Introduction

You have just ended a video conferencing call with your two friends, Jungmin and Lijing. They have an idea for an online business venture and they want your help. The business will be called *NextStar*. It will provide an application so that users can view the work of artists who have yet to be discovered.

Artists may include actors, singers, screenwriters, comedians, painters, sculptors and filmmakers. In fact, any artist who wants to demonstrate a talent will be able to upload files to the application. The uploaded content can be rated by all users. Based on these ratings, the application recommends new content to each user.

Jungmin and Lijing plan to make the *NextStar* website free to join and believe that they will be able to make money from advertising once there are enough users. They realize that this application will eventually require a great deal of storage, so they are looking at cloud-hosting companies. Once enough content has been added, the application will incorporate a recommender system.

The following information provides an outline of what has already been researched and includes some challenges for you to consider.

Cloud computing

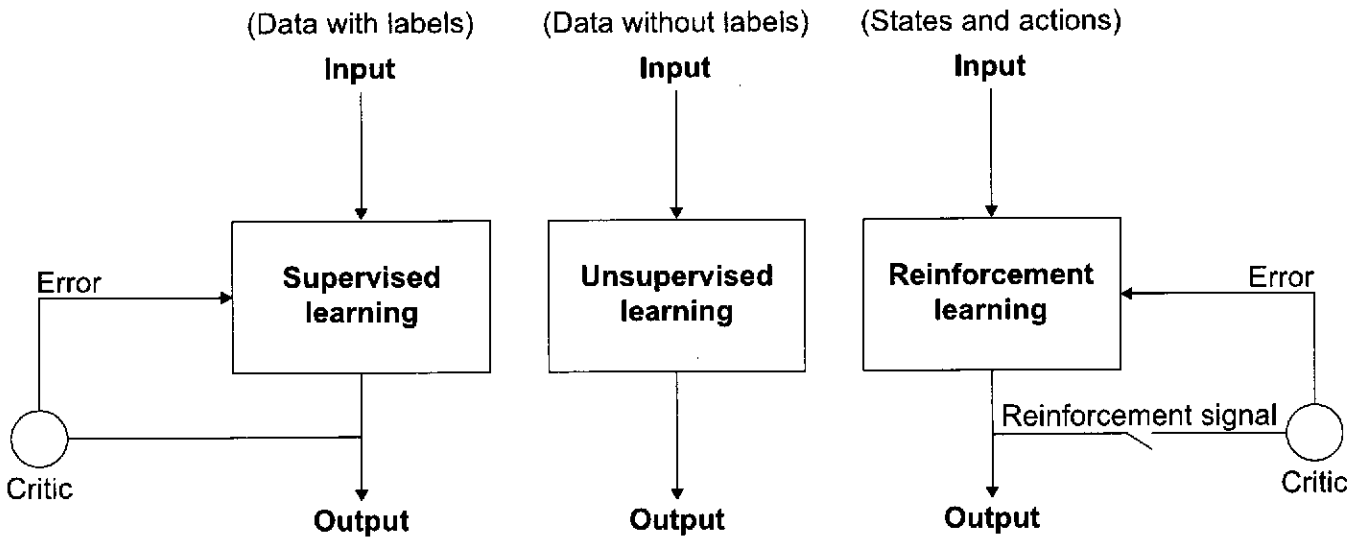
Hosting applications that utilize data at an enterprise level are widely available and affordable thanks to cloud computing. Users only pay for the resources they use, so they can start small and add more resources as they grow. This makes cloud computing ideal for a start-up like *NextStar*.

There are a number of *cloud deployment models* that could be used to host *NextStar*'s data. There are also three *cloud delivery models*: *software as a service (SaaS)*, *platform as a service (PaaS)*, and *infrastructure as a service (IaaS)*. *NextStar* intends to use IaaS.

Machine learning

Machine learning is a subfield of artificial intelligence. There are three main types of machine learning: supervised learning, unsupervised learning and *reinforcement learning* (see **Figure 1**).

Figure 1: The three main types of machine learning



A supervised learning algorithm uses labelled *training data* to learn a function that produces an appropriate output when given new unlabelled data. Typically, supervised learning is used to classify data or make predictions.

