of Sale Analytics: How to Use Data To Boost …

Express Analytics May 6, 2022

combination of data analytics and AI to produce advanced insights into the relationships between users and items.

In ML, for supervised learning, the access to, and the application of labeled data based on things past is necessary for the labeling of previously unseen data, for things in the future.

The same is true in the case of unsupervised machine learning, where patterns are being characterized because there can be no assumption made in relation to the existence of labeled training data.

What Is Cold-start Problem In Business?

REQUEST A DEMO

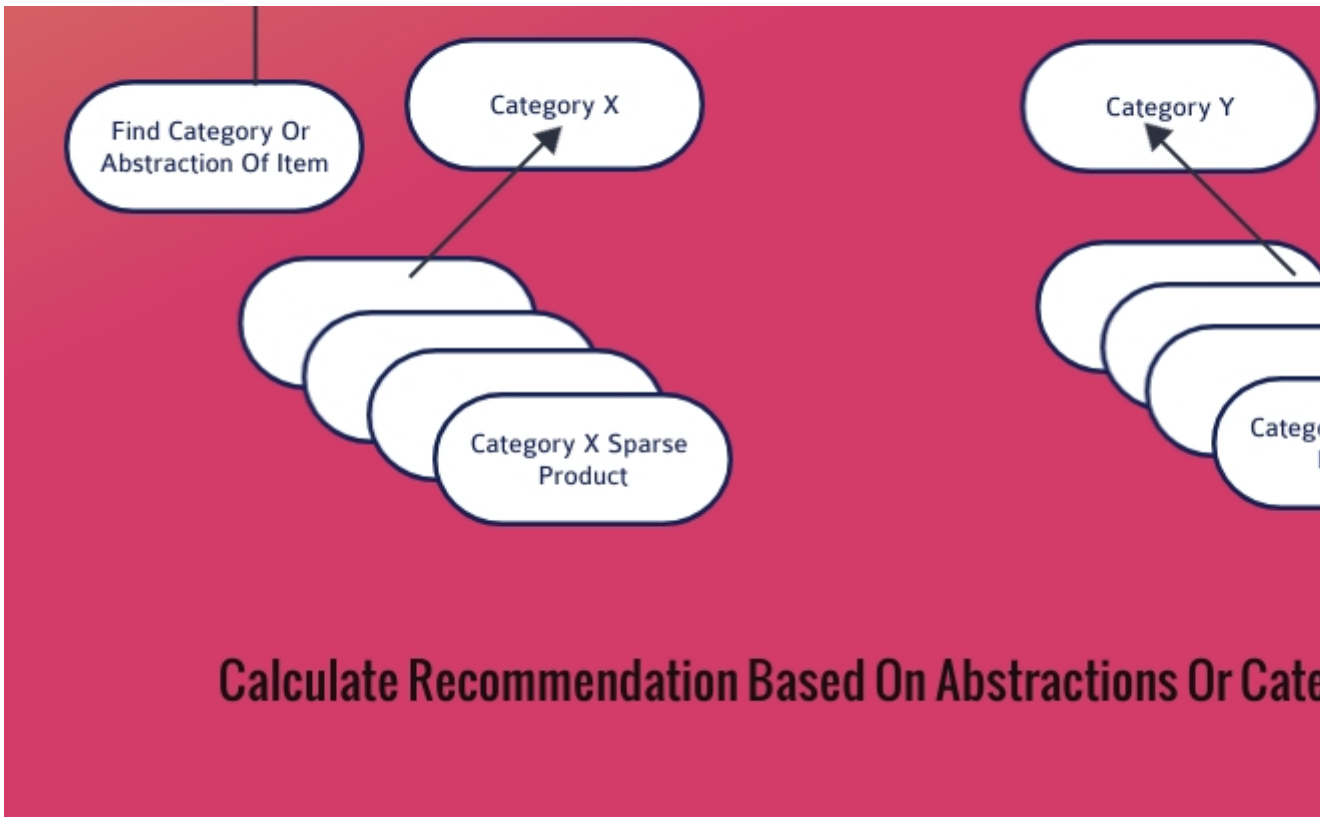
So, then, the obvious question – how does one go ahead in tagging labels to future data when there are no labels (diagnoses, classes, known outcomes) in the past data?

Here are some of the solutions that combine algorithms and the techniques of ML to produce a neutral judgment in model development and progression, which eventually leads towards optimization:

Content-based filtering: This seems to be among the favorite options used by ML developers in cold start cases. The product recommender can utilize metadata about the new product when creating recommendations. What's more, any form of additional information such as that obtained about the user from their social media networks (provided at the time of signing up with social logins) can also be utilized to tackle the initial information scarcity. Also, here, filtering algorithms are given the predisposition for specific items, and so recommend similar items based on a domain-specific concept of item content.

There are many advantages to a content-based recommender system. To begin with, this type of recommendation is based on item representation, making it independent of the user. Because of this, the issue of data sparsity does not arise. Next, content-based recommender systems are able to suggest new items to users, which resolves the new product cold start problem.

Popularity-based model: One more go-to option used frequently for cold start cases. Using Python to build the model, a new customer, at the very start of a customer journey, can be shown a list of trending (popular) products. Maybe, if there's a product on the list which almost all new customers buy, that, too, can be suggested to all new customers. Following this, each choice can be registered, so also the contextual information, i.e. location using the device's GPS coordinates, the channel the visitor came from, the device used, and so on. After the first few clicks, behavioral science will help the e-commerce site build a customer profile. to build up from there.



For a first-time product about to be listed, it has no previous “baggage”, i.e. the purchase history, and so on. Till enough purchases or likes are not gathered around this one product, eCommerce sites like Amazon and even YouTube continue to promote it frequently, in an almost in-your-face campaign. What is also done simultaneously is to display similar products, using either string similarity algorithms like Levenstein distance or Hammind distance, till enough users have either bought the product or perhaps rated the new service. There you have it, then, data around a product that proves its popularity or otherwise.

One drawback this model does have is a lack of personalization, but then again, we are talking of tackling the cold start problem, and not after the engine has been warmed and is running smoothly, figuratively speaking.

The multi-armed bandit model: The inspiration for this is drawn from a multi-lever casino slot machine. The gambler has the option of pulling not one but many levers, so he thinks his chances at winning have gone up. (Incidentally, the single lever slot machine is called the single-armed bandit). The probability ratio of the reward to each lever is different, though. The ratios are not known to the gambler (he is taking his chances) but known to the casino, which increases its profits eventually, because the House never loses (hence, the use of the term, “bandit’).

So, what relevance does this have to an algorithm trying to circumvent the cold start problem?

On a daily basis, there are scores of new items that arrive daily on a website or app. These items are like the various slot machines, single as well as multi-lever, and each comes with a different RoI. But because these are fresh items, the e-commerce site really does not know at that point in time how many users will buy them. In a multi-arm slot machine, each arm represents a reward. So recommending a subset of the new products is like deciding to pull a subset of slot machines to draw. The trick for the site is to identify obviously which lever to pull to eke out the maximum return.

