An Online Video Recommendation System Based on HDFS

Zizhan Chen	Tianyu Wang	Kai Zhang
1155135539	1155136646	1155135538
chenzz@cse.cuhk.edu.hk	tywang@cse.cuhk.edu.hk	kzhang@cse.cuhk.edu.hk

Abstract

In the field of internet information data mining, recommendation technology has been widely used to many user-oriented companies, such like Google, Amazon and YouTube. A good recommendation system will provide users with better experience, which is highly related to the users' satisfaction. In particular, video recommendation system faces more challenges than other tag-based recommendation systems due to its content-related characteristics. For instance, it requires much more computation and storage resources to deal with high-dimensional content features (over 1,000 features), which will cost too much to be processed in a single machine. How to efficiently analyze all the data becomes the major challenge.

In this project, we are about to build a video-to-video recommendation system based on YouTube-8M dataset. We firstly introduce map-reduce algorithm into our system to efficiently process high-dimensional data. To improve the effect of our recommendation system, we are trying to combine both tag and content analysis using two different methods. Finally, we will visualize all the results and further give a comparison between the methods we use.

Background and Motivation

With the popularity of Online Video, recommendation system plays more and more important roles in these services. In the Online Video Scenario, an accurate and fast recommendation system will attract the more users to watch more videos. Recent Online Video Websites such as YouTube [1] and Netflix [2] can provide precise recommendations to billions of users.

The recommendation of Video can be seen as two aspects. The first aspect is User Context [3], which means recommend what the user prefer. Each particular user has her own tastes. For example, a football fans will usually watch football game videos on YouTube. The second aspect is Video Context [3], based on the finding that different videos may have the same themes. For example, if a user watches a World Cup football game video on YouTube, there will be series of World Cup videos in the play list.

One way to solve Online Recommendation is using collaborative filtering [4] (CF) which suggest related items to a user based on other users with the similar tastes. This works well for experienced websites which have collected enough user information. But obviously, this approach is not friendly to newly established websites which do not have much user information. This is what we called cold start problem. It is the same for new videos. Because CF recommend videos base on users' preference records, the newly uploaded video which has not been seen by many people cannot be recommended. What's more, the CF bias on popular videos because of it algorithm logic. The new videos and small majorities will seldom be recommended. The CF is also not aware of the content, which lose lot of information that may be useful for recommendation.

Another way is using content-based recommendation. Users and videos will both be profiled and attached a series of tags. After profiling, users can be represented by vectors, so as the videos. The recommendation process can be seen as finding the nearest vectors based on some distance functions. The accuracy of content-based recommendation depends on how to define the item and users' profiles. But usually this is not easy, because user may have multiple tastes, we can only profile her tastes based on her actions, e.g. what kind of videos she has watched. It is difficult to predict what she may like. This is the same as a video. There are a thousand Hamlets in a thousand people's eyes. It is difficult to find out all the tags of a video.

This project adopts content-based recommendation method and focus on the Video Context. Different from the profiling-based content recommendation, we will use the raw visual features and audio features to do the recommendation. Specifically, we will use Youtube-8M dataset [5], each entry contains the video id, labels, mean_rgb (visual features) and mean_audio (audio features). This project will use HDFS to store these long feature vectors and utilize map-reduce to calculate the distance of video features and find the most similar videos. We also utilize cluster method to reduce the time cost of finding the similar videos.

Related topics

This project is related with the following topics:

(1) Recommender system.
(2) Clustering algorithm.
(3) Map-Reduce and HDFS.

We attempt to build a content-based recommendation system. We will use HDFS to store the video features and use map-reduce to find the most similar videos. We will use clustering algorithm to reduce the time cost.

Deliverables

We will demonstrate our system on following ways:

(4) Build a recommendation system based on HDFS. We will store the feature data from Youtube-8M in HDFS and use map-reduce to find the most similar videos.
(5) Establish a website that users can watch video and we will recommend videos to them. These videos will be redirected to YouTube based on the video ids.