An unsupervised learning algorithm learns patterns from unlabelled data. These algorithms draw references from observations of the live input data. The system can organize data into subsets, or clusters, that have not been pre-classified by the programmers.

A reinforcement learning algorithm learns in an interactive environment by trial and error using feedback from its own actions and experiences. Some recommender systems can be seen as a type of reinforcement learning because positive behaviour, such as reviewing content, is rewarded with better recommendations.

Recommender systems

Where there is a huge amount of content available, a recommender system directs a user to content that they have not seen but may be of interest to them. Recommender systems use data for content that users have already rated (actual data) to generate predicted preferences for content that they have not already rated (predicted data).

The majority of recommender systems utilize supervised learning. The use of unsupervised learning and reinforcement learning is less common.

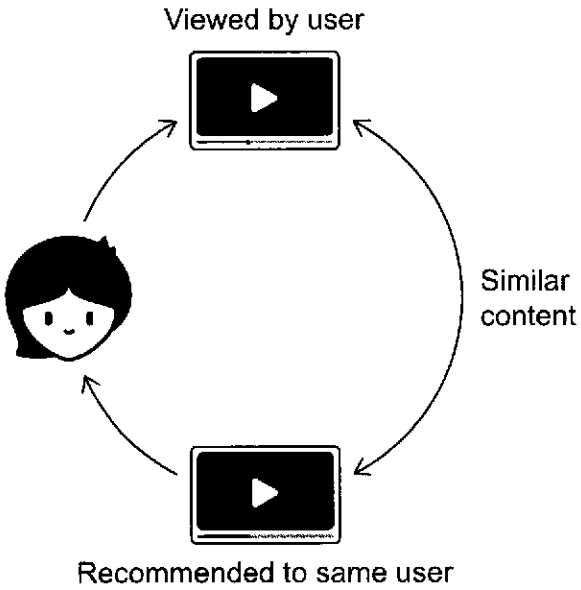
Recommender systems can use *content-based filtering*, *collaborative filtering*, or a combination of both. Hybrid recommender systems combine several machine learning algorithms. This was demonstrated on 21 September 2009, when BellKor's Pragmatic Chaos team won the Netflix Prize and USD 1 000 000 for the best collaborative filtering movie recommender system. This recommender system combined 107 different algorithms in a hybrid model that outperformed Netflix's own algorithm's *root-mean-square error (RMSE)* score by 10.06 %.

Content-based filtering

Content-based filtering, sometimes called item-item filtering, focuses on an item's attributes rather than using user interactions and feedback. The content-based approach is one of user-specific classification, in which the classifier learns the user's likes and dislikes based on an item's attributes.

Since *NextStar*'s recommender system will contain video clips, attributes might include genre, release date, artist, language, gender, and age. For example, if a user rates stand-up comedy clips highly, the system is likely to recommend more comedy clips to them (see **Figure 2**).

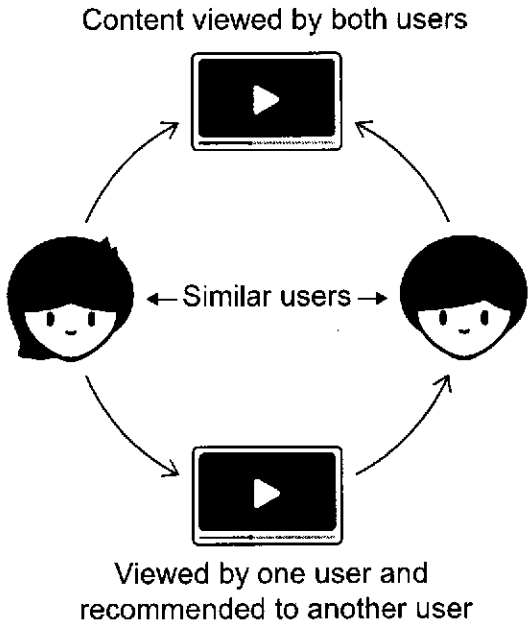
Figure 2: An example of content-based filtering



Collaborative filtering

With collaborative filtering, the recommendations for each user are generated by using the rating information from other users and items. The core assumption is that users who have agreed in the past tend to agree in the future. So, if two users scored content similarly, other content rated highly by one user is likely to be enjoyed by the other user. Thus, that content can be recommended to the second user (see **Figure 3**). One of the limitations of collaborative filtering recommender systems is *popularity bias*, where popular content is recommended too frequently.