There’s an inherent problem, though in this kind of machine learning model. It’s referred to as the “exploration vs exploitation” problem. In layman’s terms, if a particular new product is flying off the shelves, it is but natural for the seller to get greedy and show it to many more users (called exploitation). But at the same time, the remaining new products may be languishing, and the seller needs to also

Clearly, a balance has to be struck between exploration and exploitation. There exist various bandit algorithms for making optimal recommendations for the user. The popular MAB algorithms include Epsilon Greedy, Decayed Epsilon Greedy, Upper Confidence Bound, Thompson Sampling, etc. Each of these can be coded in Python. Eventually, the idea is to use any of these models so that the bandit algorithm will help the seller choose scientifically between exploiting the one item that gave it the highest reward and exploring the other products.

The seller has to learn to balance the reward maximization based on the information already obtained against implementing new actions to increase the database. In ML, this is called the exploitation vs. exploration tradeoff.

Deep Learning Approach: With the recent advent of deep learning, a sub-discipline of machine learning that mimics the human brain, there are new attempts at resolving the cold start problem or issue using this approach. There are many models and research papers out there that suggest the use of deep neural networks as an option to mitigate the cold start problem in recommenders. Using neural networks, the initial weights/ratings on the network edges are assigned in a random manner but backpropagation is used to repeat the model to the optimal.

Meta-learning approaches have gained popularity in machine learning for learning representations useful for a wide range of tasks, according to this research. In his paper, “Meta-Learning for User Cold-Start Recommendation”, Homanga Bharadwaj, Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, India, says they designed a recommendation framework that was trained to be “reasonably good enough” for a wide range of users. It was inspired by the generalizable modeling prowess of model-agnostic meta-learning. While testing, to adapt to a specific user, the model parameters were updated by a few gradient steps, and then the model was evaluated on three different benchmark datasets. The paper, using detailed simulation studies, shows that the framework was able to handle the user cold-start model much better than state-of-the-art benchmark recommender systems.

Another research paper in the same field by a team of researchers from the University of KwaZulu Natal, Westville, South Africa attempted a new approach to solve the cold start problem by using social networks and Matrix Factorization to enhance a deep learning approach. The research team explained that the social information was used to form groups of users since users within a given community were likely to have the same interests. For this, a community detection algorithm was used. Once they were so segregated, a deep learning model was trained on each community. The comparative models were then evaluated, and the metrics used were Mean Squared Error (MSE) and Mean Absolute Error (MAE). The evaluation was carried out using 5-fold cross-validation. The results showed that the use of social information improved on the results achieved from the Deep Learning Approach, and grouping users into communities was advantageous.

In conclusion: The problem of cold start in a recommender system has been around for years but with the advent of artificial intelligence coupled with data analytics, much progress has been made, and today, there are quite a few solutions around to overcome it.

Leave a comment

Your email address will not be published. Required fields are marked *

Comment *

Name * Email *

Website

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

expressanalytics
Innovative Data Solutions

Guide

Video

Subscribe to our blogs

Click below to subscribe to our newsletter

SUBSCRIBE

About

Leadership

Careers

Media

Offices

Our Partners

Connect with us

Email : info@expressanalytics.net

Phone : +1 (949) 754 3180

Animesh Goyal Follow

May 22, 2019 · 11 min read ★ · Listen

Save

Solving Cold User problem for Recommendation system using Multi-Armed Bandit

This article is a comprehensive overview of using Multi-Armed Bandit to recommend a movie to a new user

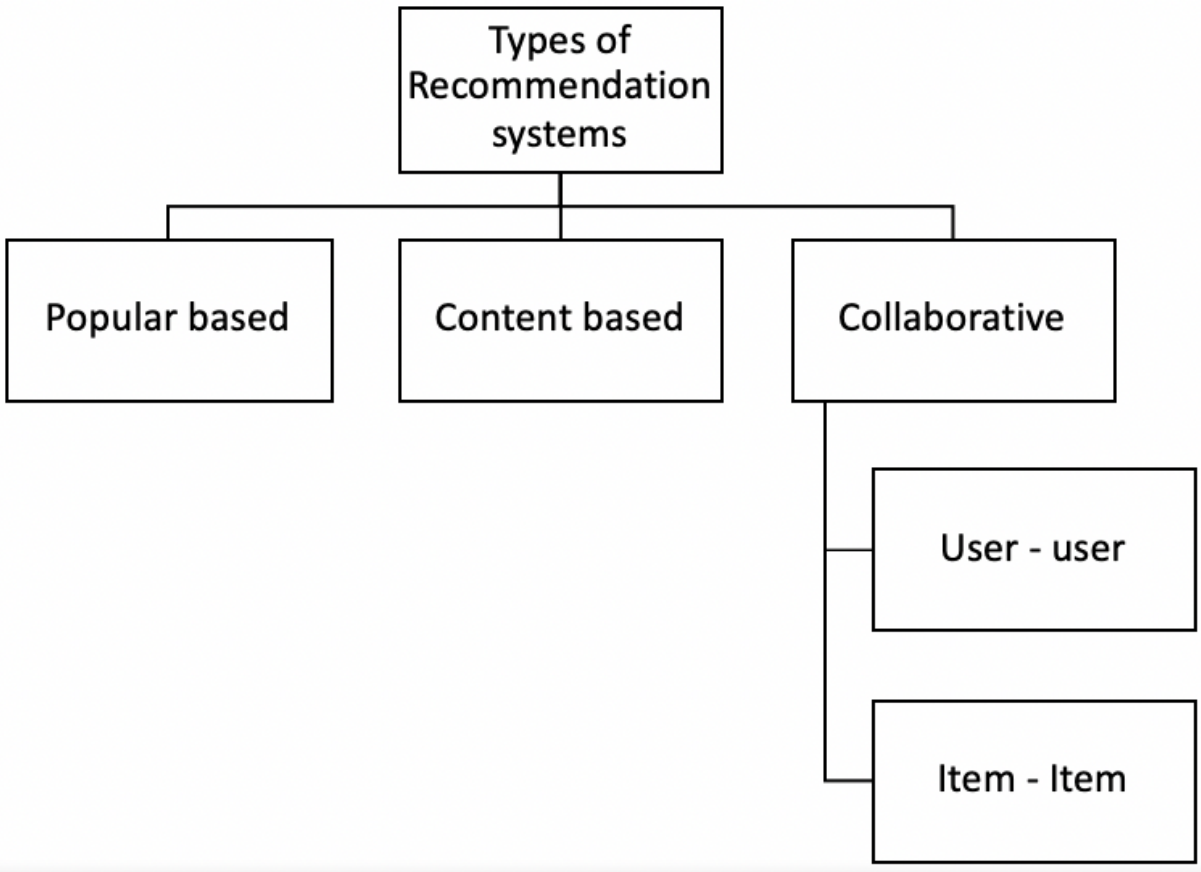
Umm not the cold user we are referring to

How often you feel after a hectic day at work that what should I watch next? As for me — yes, and more than once. From Netflix to Prime Video, the need to build robust movie recommendation systems is extremely important given the huge demand for personalized content of modern consumers.