Dataset

This project will utilize the Youtube-8M dataset [5]. Which contains

(1) 6.1 million video-ids. The videos are sampled uniformly from the YouTube video dataset.
(2) 350,000 hours of videos. This dataset is generated from 350K hours of videos. The volume is extremely big, it will take 50 CPU-years' worth of computation to process. YouTube has pre-computed them and extract features from these videos.

(3) 2.6 billion pre-computed audio and visual features from billions of frames and audio segments. The visual features are extracted by Inception-V3 image annotation model. The visual features are extracted by VGG-inspired acoustic model by Hershey et al [6].
(4) 3862 classes, including both coarse (e.g. sports) and fine-grained (e.g. football) classes.
(5) 3.0 avg. labels per video, the labels are distributed evenly for each video.
(6) The total size of the video-level features is 31 GB. Each entry has 1K float visual features and 128 float audio features.

Specifically, an example tensorflow.Example proto is shown below:

```
features: {
  feature: {
    key  : "id"
    value: {
      bytes_list: {
        value: (Video id)
      }
    }
  }
  feature: {
    key  : "labels"
    value: {
      int64_list: {
        value: [1, 522, 11, 172]  # label list
      }
    }
  }
  feature: {
    # Average of all 'rgb' features for the video
    key  : "mean_rgb"
    value: {
      float_list: {
        value: [1024 float features]
      }
    }
  }
  feature: {
    # Average of all 'audio' features for the video
    key  : "mean_audio"
    value: {
      float_list: {
        value: [128 float features]
      }
    }
  }
}
```

We will only use the video and audio signals, which is closer to the raw data. We assume we do not know the tags of videos before. This make sense because (1) We assume cold start, which means we do not know the tags before and (2) the tags will constrain our using of more information of video.

Techniques and Algorithms

Although we do not have an extremely big dataset, we need to provide recommendation service within a certain latency, which requires us to build a distributed processing environment to handle more than 30GB data. Hadoop is a widely used distributed data processing framework which could meet our requirement well. We will use map-reduce algorithm to process our dataset in parallel, which will significantly improve the performance and reduce the latency.

We will divide our dataset into several pieces and store them into different Hadoop Data Nodes. For the recommendation dataset, the reliability requirement is not very high because the original video is not included in the dataset, we choose to store each data record twice (two duplications). Although more copies will provide more reliability, it only has little contribution to our recommendation result.

For each piece of dataset stored in Hadoop Data Node, if we directly analyze them without any pre-processing, it will still be too difficult due to the large scale of the dataset. Thus, we will perform following two methods to do the data pre-processing:

One is clustering, which means data will be grouped into some number of clusters, so that members of a cluster are similar to each other and members of different cluster are dissimilar.

Here, our goals to use clustering as pre-processing is:

(1) View data distribution, which will help us to know more clearly about the whole dataset;
(2) With this clustering result, when we do recommendation after, it will provide us the information of one cluster and help us to reduce the number of useless calculations, e.g., calculating the similarity of very unrelated data.

So, to make sure that we will have relatively good clustering result, several different clustering algorithms will be implemented and compared with each other.

To implement clustering, there are several steps:

(1) Due to the large number of dimensions of features, at first, we need to determine how many dimensions and what dimensions we will use;
(2) According to the dimensions we select before, it is very important to choose appropriate similarity measurement;
(3) Choose efficient clustering algorithm to group data into clusters.

Actually, in step (1), because the data have features about labels, video and audio, generally the features of video and audio are handled together. So, for labels information provided originally by YouTube, we will consider how to use this feature. Both just using video and audio features and including labels features will be taken into consideration. With different number of dimensions used, there will be some difference between the accuracy of clustering results.

There are many similarities measures we can choose for step (2). For every record, the features contain different kind of variables, some of them are binary variables (labels information) and some are quantitative (floating-point) variables (video and audio features). We call these homogeneous attributes. Often, a feature vector contains several types of variables and an overall similarity is needed, so the combination for different similarities will be calculated in our system, which is called heterogeneous

attributes. For binary variables, we can just use simple matching coefficient or Jaccard's coefficient to test the similarity between two records. For quantitative variables, we can use distance to represent the similarity. And many methods can be selected from: Euclidean distance, Manhattan distance and Minkowski distance. Also, we can directly use cosine Similarity to measure the similarity between two vectors.

