Master thesis

omernivr12

July 2017

1 Abstract

Recommendation systems has become prevalent in recent years since the boom of web. Any internet based service has the advantage of having huge collection without the need to physically present it to the customer[2]. But, even if holding millions of products, any user can only see one web page at a time. Hence, for a user not to be lost in space, the service has to organize smartly the products to its liking. Recommendation of videos is one field that gained a lot of traction since the Netflix competition[4] was presented. Though many innovations were made over the years, there is still a wide search of combining the different features of videos and users into a comprehensive network for recommendations. This paper is tackling two blocks concerning the recommendation problem. Firstly, conceptually, it offers two novel ways to think about the recommendation problem. The first one is about selecting features, while the second one concerns a different architecture. Secondly, the promising architecture of GANs is used for the first time, as far as we are aware, to train a model in the video recommendation field.

2 Introduction

3 Background

This chapter will outline the framework in which recommendation systems are developed and the progress made over the years. The chapter is constructed in the way of first drawing the image of the general framework and then delving into explaining the specific popular methods used.

3.1 Problem

The generic problem for recommendation systems is that of inferring whether an item is active or not - 0,1, which is commonly known as an implicit data set. This is implicit since if 'Joe' read the book 'Mein Kamf', it does not immediately imply of its liking. The explicit data set consists of some scoring of items made by the user, as an example we can have a 1-5 rating. The recommendation problem could be viewed as interpolation problem [9] - this can simply be thought of just as given two points in space, choose a third point location and connect all to a triangle. Or, it could be viewed as a prediction problem - given the points viewed so far, predict the most likely points to appear next. These two problems require different solutions, since the former is time-independent and the latter is time dependent. The simplest form of the problem is that of having a rating matrix where each user is a row and each item is a column. This matrix would be very sparse - many values are missing since an individual only viewed few movies from Netflix libraries, or bought only few items from Amazon's stock. It is possible to ask users to rate items, but it is limited to the user's patience and more worrisome it is biased. Meaning a user might not rate disliked movies. To the simple matrix form of the problem it is possible to add many other dimensions representing item attributes and/or user attributes. However, some solutions become irrelevant when doing so. As mentioned above, the implicit problem offers the challenge of differentiating between active to liked item. Most solutions concerning implicit data sets only check what is being 'liked' but put aside the 'disliked' issue. This paper will offer a way to tackle this problem. This is based on the assumption that if a user watched an item for certain period and then stopped, it might flag a dislike. So a network representing dislikes combined with a network representing activity, could be used to infer if an active item is more driven by likes or by dislikes.

3.2 Recommendation framework

At the top most level we can differentiate between history based recommendation to session based recommendation. History based means tracking user's behaviour through long period of time, whereas session based refers to short term behaviour of an individual that is new to the system at every visit. History based has very broad research and some of its applications are of fit to session based as well. Choosing the branch of history based recommendation we are presented with solutions under one of content based, collaborative filtering (CF)[6] or hybrid blocks [1] , which are then divided to heuristic based

approaches and model based approaches. These separations fit also session based, but heuristic approaches are unstable due to sparsity. The subsections below will explore each block, its advantages and deficiencies.

3.3 Content based recommendations

The simple idea behind content based methods says: if a user watched a video about crime[7], recommend content related to crime. In order to do so, we should construct an item profile; containing any relevant features characterizing the item e.g. words that characterizes item topic. The latter example will play a major role in our novel architectures. Once a profile representing an item is constructed, it requires only to measure similarity between item' properties, and then recommending those that are most similar. To be more concrete, we calculate a matrix A where a_{ij} is the entry to i-th row and j-th column representing similarity between item i and item j. Given user U rated items i,j and k, go to rows i,j,k, order by descending value and pick top-k to recommend. Popular similarity measures in information retrieval are cosine measure and Jaccard similarity. Both similarities are intuitive, with cosine defining similarity by geometric meaning - closeness of angle between two vectors. While Jaccard is measuring similarity by proportion of overlapping items.
In a more formal way:

1. $cosine_\theta = \frac{A \cdot B}{\|A\| \cdot \|B\|}$
2. $Jaccard_{A,B} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$

Some of the main arguments rising against the use of content based methods are the inability to distinguish items with same features, the inability to offer to users content different from what they have already watched, the issue of recommending something which is too similar to previous items and the problem of the new user[1]. It is worth to dwell on the new user problem [9][] since it is something that most solutions still find hard to deal with. A new user must rate/buy/use few items before she can get similarity to other items and reasonable recommendations. An alternative is to include user features which leads us to the collaborative filtering block.

