Report

Hotel Recommendation Engine using Machine Learning

By:

Shivani Patel
14BCE118

Kavi Sanghvi
14BCE103

Computer Engineering Department

Ahmedabad 382481

Hotel Recommendation Engine using ML

Minor Project

Submitted in fulfillment of the requirements for the degree of

Bachelor of Technology in Computer Engineering

By

Shivani Patel
14BCE118
Kavi Sanghvi
14BCE103

Guided by
Dr. K P Agrawal
(Computer Engineering Department)

Computer Engineering Department

Institute of Technology, Nirma University - 382481.

CERTIFICATE

This is to certify that the Minor Project entitled *Hotel Recommendation Engine using ML* submitted by *Shivani Patel (14BCE118) and Kavi Sanghvi (14BCE103)*, towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Engineering of Nirma University is the record of work carried out by them under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Dr. K P Agrawal
Associate Professor
Computer Engg. Department
Institute of Technology,
Nirma University,
Ahmedabad.

Dr. Sanjay Garg
Head of Department
Computer Engg. Department
Institute of Technology,
Nirma University,
Ahmedabad.

ACKNOWLEDGEMENT

Our sincere thanks goes to all those who reviewed the work before finalising the report. Received a lot of guidance and support from our mentor for the seminar, Dr. K P Agrawal. We truly appreciate the time and advice he gave to encourage us to explore further deep into the topic. We would also like to thank our peers and colleagues for providing sincere and honest appraisal, comments and suggestions.

We wish to make a special mention of Prof. Tarjni Vyas, panel member for Review 2, for providing us directions for further enhancement of the studies conducted so far.

Sincerely,

Shivani Patel
14BCE118
Kavi Sanghvi
14BCE103

ABSTRACT

This report enunciates the details of the project titled *Hotel Recommendation Engine with ML*. On a broader sense, it deals with the machine learning aspects of any recommender system - analysing and managing the huge dataset, extracting trends and patterns from it, deciding the distance metric and model best suited for the problem at hand, and iteratively improving it till it surpasses the already set benchmark. Every day, a great many Terabytes of information is being created on the web. It is a typical idea that the more decisions a human gets, the more confounded he moves toward becoming. By making a proposal engine, we intend to tackle this perplexity by anticipating the most pertinent items particular to every client. This is valuable to both the user and the property owner. Explained in the report are theoretical concepts central to the project - Mean Average Precision (MAP5) and Principal Component Analysis (PCA) and their implementations. Furthermore, also discussed in the end are the limitations and the future scope of the said project and improvements in the IR and ML models.

TABLE OF CONTENTS

CHAPTER 1 Introduction

1.1 General

1.2 Objective of Study

1.3 Scope of Work

CHAPTER 2 Expedia Dataset

2.1 Literature Survey

2.2 Dataset Explanation

2.3 Visualisation of Trends in the Dataset

CHAPTER 3 Distance Metrics

3.1 Overview

3.2 Mathematical Aspect of Metrics

CHAPTER 4 The ML Approach

4.1 Principal Component Analysis

4.2 Evaluation by Mean Average Precision

4.3 Constructing Benchmark

4.4 Further Improvements - Exploiting Data Leaks

CHAPTER F Epilogue

F.1 Summary

F.2 Conclusion

References

Appendix - A

CHAPTER 1 INTRODUCTION

1.1 GENERAL

With each passing second we are witnessing dumps of terabytes of data that needs to be processed to be presented to the user. If we could be able to recommend each user handpicked data, it could revolutionize how we use the internet.

We have implemented a recommender system which recommends the most apt hotel according to each user preference based on variety of factors. This would help Expedia to increase it's sales and reduce the booking time for each. This way we are achieving optimisation in the booking system, while increasing the customer satisfaction too.

1.2 OBJECTIVE OF STUDY

Each day, thousands of Terabytes of data is being generated on the internet. It is a common notion that the more choices a human gets, the more confused he becomes. By making a recommendation engine, we aim to solve this confusion by predicting the most relevant products specific to each user. This is beneficial to both the buyer and the seller. Buyer gets what he wishes with convenience and in minimum time while the seller increases his sales. Imagine a system in which as soon as the user searches for the product, he gets the products ranked in a system in which the higher the product is in the results, the higher would be its probability to be bought. Such a request will guarantee, to the point that a client sees the most applicable hotels first and increment the probability that the online travel office wins the exchange. The objective behind choosing this topic and working upon it is thus to attempt to build such a system.