Figure 3: An example of collaborative filtering



Collaborative filtering can use different algorithms to recommend new content. Two types of algorithm that can be used are *k-nearest neighbour (k-NN)* and *matrix factorization*.

K-nearest neighbour (k-NN)

The k-NN algorithm uses feature similarity to predict the values of any new or missing data. This means that new data points are assigned a value based on how closely they resemble other data points in the training set.

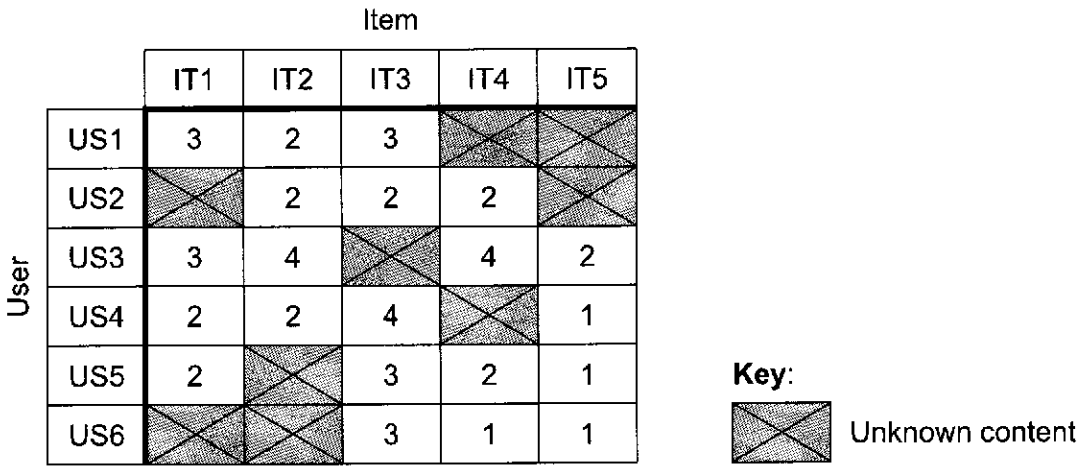
The k-NN algorithm makes its predictions based on the nearest neighbours. The "k" aspect of this algorithm represents the number of neighbours and is simply a *hyperparameter* that can be adjusted using a trial-and-error approach.

Matrix factorization

Matrix factorization is an alternative to the k-NN algorithm. The difficulty with using standard matrix factorization approaches for recommender systems is that the dataset is not complete. To overcome this limitation, values need to be estimated for the smaller matrices using an iterative algorithm.

In **Figure 4**, the user–item interaction matrix represents each user's rating (rows) of each content item (columns). User 1, represented by US1, has rated the first three items but has not rated item 4 or item 5, represented by IT4 and IT5.

Figure 4: User–item interaction matrix



Matrix factorization works by decomposing the large user–item interaction matrix into two smaller matrices—an item–feature matrix and a user–feature matrix—to capture the most important features required for learning. If the values in the item–feature matrix and the user–feature matrix are changed, the corresponding values in the user–item interaction matrix will also change (see **Figure 5**).

Figure 5: Matrix factorization

Item–feature matrix

Feature	Item	IT1	IT2	IT3	IT4	IT5
	F1	1.2	3.1	2.1	4.5	0.7
	F2	2.6	1.5	4.4	0.4	1.1

User–feature matrix

User	Feature	F1	F2
	US1	0.3	0.7
	US2	0.3	0.4
	US3	0.7	0.8
	US4	0.4	0.6
	US5	0.7	0.4
	US6	0.1	0.6

User–item interaction matrix

User	Item	IT1	IT2	IT3	IT4	IT5
	US1	2.18	1.98	3.71	1.63	0.98
	US2	1.4	1.53	2.39	1.51	0.65
	US3	2.92	3.37	4.99	3.47	1.37
	US4	2.04	2.14	3.48	2.04	0.94
	US5	1.88	2.77	3.23	3.31	0.93
	US6	1.68	1.21	2.85	0.69	0.73

The matrices are tuned by generating predicted preferences for content where actual preference data already exists. Once the prediction values approach the actual rating, the assumption is that the matrices will be able to effectively predict preferences for which no actual data exists.