3.4 Collaborative filtering

Collaborative filtering[1] is the umbrella name for all methods that use some sort of similarity among users likes or items ratings to measure their similarity. The main idea at its core is the construction of a utility matrix. A utility matrix is one where users are represented by rows and items are represented by columns. We then have many missing values, with some filled with the ratings. Our aim is to fill the matrix either in full or with K values such that recommendations

will be possible.
The data inside the utility matrix is obtained by either asking the user - which is often referred as an explicit data set, or by inferring the behavior of the user - which is often referred as an implicit data set.
Asking the user to rate items involves the patience of the user, her inability to rate unknown items[10] and the bias associated with the 'not missing at random problem'[3]. This means that whether a rating is missing is missing depends on the video. In video rating this means, a user might not rate disliked videos or video that he is embarrassed sharing.
Inferring the user preferences through its activity is the problem we are facing with the channel four data at hand. So, we know that a user has watched a video or has not watched a video inside the platform. However, we do not know how a view maps to the liked/disliked space. With that in mind, even if we can reconstruct the perfect matching of the utility matrix it might still be useless in practice. Hence, one of our main ideas in the paper, is to tackle the problem of implicit information to some extent using additional information which is not less important.
Under the umbrella, the separation breaks to two major groups. The one being memory or heuristic based approach and the other being model based. The main difference is that a model based approach is to use the database to learn a model which is then used for predictions. (empirical analysis of predictive algo.).
The simplest and what was quite popular way with memory based is to find N users similar to user U, then we can average the ratings for item I. Where similarity can be measured by the cosine or Jaccard similarity mentioned above. However, this naive way suffers from many issues such as being slow, it does not work well for users with unusual preferences. In addition, the new user problem is evident with the problem of what items to show.
What items to show can be mitigated in few ways: 1. random 2. popularity 3. entropy 4. item-item. (getting to know you)
by entropy, calculate each movie using relative frequency of each of the five possible ratings and present movies with highest entropy. If user i can rate item j, it is considered as part of the users that are contributing to the entropy of user a. we then have the probability of user i(PMCF). ...
The slow computation render this user to user method impractical in real world problems. What brought upon the item-item method (Amazon). The change is to look at items in the utility matrix instead of the users. i.e we can compute cosine similarity between pairs if columns in utility matrix. all these calculations can be made offline so it can be very quick to recommend an item in the online setting. This is in contrast to the user-user setting where a new user needs to be in the online setting calculated against many other users.
Here the problem of the new user is replaced by the new item issue. The new item does not have any ratings associated with it, so can not be recommended. Another issue is when an item is liked by all. Universally liked items are not useful in capturing similarity.
The other group under the umbrella is the model based collaborative filter-