1.3 SCOPE OF WORK

The primary objective of the project was just to improve the backend of recommender and booking sites (Expedia, in context) and thus focus has only been given to the Information Retrieval and Machine Learning part and not to building the front-end. However, as a standalone project, the end result of the project can very well be combined with a supplementary web-based GUI powered by Django - which is a web (HTML) framework for Python, to seamlessly integrate with the backend code which is also in Python.

CHAPTER 2 Expedia Dataset

2.1 Literature Survey

Our project is based on the Expedia Hotel Recommendations challenge on Kaggle.com. The goal of this project is to predict which of the 100 hotel clusters that a random Expedia visitor will book a hotel from. The high-level application of this project is to allow Expedia to provide the optimal personalized hotel recommendations for the user based on a user search event, which will increase the number of hotels booked through Expedia and simultaneously increase user satisfaction in the product. However, since the problem involves presenting optimal recommendations, which the user is presented to choose from, this problem, is not simply another multi-class classification problem. We must instead predict five hotel clusters that the user is most likely to book.

2.2 Dataset Explanation

The dataset provided officially by Expedia is a collection of logs of user behaviour, as well as property details for tens of thousands of hotels and 149 latent features about the destination - normalised to floating point numbers so as to preserve the exact details. There are internal mappings from the IDs to the actual hotels and users which is abstracted for obvious privacy reasons.

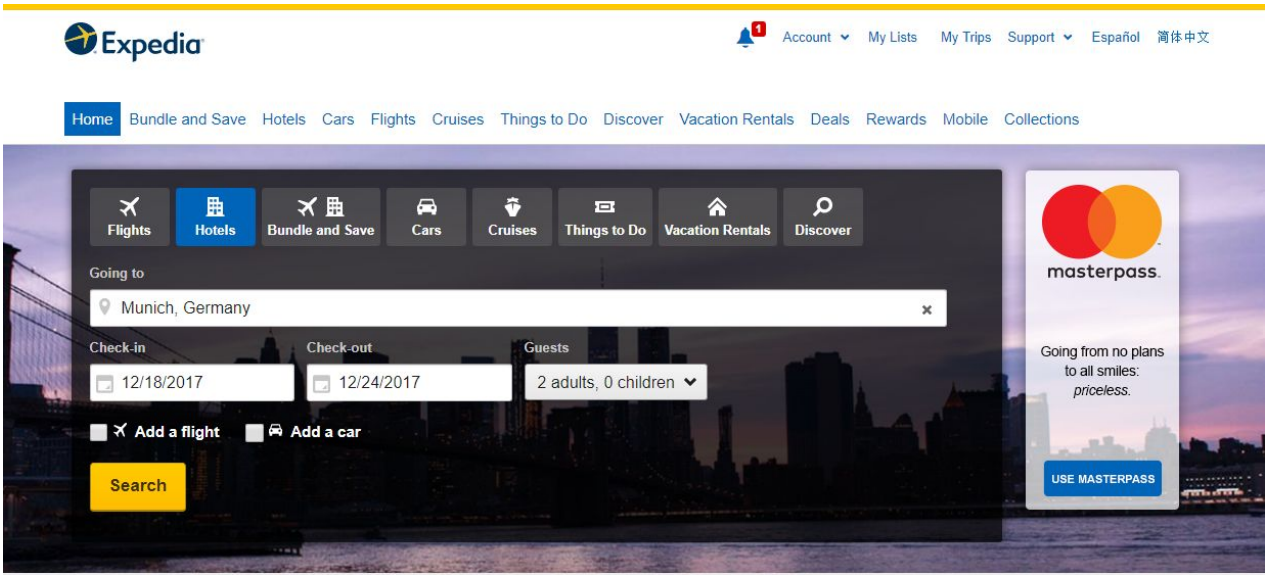
Taking a look at the description of each column in the table below, it creates the impression that we have a considerable amount of information about the searches clients are leading on Expedia, alongside information on what hotel group they in the end booking, in test.csv and train.csv. destinations.csv contains data about the locales clients scan in for lodgings.