Once at home, sitting in front of the TV seems like a fruitless exercise with no control and no remembrance of content that we consumed. We tend to prefer an intelligent platform which understands our tastes and preferences and not just run on autopilot.

In this article, I have tried to build a recommender system that would suggest the best movie to you in the shortest span of time. This recommender system can also be utilized in the recommendation of a broad range of items. For instance, it can be used to recommend products, videos, music, books, news, Facebook friends, clothes, Twitter pages, Android/ios apps, hotels, restaurants, routes, etc.

Types of Recommendation system

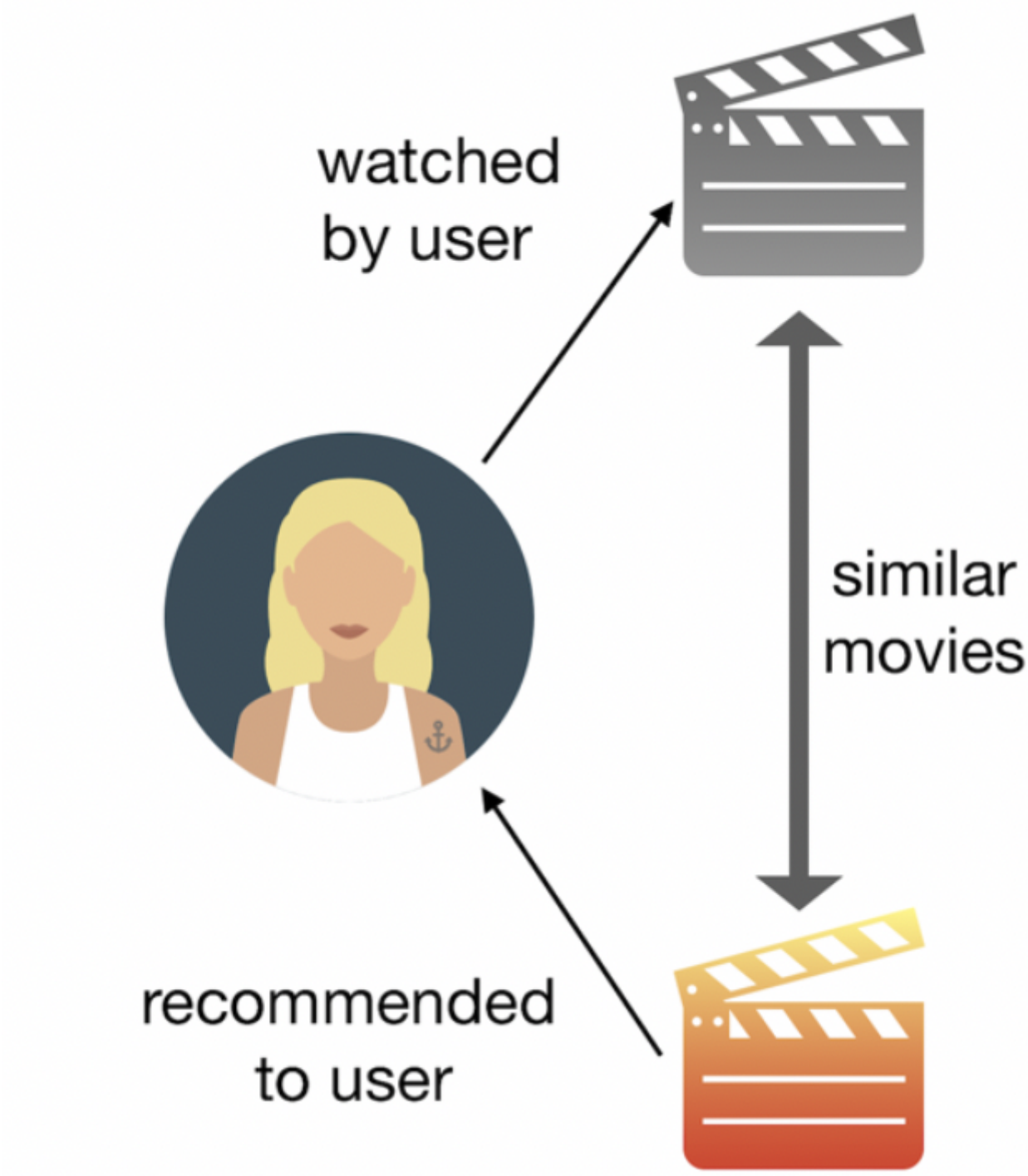


1. Popularity based recommendation system

As the name suggests the Popularity based recommendation system works with the trend. It basically uses the items which are in trend right now. For example, if any product is usually bought by every new user then there are chances that it may suggest that item to the user who just signed up.

The problem with the popularity-based recommendation system is that personalization is not available with this method i.e. even though you know the behavior of the user you cannot recommend items accordingly.