ing. The most renowned of these methods for many years is the dimensionality reduction and in particular Matrix Factorization in its various forms (PMCF, PMF,...). The simplest form is derived from the Principal component analysis (PCA) method.
PCA is a way to deconstruct a matrix to a lower dimension representation matrix times a basis matrix. We want to find the dimensions that are communicating the most amount of variation in the data. The idea is building upon the notion that it is many times the case that lower dimension manifold is lying inside a higher dimensional space that does not add much information at inference. Think about a data set that contains both the weight in grams and in Kg. One of the two is redundant since they communicate exactly the same information and hence can be disregarded. We would want the PCA to be able to find these so called redundancies. Formally, it is....Can be preformed more efficiently by SVD.
The idea of dimension reduction is that there are relatively small set of features of items and users that determine reaction of users to items. In the field of videos, an easy example can be made by thinking about the shared space of videos and items as finding the genres shared by the different videos. The resulting matrix would be the number of videos as rows and the number of genres as columns. the basis matrix would be the weighting of each of these genres to each user, such that the original matrix is the result for.
Fortunately, PCA can also serve to complete a partially filled matrix. The method is the same as it is for the regular PCA, just that this time we would only measure the loss in the places that values do exist (barber). We can solve this by two ways : 1. Fix U then solve linear system for V, then fix V and solve linear system for U.
2. Initialize values of the U, V matrices and preform gradient descent on the RMSE loss. As PCA has no probabilistic interpretation, it might be desired to use its probabilistic model (PMF). In this view, the rating of user i to item j is distributed around the mean UV_{ij} of a multi dimensional normal distribution. The user and video matrices are getting also a normal prior distribution with 0 mean and constant variance. the posterior is then... A problem that arises in the context of matrix completion in all its forms is, how to choose the number of features D that will constitute the lower dimension representation.
In the recent years with the rise of Neural Networks and its family, many other methods were presented for matrix completion among them RBM [**Salakhutdinov2007**], BRNN [**Schuster1997**] for filling missing values and auto-encoders [**Sedhain2015**]. With the latter being the closest to the PCA completion method and has shown the most promising results regarding video recommendation using collaborative filtering[**Sedhain2015**]. Due to its simplicity, conceptual matching to older collaborative filtering techniques and its results, we chose to employ the auto-encoders to the disliked data. Auto-encoders [3] are a fancy name for dimensionality reduction using neural networks. We construct them by having in the minimal case a layer where we input the original matrix and then we have a lower dimensional layer where we apply a function above X. We then apply another function above the lower dimension layer to try and reconstruct X. The

loss that is being measured is the reconstruction loss between output and input. This is an unsupervised method. If we use only linear functions such as f(x) = Wx at the reduction layer and output layer we get a similar reconstruction to PCA.
Same as in matrix completion methods, we can also use auto-encoder to fill in values. However, as we usually input to the first layer of our network the original matrix, an extra care should be applied when considering the missing values. If these are set to zero, we are biasing the model towards zero values which we do not want to do, especially not in our implicit data set where zeros have a meaningful interpretation as non active. Few initial values could be considered, such as the mean value of items, or the output of a matrix completion procedure. These values would still not be considered in the loss, and will not be back propagated during training, but would affect the interaction of the other weights and hence should be reasonable.
Until now we have considered models and memory based techniques that are neither time dependent nor feature rich. For example we have not considered the age of our user to affect her viewing behavior. But it is quite intuitive that a six year old would have different preferences than fifty years old. If our technique takes this feature and others into account, then it will be much easier to group the different possible videos to recommend next for users. Few methods that seem interesting ad promising in the dimension reduction sphere are inductive matrix completion, that extends the idea that we can decompose the original matrix to

3.5 Features

As mentioned in the content based part above, an important part is to find similarity among properties of the items or videos in our case. Some of the common features to use are genres, which we use as well. Genres are assumed to be very helpful to recommendation, since if a user is watching only teenage comedies, then it will be quite simple to construct the table of possible videos she is going to watch next. Since the number of genres is not so big, we can use one hot encoding to represent them. i.e. construct a list of genres, and place a 1 if genre is active in this specific video, otherwise place 0. example: With genres $\mathcal{G} = \{Comedy, Horror, Thriller\}$ if user i watched 'When Harry met Sally' the corresponding vector would be {1, 0, 0}.
Another feature, which is less common for video recommendation is to use the script/subtitles of the video as a characteristic. We have decided to use it with the use case that a stand-up video might be tagged in genres of comedy and stand-up, but if the text is about drug abuse, and then you move subsequently to watch 'Trainspotting' the text itself would be more telling than the genre tags. Although this idea is not very common in the video recommendation industry, the idea is prevalent in written documents classification to topics.
Several techniques are available with the widely used TF.IDF [8] from information retrieval, LDA [5] from unsupervised learning using variational infernce and Word2Vec from . The common to the three is that they are all can be used