Column name	Description	Data type
date_time	Timestamp	string
site_name	ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...)	int
posa_continent	ID of continent associated with site_name	int
user_location_country	The ID of the country the customer is located	int
user_location_region	The ID of the region the customer is located	int
user_location_city	The ID of the city the customer is located	int
orig_destination_distance	Physical distance between a hotel and a customer at the time of search. A null means the distance could not be calculated	double
user_id	ID of user	int
is_mobile	1 when a user connected from a mobile device, 0 otherwise	tinyint
is_package	1 if the click/booking was generated as a part of a package (i.e. combined with a flight), 0 otherwise	int
channel	ID of a marketing channel	int
srch_ci	Checkin date	string
srch_co	Checkout date	string
srch_adults_cnt	The number of adults specified in the hotel room	int
srch_children_cnt	The number of (extra occupancy) children specified in the hotel room	int
srch_rm_cnt	The number of hotel rooms specified in the search	int
srch_destination_id	ID of the destination where the hotel search was performed	int
srch_destination_type_id	Type of destination	int
hotel_continent	Hotel continent	int
hotel_country	Hotel country	int
hotel_market	Hotel market	int
is_booking	1 if a booking, 0 if a click	tinyint
cnt	Numer of similar events in the context of the same user session	bigint
hotel_cluster	ID of a hotel cluster	int

Column name	Description	Data type
srch_destination_id	ID of the destination where the hotel search was performed	int
d1-d149	latent description of search regions	double

2.2.1 Data Gathered through Expedia UI

Alongwith the numerous fields of data for each destination and hotel that can be fairly constituted as static, a lot of parameters are also passed to this backend program when a booking is made. Glancing at the homepage of the site:



The box labelled Going-To maps to the *srch_destination_type_id, hotel_continent, hotel_country,* and *hotel_market* fields in the data.

site_name is the name of the site you visited, whether it be the main Expedia.com site, or some other domain like Expedia.co.in.

The box labelled Add a Flight maps to the *is_package* field in the data.

The box labelled Check-in maps to the *srch_ci* field in the data, and the box labelled Check-out maps to the *srch_co* field in the data.

The box labelled Guests maps to the *srch_adults_cnt, srch_children_cnt, and srch_rm_cnt* fields in the data.

user_location_country, user_location_region, user_location_city, is_mobile, channel is_booking, and *cnt* are all attributes that are determined by where the user is, what their device is, or their session on the Expedia site.

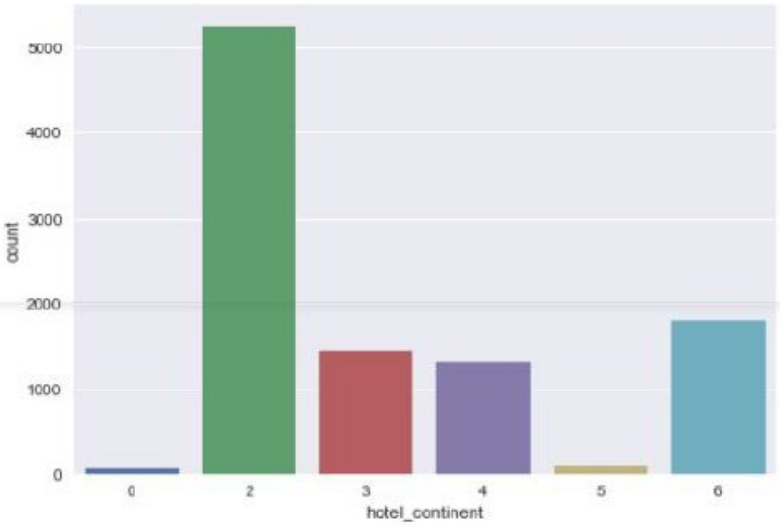
2.3 Visualisation of Trends in the Dataset

The given dataset includes a training dataset with about 30 million examples and an evaluation dataset of about 2.6 million examples with hidden output hotel cluster. It contains 17 features. We haven't found any strong correlation between any features and so we couldn't eliminate any of the feature directly.