For the clustering algorithm we will use in this project, it needs to guarantee that the result is reasonable and can utilize the features from dataset. We will mainly select algorithms from following cluster models [6]: Connectivity-based clustering (prominent method: hierarchical clustering), Centroid-based clustering (prominent method: k-means clustering), Distribution-based clustering (prominent method: Gaussian mixture models) and Density-based clustering (prominent method: mean-shift). Then, some evaluations will be done on these different results to find better clustering result.

Except the pre-clustering method, we also propose a two-phase processing architecture to perform recommendation. Different from the concept of pre-clustering, that is, directly combining the tag vector and the feature vector, the two-phase processing architecture prefers to separate them into two stages, which only use one of them in each stage. However, the recommendation result may be different in the following two strategies:

(1) Using tag characters in phase-1, then integrating content characters to reduce the scope from the tag-based result.
(2) Using content characters in phase-1, then integrating tag characters to narrow the range from the content-based result.

For each stage, we can either choose the same algorithm or different algorithms to use. Different from the pre-clustering method, we will calculate the similarity from a particular variable type instead of combining different similarities together within a single stage. However, for different variable types between two stages, we can utilize the same functions as we prepared in the pre-clustering method to calculate the similarity.

We will try both two strategies above and give out the comparison between them (also the different algorithms we use). Results will tell us which one is better.

Demonstrate

We will show the experiment result of our system which include the accuracy of the recommendation, the time cost and storage cost of our system. We will compare our recommendation method with the traditional content-based methods. We will also compare the cost of different clustering methods.

Timeline

October 1st - October 15th: Determine the clustering algorithm.

October 15th – October 31st: Implement the map reduce to find the most similar videos.

November 1st – November 15th: Build the system, which includes HDFS and website.

November 15th – November 30th: Do experiments and write the final report.

Related Works

Joonseok et al. [8] utilize neural network to extract the features of videos and build a scalable recommendation system based on youtube-8m. Han et al. [9] proposed a dance-style recommendation method based on action representation. Van den Oord et al. [10] explored deep content-based music recommendations, learning CNNs fitting mel-frequency cepstral coefficients from songs to the ratings. Rothe et al. [11] used computer vision features to "regularize" matrix factorization (MF). They included an additive term to the MF model, which penalizes if the dot-product of two latent vectors from the MF model is far from the cosine similarity of their visual features.

These works are similar with our work, but we do not utilize neural network because it cannot match the map-reduce framework. We use a simpler method, but it is more scalable.

References

[1] Youtube, https://www.youtube.com/

[2] Netflix, https://www.netflix.com/

[3] Lee, Joonseok, and Sami Abu-El-Haija. "Large-scale content-only video recommendation." *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

[4] Sarwar, Badrul Munir, et al. "Item-based collaborative filtering recommendation algorithms." *Www* 1 (2001): 285-295.

[5] Abu-El-Haija, Sami, et al. "Youtube-8m: A large-scale video classification benchmark." *arXiv preprint arXiv:1609.08675* (2016).

[6] Hershey, Shawn, et al. "CNN architectures for large-scale audio classification." *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2017.

[7] https://en.wikipedia.org/wiki/Cluster_analysis

[8] Lee, Joonseok, and Sami Abu-El-Haija. "Large-scale content-only video recommendation." *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

[9] Han, Tingting, et al. "Dancelets mining for video recommendation based on dance styles." *IEEE Transactions on Multimedia* 19.4 (2016): 712-724.

[10] Van den Oord, Aaron, Sander Dieleman, and Benjamin Schrauwen. "Deep content-based music recommendation." *Advances in neural information processing systems*. 2013.

[11] Rothe, Rasmus, Radu Timofte, and Luc Van Gool. "Some like it hot-visual guidance for preference prediction." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.