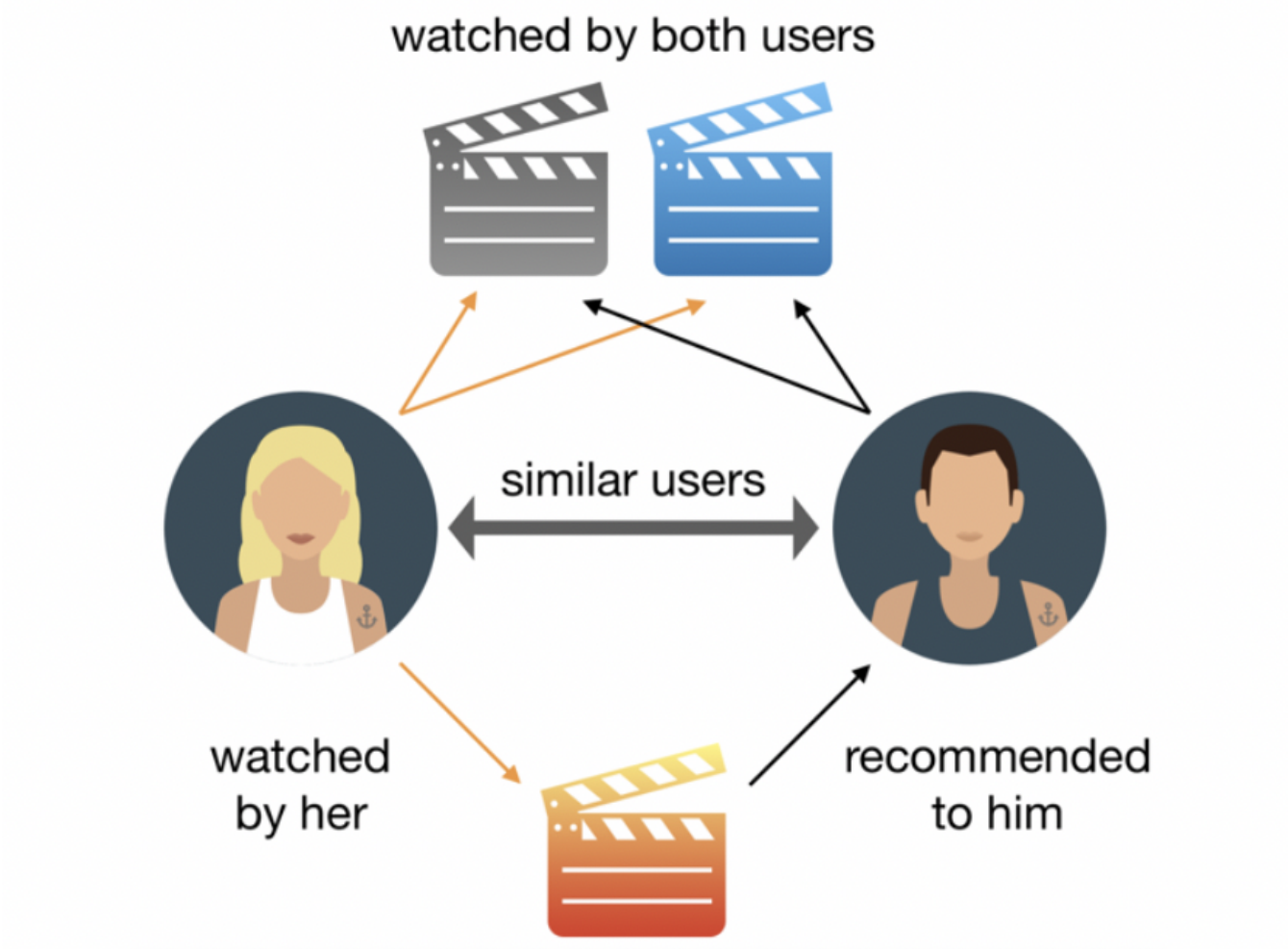
2. Content-based recommendation system



Content-based recommendation system

Content-based filtering is using the technique to analyze a set of documents and descriptions of items previously rated by a user, and then build a profile or model of the user's interests based on the features of those rated items. Using the profile, the recommender system can filter out the suggestions that would fit for the user.

3. Collaborative based recommendation system



Collaborative based recommendation system

The key idea behind the Collaborative based recommendation system is that similar users share the same interest and that similar items are liked by a user. There are two types of Collaborative based recommendation systems: User-based and Item-based. We will be using a User-based filtering process.

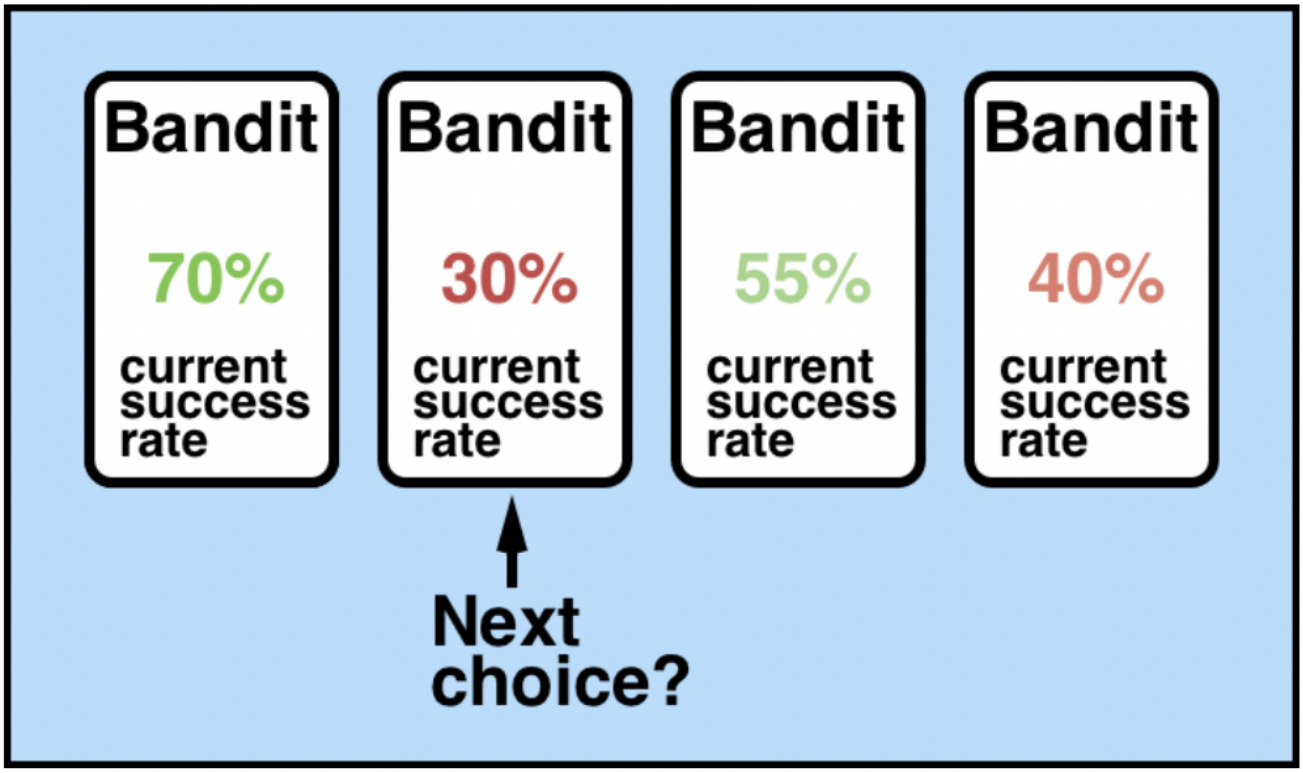
But these are unable to tackle the problem of Cold user.

The Problem — Cold Start

So, what is the cold start problem? The term derives from cars. When it's really cold, the engine has problems with starting up, but once it reaches its optimal operating temperature, it will run smoothly. With recommendation engines, the "cold start" simply means that the circumstances are not yet optimal for the engine to provide the

To solve this problem, we are introducing the concept of using Multi-Armed bandit

Multi-Armed Bandit (MAB)