The trends in the data are visualised using matplotlib for Python, and importing Seaborn library to make it look more aesthetic for user to visualise. Some of the screenshots of the results are: (all features NOT shown)

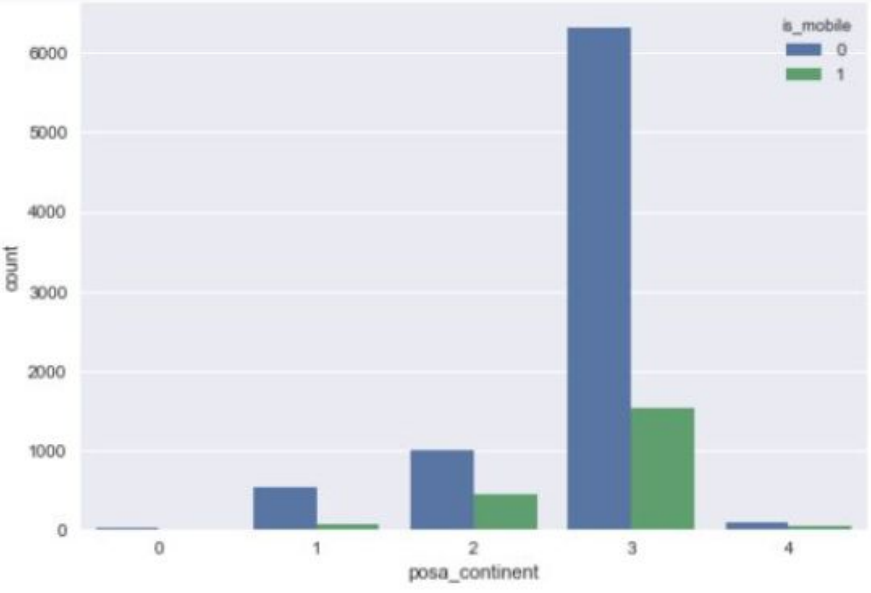
```
In [3]: import seaborn as sns
        import matplotlib.pyplot as plt
        # preferred continent destinations
        sns.countplot(x='hotel_continent', data=train)
```

```
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x155a22700b8>
```



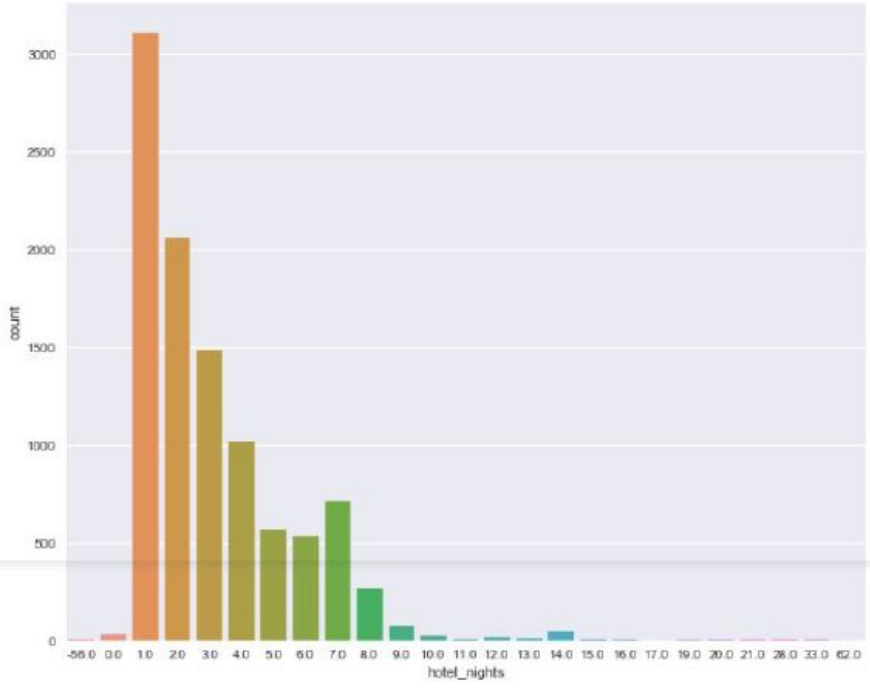

```
In [6]: # how many people by continent are booking from mobile
        sns.countplot(x='posa_continent', hue='is_mobile', data = train)
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x155a259eba8>
```



In [9]:
```
# get number of booked nights as difference between check in and check out
hotel_nights = train['srch_co'] - train['srch_ci']
hotel_nights = (hotel_nights / np.timedelta64(1, 'D')).astype(float) # convert to float to avoid NA problems
train['hotel_nights'] = hotel_nights
plt.figure(figsize=(11, 9))
sns.countplot(x="hotel_nights", data=train)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x155a3a89080>
```



CHAPTER 3 Distance Metrics

3.1 Overview

The first step in the actual implementation of the project is to comprehend the nature of the dataset and figure out which distance metric to use. Measuring similarity or distance between two data points is fundamental to many Machine Learning algorithms such as K-Nearest-Neighbor, Clustering etc. Depending on the nature of the data points, and understanding the relationship among different distance measures is helpful in choosing a proper one for a particular application.