A process called *stochastic gradient descent* uses a *cost function* to adjust each cell by making a small change to the item–feature and user–feature matrices. For example, in **Figure 4**, the value in the US1 and IT1 intersecting cell is 3, but in **Figure 5** this value is 2.18. So, the error for this cell is $(3 - 2.18)^2$, or 0.6724.

Training recommender systems

A recommender system can be evaluated using train/test splits. The ratings data is split into a training set and a testing set. A commonly used split is when 80 % of the data is assigned to the training set and the other 20 % to the testing set.

A recommender system learns the relationships between items and the relationships between users. Once trained, it makes predictions about how a user might rate an item that they haven't rated yet.

A common problem of training a machine learning algorithm is *overfitting*, where the model fits too closely to the training dataset. When the model trains for too long on the training data, or when the model is too complex, it can start to learn the irrelevant features within the dataset. Consequently, the model fails to generalize effectively against new data.

Evaluating recommender systems

Recommender system accuracy can be evaluated through two different measures: *mean absolute error* (MAE) and root-mean-square error (RMSE). These measures give an indication of how the recommender systems perform on training/test data.

However, the effectiveness of a recommender system is not fully known until it has been used by the public. A recommender system is not performing well if it fails to recommend content the user would like or recommends content that they do not like.

Precision and *recall* are performance metrics used on live data. Precision is a measure of exactness, the fraction of relevant instances among the retrieved instances. Recall is a measure of completeness. The *F-measure* provides a single score that balances the concerns of precision and recall.

The way that recommendations are displayed to users is also important. A list might be sufficient, or it may be possible to select recommended content by groups or subgroups. These groups might be organized by genre, gender, age or any number of possible categories.

Social and ethical concerns

When building a model from users' behaviour, two types of *behavioural data* can be used: explicit data and implicit data.

Explicit behavioural data refers to data gathered from users' submitted data, such as when a user rates a video clip, enters their preference, or searches for an item. Users may believe this is the only data that is used to make recommendations.

Implicit behavioural data refers to data that the user is not aware is being collected. This might include click data, purchase data, or even the use of a key logger.

The quality of user data is critical to the success of the *NextStar* project, but there are ethical concerns about the collection, storage and use of behavioural data. *NextStar* also needs to consider its users' *right to anonymity* and *right to privacy*.

Challenges faced

To help your friends with their new business venture, there are a number of challenges that you need to research:

- Understanding the similarities and differences between supervised learning, unsupervised learning and reinforcement learning.
- Understanding how the k-NN algorithm and matrix factorization can be used within recommender systems.
- Understanding how to train, test and evaluate a recommender system.
- Comparing content-based filtering and collaborative filtering recommender systems.
- Understanding the ethical concerns linked to the collection, storage and use of users' behavioural data.

Candidates are not required to know the mathematical equations relating to recommender systems.

Additional terminology

Behavioural data
Cloud delivery models:
- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as a service (SaaS)

Cloud deployment models
Collaborative filtering
Content-based filtering
Cost function
F-measure
Hyperparameter
K-nearest neighbour (k-NN) algorithm
Matrix factorization
Mean absolute error (MAE)
Overfitting
Popularity bias
Precision
Recall
Reinforcement learning
Right to anonymity
Right to privacy
Root-mean-square error (RMSE)
Stochastic gradient descent
Training data

Some companies, products, or individuals named in this case study are fictitious and any similarities with actual entities are purely coincidental.

References:

Figure 1 Jones, M. T., 2017. *Models for machine learning.* [online] Available at: https://developer.ibm.com/articles/cc-models-machine-learning/ [Accessed 15 October 2021]. SOURCE ADAPTED.