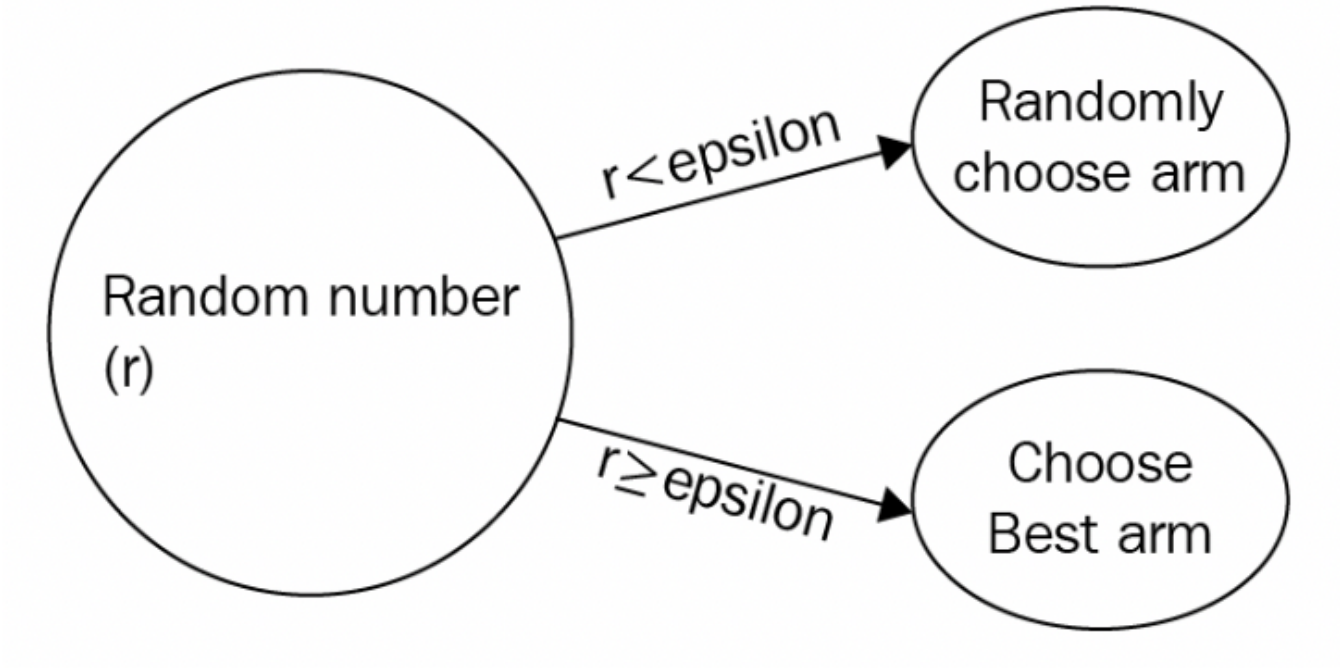


Multi-Armed Bandit Problem

Multi-armed bandit problem is a classical problem that models an agent (or planner or center) who wants to maximize its total reward by which it simultaneously desires to acquire new knowledge("exploration") and optimize his or her decisions based on existing knowledge("exploitation"). MAB problem captures the scenario where the gambler is faced with a trade-off between exploration, pulling less explored arms optimistically in search of an arm with better reward, and exploitation, pulling the arm known to be best till the current time instant, in terms of yielding the maximum reward.

MAB for cold users

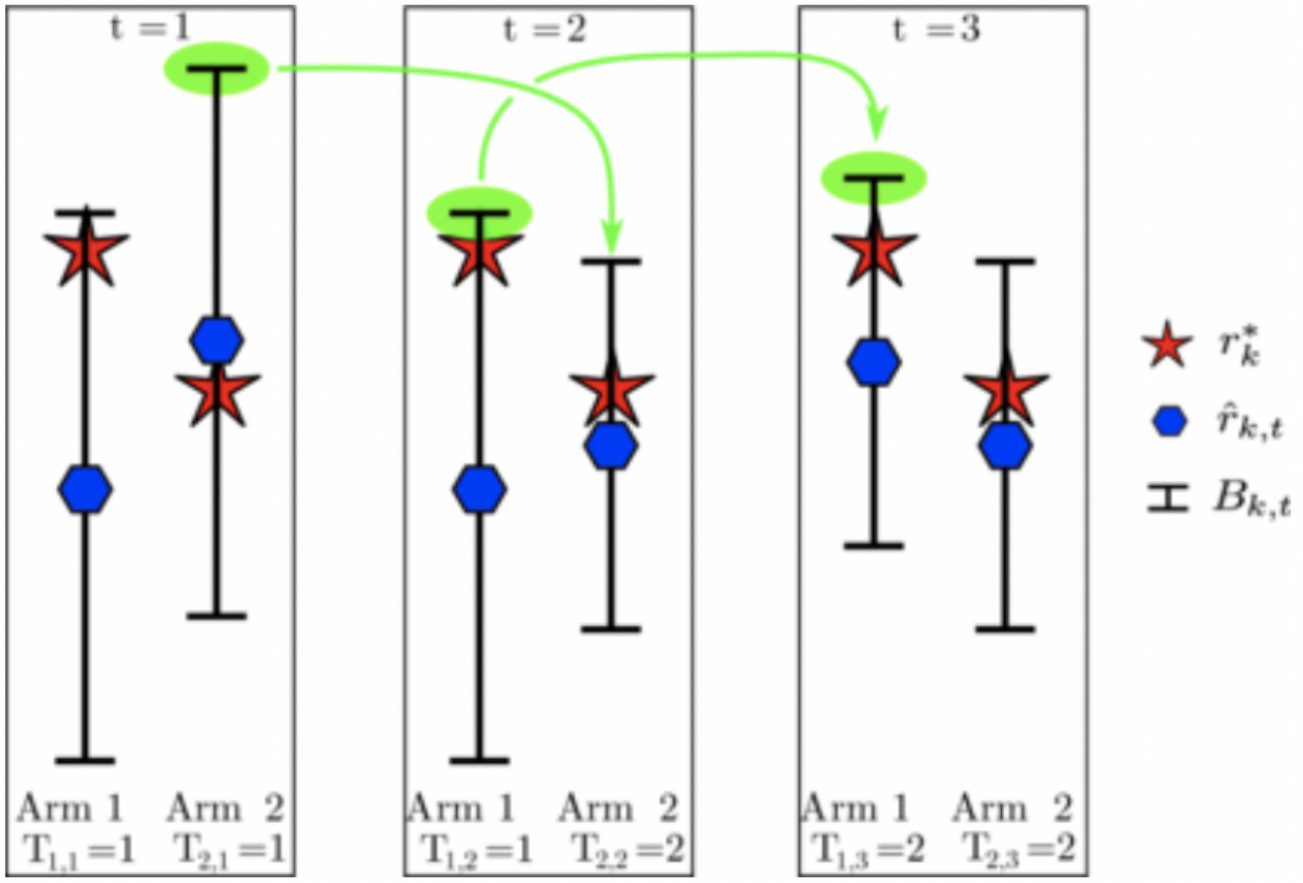
Our goal is to use different bandit algorithms to explore/ exploit optimal recommendations for the user. There are several MAB algorithms, each favoring exploitation over exploration to different degrees. Three of the most popular are



Epsilon Greedy Approach

Epsilon Greedy, as the name suggests, is the greediest of the three MAB algorithms. In Epsilon Greedy experiments, the constant ε (valued between 0 and 1) is selected by the user before the experiment starts. When allocating contacts to different variants of the campaign, a randomly chosen variant is picked ε of the time. The other 1-ε of the time, the variant with highest known payoff is chosen. The higher ε is, the more this algorithm favors exploration. For all our examples, ε is set to 0.1.