3.2 Mathematical Aspect of Metrics

Now that the deciding or 'discriminative' features have been singled out, the next step involves calculating the difference (or similarity) between the query and inherent parameters and each hotel/property. This distance between the two can be calculated in numerous ways - and depending upon the dataset characteristics and application, the ML model (prediction or clustering) to be applied on a later stage, the appropriate distance metric is finalised after analysing the following options:

1. **Euclidean**

$$d = \sqrt{\sum_{i=1}^{N} (Xi - Yi)^2}$$

When? Most widely used metric in almost all clustering algorithms (KMeans, KMeans++), Euclidean dist is also known as L2 norm.

2. **Manhattan Distance**

$$D = \sum_{i=1}^{n} |x_i - y_i|$$

When? Used when a grid-like path is followed, so that the differences in the individual dimensions can be added up.

3. **Pearson Correlation Coeff. (PCC)**

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

When? Pearson Correlation measures the similarity in shape between two profiles. But it only holds true when the relationship is linear and continuous.

4. **Chebychev Distance**

$$D_{\text{Chebyshev}}(p, q) := \max_i(|p_i - q_i|).$$

When? The Chebychev distance may be appropriate if the difference between points is reflected more by differences in individual dimensions (i.e. more prominent in one dimension) rather than all the dimensions considered together.

5. **Canberra Distance**

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|}$$

When? When data is scattered around an origin; therefore is very sensitive to data values close to 0 (sensitive to proportional instead of absolute differences). Used to detect outliers and deviations from the usual trend.

6. **Cosine similarity**

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

When? Ranking of documents (or any results for that matter) according to their relevance to a query; in Information Retrieval Systems.