to find similarity in space of words and/documents to one another.. Our choice is the word2vec method, which was chosen since it is an inherent part of the Neural network methods that are used in our architecture. For completion we still give a short explanation regarding TF.IDF and LDA.
TF.IDF can be explained easily as a counting method. Where we count the number of times a term occurs in a document and we weight this counter by the frequency across all documents in the set. If the same word appears frequently in all documents, there is not much we can learn about the specific topic. The simplest example would be the so-called stop words like {the, on, a} which are widely used with no meaning behind. This idea is both intuitive and fits good writing practices[11], which emphasize non-redundancy and simplicity as its main properties. Once a score vector is obtained, we can pick the top K words as characterizations. Counting however does not necessarily imply on the underlying characterisitcs of the text.
LDA is an unsupervised, meaning we do not have topics beforehand but we construct them from scratch. The approach is used to classify words in a document to topics. Whilst the idea is simple, its mathematical formulation is more evolved, and will not be stated here. The idea is that once we have a collection of documents each containing of words. Each of these documents has a different distribution over words. We then want to assign a word to a topic without forgetting the probability of this word under this topic. To clarify, It might be that a document consists of two topics with one 99 percent and the other 1 percent. However, the probability of the word *'qualia'* is zero under this topic. So it would still make sense to choose the topic with the one percent chance.
Word2Vec

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*. 2005. DOI: 10.1109/TKDE.2005.99. arXiv: 3.

[2] Chris Anderson. "The Long Tail". In: *WIRED magazine* 12.10 (2004), pp. 170–177. ISSN: 1580979X. DOI: 10.3359/oz0912041. URL: http://www.geeknewscentral.com/2010/08/13/the-long-tail/.

[3] David Barber. "Bayesian Reasoning and Machine Learning". In: *Machine Learning* (2011), p. 646. ISSN: 9780521518147. DOI: 10.1017/CBO9780511804779. arXiv: arXiv:1011.1669v3. URL: http://eprints.pascal-network.org/archive/00007920/%7B%5C%7D5Cnhttp://scholar.google.com/scholar?hl=en%7B%5C&%7DbtnG=Search%7B%5C&%7Dq=intitle:Bayesian+Reasoning+and+Machine+Learning%7B%5C#%7D0.

[4] James Bennett and Stan Lanning. "The Netflix Prize". In: *KDD Cup and Workshop* (2007), pp. 3–6. ISSN: 1554351X. DOI: 10.1145/1562764.1562769.

[5] David M Blei et al. “Latent Dirichlet Allocation”. In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022. ISSN: 15324435. DOI: 10.1162/jmlr.2003.3.4-5.993. arXiv: 1111.6189v1.

[6] Jonathan L Herlocker et al. “Evaluating collaborative filtering recommender systems”. In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pp. 5–53. ISSN: 1046-8188. DOI: 10.1145/963770.963772. arXiv: 50. URL: http://portal.acm.org/citation.cfm?id=963770.963772.

[7] Anand Rajaraman and Jeffrey D Ullman. “Mining of Massive Datasets”. In: *Lecture Notes for Stanford CS345A Web Mining* 67 (2011), p. 328. ISSN: 01420615. DOI: 10.1017/CBO9781139058452. arXiv: arXiv:1011.1669v3. URL: http://ebooks.cambridge.org/ref/id/CBO9781139058452.

[8] Amit Singhal. “Modern information retrieval: A brief overview”. In: *IEEE Data Engineering Bulletin* (2001), pp. 1–9. ISSN: 00218979. DOI: citeulike-article-id:1726446. arXiv: 9780201398298. URL: http://ilps.science.uva.nl/Teaching/0405/AR/part2/ir%7B%5C_%7Doverview.pdf.

[9] Chao-yuan Wu et al. “Recurrent Recommender Networks”. In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17*. 2017, pp. 495–503. ISBN: 9781450346757. DOI: 10.1145/3018661.3018689. URL: http://dl.acm.org/citation.cfm?doid=3018661.3018689.

[10] Kai Yu et al. “Probabilistic Memory-Based Collaborative Filtering”. In: *IEEE Transactions on Knowledge and Data Engineering* 16.1 (2004), pp. 56–69. ISSN: 10414347. DOI: 10.1109/TKDE.2004.1264822.

[11] William Zinsser. “On Writing Well”. In: *Quill* (2001), pp. 1–308. ISSN: 0060891548.