2. Upper confidence bound (UCB)



Upper confidence bound approach

For each variant of the campaign, we will identify upper confidence bound (UCB) that represents our highest guess at the possible payoff for that variant. The algorithm will assign contacts to the variant with the highest UCB.

The UCB for each variant is calculated based on both the mean payoff of the variant, as well as the number of contacts allocated to the variant, with the equation:

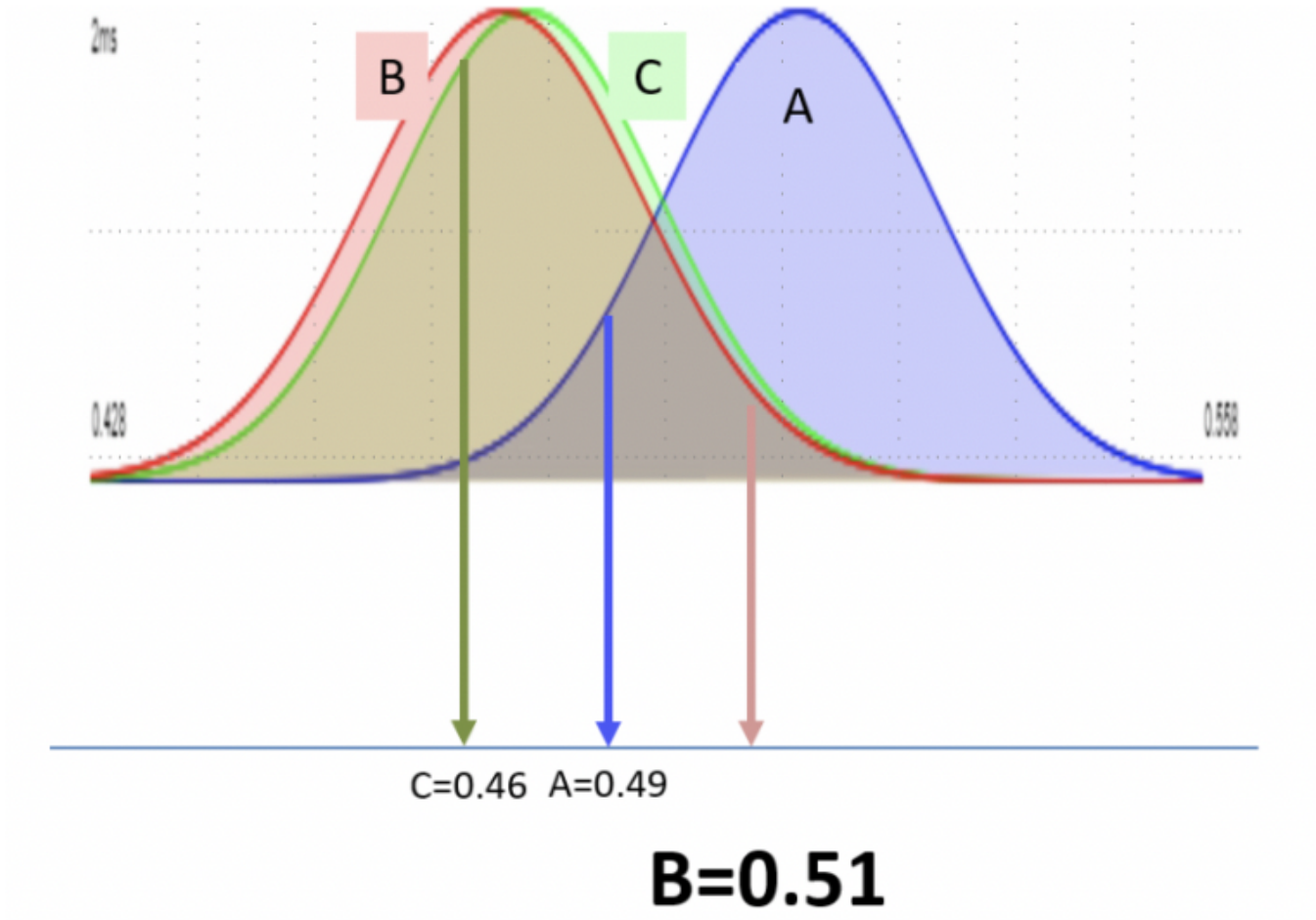
$$\text{UCB} = \overline{x_j} + \sqrt{2\,log\ t/n_j},$$