CHAPTER 4 The ML Approach

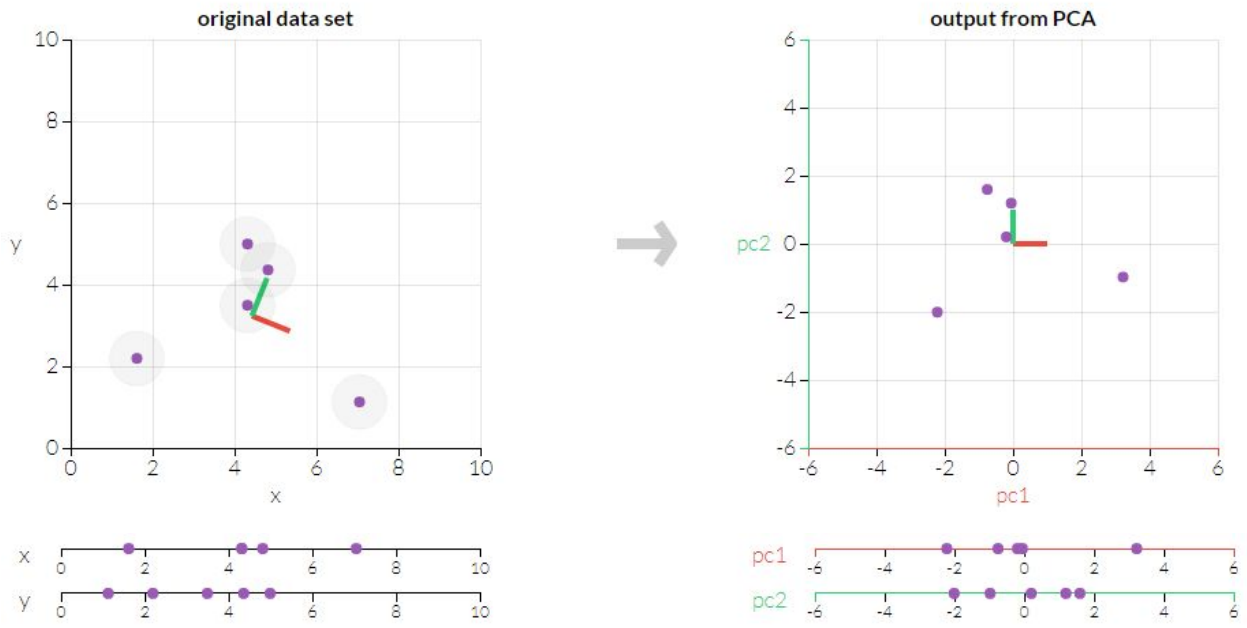
4.1 Principal Component Analysis

With a huge number of variables (or axes), the scattering network might be too huge to consider and decipher legitimately. There would be excessively numerous pairwise relationships between the factors to consider. Graphical show of information may likewise not be of specific help on the off chance that the informational index is substantial. With 12 variables, for instance, there will be more than 200 three-dimensional scatterplots to be considered.
To translate the information in a more significant frame, it is in this manner important to diminish the quantity of variables to a couple of, interpretable direct blends of the information. Each straight blend will relate to an essential segment.

Principal Component Analysis (PCA) is, thus a procedure used to underscore variety and bring out solid trends in a dataset. It's frequently used to make information simple to investigate and picture.

The objective is to reduce the dimensionality of the original dataset and pack the information with the end goal that by choosing a subset of factors from a bigger set, in view of which unique factors have the most noteworthy relationships with the central segment.

For understanding the idea, taking a straightforward illustration. Consider a dataset in just two measurements, similar to (height, weight). This dataset can be plotted as points in a plane. Be that as it may, in the event that we need to coax out variety, PCA finds another arrange framework in which each point has another (x,y) coordinates. These axes don't really mean anything physical; they're blends of height and weight called "principal components" that are given one axes loads of variety, as shown in the figure below:



The primary (first PC) principal component represents however much of the changeability in the information as could be expected, and each succeeding component represents however much of the rest of the variability as could be expected.

PCA is like other multivariate methodology called Factor Analysis. They are frequently confounded and numerous researchers don't comprehend the distinction between the two techniques or what sorts of examinations they are each most appropriate. Although for the sake of the current project, PCA suffices to be able to reduce the huge number of variables (i.e. 149 latent fields) in the *destination.csv* dataset to ultimately 3 axes.

4.2 Evaluation by Mean Average Precision

The submission for the Kaggle competition required the output format for the project to be in a particular format. For each user search query, the goal was to anticipate a space-delimited list of the hotel groups they could book. There might be up to 5 forecasts for every such event. The record ought to contain a header and have the following format:

```
id,hotel_cluster
0,99 3 1 75 20
1,2 50 30 23 9
etc...
```

Submissions are evaluated according to MAP @ 5, whose formula is:

$$MAP@5 = \frac{1}{|U|}\sum_{u=1}^{|U|}\sum_{k=1}^{min(5,n)} P(k)$$

Where P(k) is the precision at cutoff k, n is the no. of predicted hotel clusters till now and |U| is the number of user events.

Intuition behind MAP

Average Precision (AP), all the more ordinarily, additionally found the middle value of over all queries and presented as a single score — Mean Average Precision (MAP) — is an exceptionally well known performance measure in IR (Information Retrieval).

As simple and intuitive as it seems, the MAP value cannot be compared or converted to efficiency or precision in percentage. Taking an example of how MAP is calculated - suppose we are querying for images of a flower and we provide our IR system an example image of a rose (query), we do get back a bunch of ranked images (from most likely to least likely) although not all of them are correct. Hence we compute the precision at every correctly returned image, and then take an average as - if our returned result is 1, 0, 0, 1, 1, 1 - where 1 is an image of a flower, while 0 not, then the precision at every correct point is: how many correct images have been encountered up to this point (including current) divided by the total images seen up to this point. That is: 1/1, 0, 0, 2/4, 3/5, 4/6 - which amounts to 2.766/6 = 0.461.

The reason MAP values cannot be converted to accuracy in percentage is clear now - even though 4 out of 6 results are correct (more than 50% statistically) that MAP is not >0.5 as wrong images are higher up in the retrieved results. Hence it can be said that Mean Average Precision encompasses not only the accuracy, but also the relative ranking of the retrieved results.