where

$\overline{x_j}$ = the average payoff at the jth step

t = total number of contacts that have entered the experiment

n_j = total number of contacts allocated to a particular variant

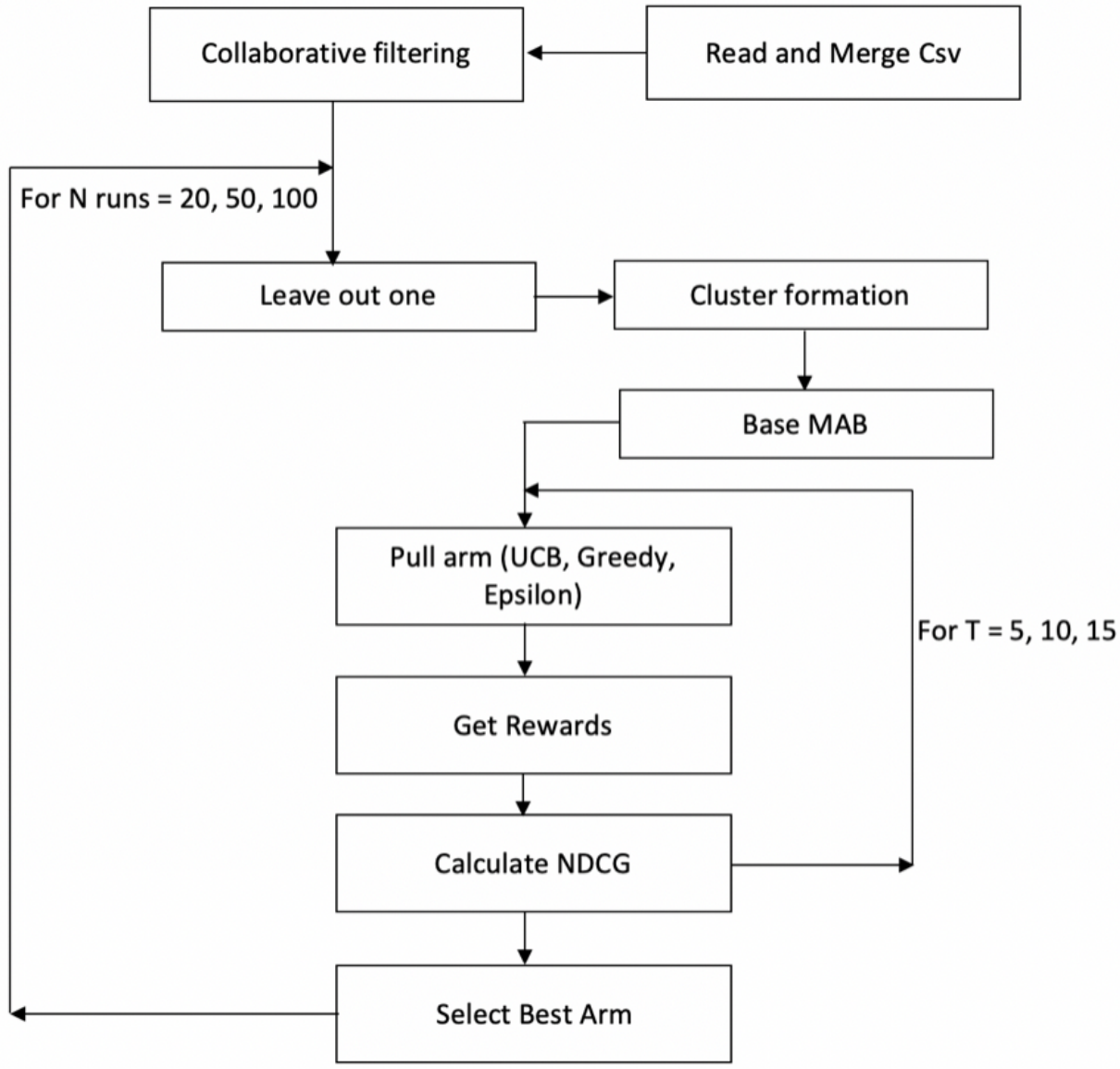


Thompson Sampling Approach

Thompson Sampling, by contrast, is more principled approach, which can yield more balanced results in marginal cases. For each variant, we build a probability distribution (most commonly a beta distribution, for computational reasons) of the true success rate, using observed results. For each new contact, we sample one possible success rate from the beta distribution corresponding to each variant and assign the contact to the variant with the largest sampled success rate. The more data points we have observed, the more confident we will be about the true success rate, and so as we gather more data, the sampled success rates will be more and more likely to be close to the true rate.

For a detailed description: https://www.youtube.com/watch?v=p701cYQeqew

Methodology



Infrastructure for solving MAB cold start problem

We have used Movie Lens dataset for solving the MAB problem which contains 4 files. These files were merged and used for collaborative filtering.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	5	2	4	-	5	1	-
u_2	4	-	5	-	5	-	1
u_3	2	5	3	5	-	-	-
u_4	1	-	2	-	2	-	-
u_5	-	-	3	4	1	-	-

Sparse Matrix from User Ratings

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	4.6	2.09	4.23	4.24	4.84	1.07	1.0
u_2	4.2	3.8	4.42	5.0	4.86	2.28	1.2
$\hat{\theta}_1$	4.4	2.94	4.32	4.62	4.85	1.67	1.1
u_3	1.97	4.84	3.22	4.87	2.68	2.68	1.61
u_4	1.19	3.24	2.17	3.56	1.92	1.23	1.0
u_5	1.77	3.16	2.81	4.07	1.14	1.6	1.56
$\hat{\theta}_2$	1.64	3.74	2.73	4.16	1.91	1.83	1.39

contribute to learning users preference faster. For example, the correspondent θ1 in above matrix will have the sorted list of items equal to {i5,i4,i1,i3,i2,i6,i7}, while for θ2 we will have {i4,i2,i3,i5, i6,i1,i7}.

This approach is applied to make recommendations to users that are initially new users who remain cold until they have provided a certain amount of preferences/feedback on recommended items. The threshold to determine when a user is not cold anymore can be distinct. In this blog, we measure the performance after making 5, 10, 15, 40, 100 recommendations using NDCG metric (Size of ranked list). As soon as the user is not cold anymore the system can switch to a personalized prediction model.

Reward Function

$$X_{M_s}(t) = \frac{r_{u,i}}{r_{max}}$$

Reward function

The reward function is defined as the ratio of feedback of the user provided to the maximum rating that the user provided. In this way, when a user gives higher feedback for a recommended item, we will have a high reward. For example, suppose a cold user enters the system and using UCB algorithm, item i5 is selected from θ1. If the user gives feedback of 4, then the reward for this movie would be $4/5 = 0.8$ for cluster θ1. In the next recommendation, if cluster θ2 is chosen, item i4 is selected and it receives feedback of 1 then the reward will be $1/5 = 0.2$ for cluster θ2. At this point of time, the mean reward for cluster θ1 is 0.8 and for cluster θ2 is 0.2. The next recommendation would be i1 from θ1 which represents the arm with the highest mean reward at that moment.

Metric for Evaluation

We will be using NDCG (Normalized Discounted Cumulative Gain) which values accurate recommendations in earlier round more than later ones. Since we focus on

$$DCG(u) = r_{u,1} + \sum_{t=2}^{T} \frac{r_{u,t}}{\log_2 t} \qquad NDCG = \frac{1}{N} \sum_{u} \frac{DCG(u)}{DCG^*(u)}$$

where

DCG(u): Discounted cumulative gain of predicted ranking for a target user u

DCG∗(u): Ground truth

N: Number of users in the result set

r (u,1): Rating (according to user feedback) of the item first recommended for user u

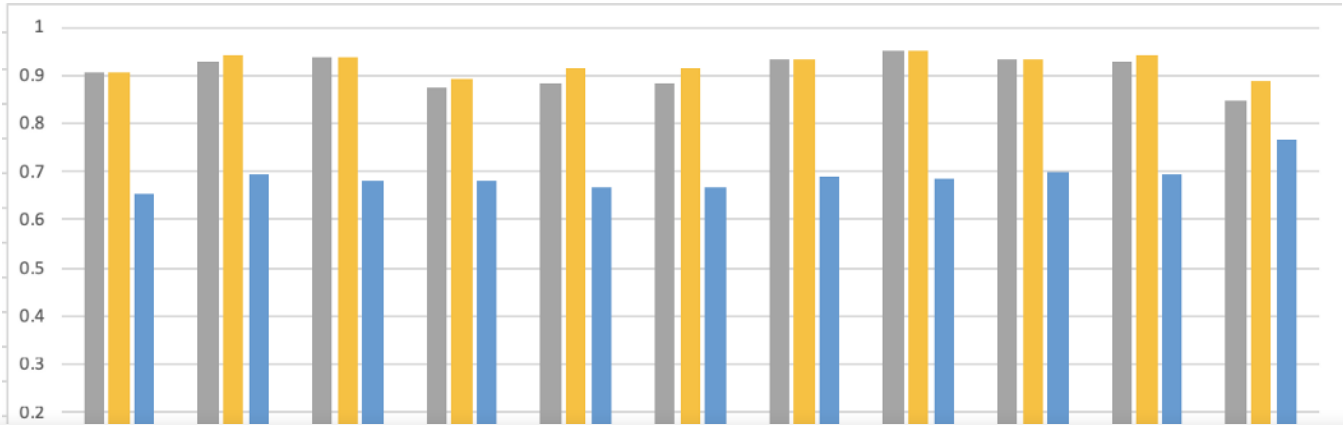
r (u, t): User feedback for the item recommended in turn t

t: Recommendation time

T: Size of the ranked list

Results

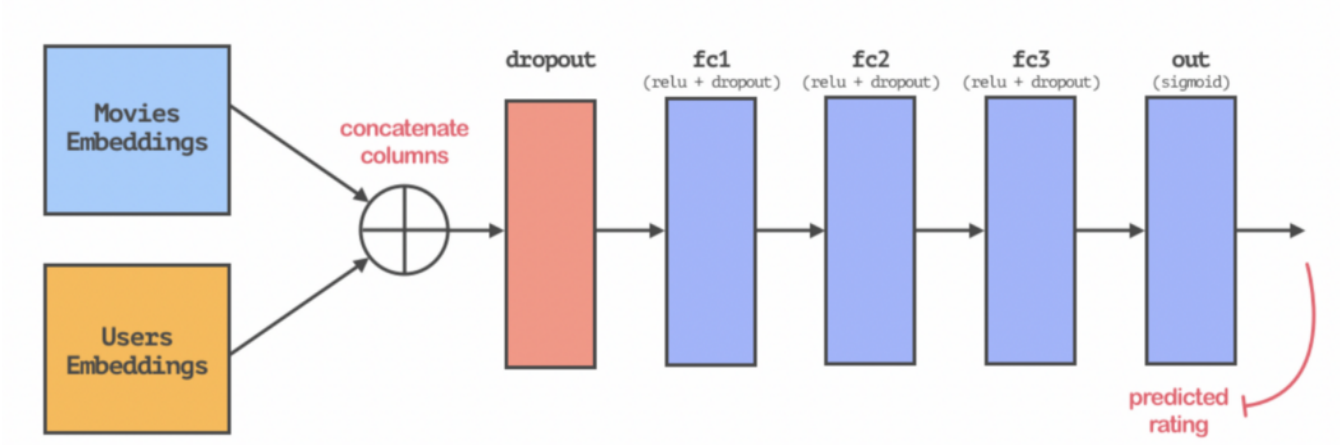
The table below provides the NDCG score for the rank-size of 5, 10, 15, 40 and 100. Thompson consistently performed better than Greedy and UCB. It took less time to warm up and was able to provide better results quickly. UCB, on the other hand, performed the worst for different values of N and T.



NDCG Results for varying rank-size and number of users

Embedding Networks

While embedding networks are typically used in a different problem, we wanted to see their effectiveness with the cold-start problem. Embedding networks are neural networks that learn embeddings between features. With this knowledge, it should be able to fill in empty values. So for the Movie Lens dataset, it can embed the relations between users and movies, and can then fill in the missing ratings that a user did not provide for a movie. While we just performed this task with collaborative filtering, using a neural network allows for the learning of complex, non-linear features.



Embedding network

So, an embedding network like one shown above can be used to fill in the missing values of an embedding below:

Users vs Movies

	A Game of Thrones	The Lord of The Rings	Star Trek	Star Wars	Titanic
Alice	5	4	3	3	5
Bob	5	n/a	4	5	3
Carol	n/a	3	5	n/a	2
Danny	4	4	5	3	4
Eve	5	3	?	3	5

setting, an embedded neural network can train once on a dense dataset, learn very complex relations, and give highly educated recommendations. However, our problems do not have the luxury of warm-users that result in a large dataset.

With the cold-start problem, we have to give good recommendations to users that have provided no feedback. Essentially, their data is empty, and there aren't any relations that can be inferred for this user. The naive solution for using an embedding network for this problem could be to first give a random recommendation, and for every following recommendation: take feedback, retrain, sort the predicted ratings, and return the highest projected recommendation. However, this would result in insane computational costs and delayed recommendations, which could provide disastrous for the user experience.

We had the idea that even though an embedding network has no knowledge of a cold-user, the networks knowledge of datasets relations among users and items could still provide some value. So, we wanted to analyze performance if, for every user request, we considered multiple recommendations. These recommendations could come from global recommenders such as random, global average, or most popular. They could also come from MAB algorithms such as Thompson, Epsilon-greedy, and UCB. Instead of directly feeding back a single algorithms recommendation, we could first let the embedded network estimate which of these recommendations would score highest. The highest projected recommendation would then be given to the user. Depending on computation and timing constraints, the network can be retrained on the user's feedback.

This idea was tested in a toy example reusing the MAB evaluation framework described in previous sections. Due to time constraints, the embedded network received hardly any training (5 epochs), and no hyperparameter tuning. It was also not retrained on user feedback. For each user request, it considered 3 random recommendations and gave to the user the one that it thought would score highest. However, it gave better results than UCB algorithm in this small test with 15 trials and 5 random users. One must note that strictly using random recommendations yields better results from this limited test shown below:


T	N	Random	Embedded NN	UCB
5	15	0.6933	0.6744	
5	20	0.7320		0.6555

While the results are not ground-breaking, it should be noted that further experiments have the opportunity to show much greater performance. More training, actually tuning the hyperparameters, retraining the network on the user feedback, and using other sampling recommenders (rather than just random) all offer the potential to greatly boost performance.

Conclusion and Future work

Although cold-start users in recommendation system pose a unique problem due to lack of knowledge about the user, MAB has done a pretty good job in recommending movies and is constantly evolving with data inputs. To sum up, our contributions are:

> *A formalization of the model selection as a multi-armed bandit problem*
>
> *Using UCB, Thompson Sampling and Epsilon greedy which is an effective approach to recommend for users without prior side information*
>
> *An empirical evaluation using NDCG on Movie Lens dataset*
>
> *Evaluating the effectiveness of Embedding network on a Recommendation system*

For future work, we can try

> *Having different bandit algorithms as the arms as we see epsilon-greedy performs better at*


> *We rely on NDCG score to access our approach, mainly because we were investigating the recommendation quality. Overall it presents high accuracy levels, but we might check other metrics to ensure a fair evaluation*
>
> *Tuning the hyperparameters of the Embedding network, retraining the network on the user feedback, and using other sampling recommenders (rather than just random)*

Thank you very much for reading, and we hope that you learned something from the project!

Feel Free to check out:

Github Repository of this post

My Other Medium Posts

My Linkedin Profile

References

https://medium.com/datadriveninvestor/how-to-built-a-recommender-system-rs-616c988d64b2

https://hal.inria.fr/hal-01517967/document

http://klerisson.github.io/papers/umap2017.pdf

https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/

https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

https://medium.com/the-andela-way/foundations-of-machine-learning-singular-value-decomposition-svd-162ac796c27d

https://sebastianraschka.com/Articles/2014_pca_step_by_step.html

https://medium.com/@iliazaitsev/how-to-implement-a-recommendation-system-with-deep-learning-and-pytorch-2d40476590f9

https://github.com/devforfu/pytorch_playground/blob/master/movielens.ipynb

https://nipunbatra.github.io/blog/2017/recommend-keras.html

https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526

https://medium.com/heycar/neural-network-embeddings-from-inception-to-simple-35e36cb0c173