4.3 Constructing Benchmark

The normal estimation of the MAP-5 assessment score for a yield rundown of five hotel clusters delivered by random speculation is one benchmark for assessing our work. We compute this benchmark as described - assume we were arbitrarily choosing 5 bunches for each testing case, at each position in our last 5 guesses. We get more score the nearer the top of our guess the right bunch is, and 0 on the off chance that it is absent. This implies our MAP-5 score for irregular speculating would be:

$$\text{MAP-5}_{\text{random}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{5}\frac{^{100}P_4}{^{100}P_5}\text{score}_j$$

In the above, n is the number of testing illustrations, and score j maps to the score granted for effectively speculating the right cluster in the jth position. Note that the 100 originates from the way that there are 100 conceivable hotel clusters to be assigned. We get 5 points added to the score if the right guess is in the first position,..., 1 in the event that it is in the last position, and 0 otherwise.

So we can characterize score j as takes after:

$$\text{score}_j = 5 - j + 1 = 6 - j$$

Substituting, we get:

$$\text{MAP-5}_{\text{random}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{5}\frac{^{100}P_4}{^{100}P_5}(6-j) = \sum_{j=1}^{5}\frac{^{100}P_4}{^{100}P_5}(6-j)$$

Solving it we get the benchmark value for random guessing to be ~0.15625.

4.4 Further Improvements - Exploiting Data Leaks

Fiddling around with different models and the dataset itself, we came across an interesting pattern amongst some of the columns and the output hotel cluster labels. As it turns out, about one third of the holdout data suffers from this apparent “leak” that deals with the orig_destination_distance attribute. One can find *hotel_clusters* for the affected rows by matching rows from the train dataset based on the following columns: *user_location_country, user_location_region, user_location_city, hotel_market* and *orig_destination_distance* without applying any model - it’s a linear or direct match.

However, this will not be 100% accurate because hotels can change cluster assignments (hotels popularity and price have seasonal characteristics). It is for this reason that Expedia has confirmed this in a forum [link in appendix] but not tried to rectify it.

Apart from exploiting the data leaks or loopholes in the dataset (that seems like a workaround), following ways can be employed to further improve the code and results quality:

1. Sampling down the information significantly more.

2. Parallelizing operations over different processor cores.
3. Utilizing Spark or different tools where tasks can be kept running on parallel threads.
4. Investigating different approaches to compose code and benchmarking to locate the most proficient approach.
5. Abstaining from repeating over the full testing & training sets, and rather utilizing aggregated sets (or groups).

CHAPTER F EPILOGUE

F.1 SUMMARY

The objective of this project is to foresee which of the 100 hotel groups that a random Expedia guest will book a hotel from. The application of this project is to enable Expedia to give the ideal customized hotel proposals for the client in view of a client booking, which will build the quantity of hotels booked through Expedia and all the while increment client satisfaction.

Limitations - Yet in its nascent stage, the application has a few limitations. As we are implementing this based on the dataset provided from Expedia which contains about 30 million records, it requires high computing power to process 30 million records for a good efficiency. Our systems could not process all the 30 million records to train our model. We have just trained the model on 10,000 records due to CPU and memory restrictions. In future, If provided with such computing power, we can implement neural network model which would help us further increase the accuracy.

F.2 CONCLUSION

Of the numerous algorithms, we have used collaborative filtering as it provides the best results. The choice of language is Python as it is an open source scripting language which leads to huge community support and scope for future enhancements. Another advantage it provides is the portability and robustness which is essential for such huge enterprises.

REFERENCES

[1] *An Efficient k-Means Clustering Algorithm: Analysis and Implementation* - Tapas Kanungo, Senior Member, IEEE, David M. Mount, Member, IEEE, Nathan S. Netanyahu, Member, IEEE, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu, Senior Member, IEEE (IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 7, JULY 2002)

[2] *Survey of Clustering Algorithms* - Rui Xu, Student Member, IEEE and Donald Wunsch II, Fellow, IEEE (IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 16, NO. 3, MAY 2005)

[3] *Feature Selection for Ranking (2007)* - Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. - SIGIR '07 Proceedings of the 30th annual international ACM SIGIR conference on Research and Development in Information Retrieval

[4] *Comparative study of distance metrics* - http://www.ijettcs.org/NCASG-2013/NCASG%2038.pdf

[5] *Predicting results to user search queries* - http://cs229.stanford.edu/proj2016spr/report/065.pdf

APPENDIX - A

LIST OF USEFUL WEBSITES

[A] Principal Component Analysis visualisation - http://setosa.io/ev/principal-component-analysis/

[B] Understanding the Expedia Dataset - https://www.dataquest.io/blog/kaggletutorial/

[C] Measuring Similarity and Distance - https://dzone.com/articles/machine-learning-measuring

■ ■ ■