Cosine Similarity Function For The Temporal Dynamic Web Data

Meghna Khatri
Department of computer science and engineering
Maharishi Dayanand University
Rohtak, India
Meghna823@gmail.com

Abstract— **Cosine similarity function is one of the most popular similarity function for handling the web data in various applications such as recommender system, collaborative filtering algorithms, classification algorithms, etc. Cosine similarity function calculates the similarity between the items as an angle between the vectors representing the two items. The smaller angle is for more similar items. In this paper, a new cosine similarity function is proposed that incorporates the temporal dynamic nature of the web data when calculating the similarity and provide more accurate results.**

Keywords- Cosine similarity function, temporal dynamics, web data.

I. INTRODUCTION

The amount of information is continuously increasing and changing on the web. As more and more information services move onto web, the data on the web is likely to have an exponential growth. How people can use their limited time to get the information of their interest has become an important issue in our daily life. The cosine similarity function is a very popular function when calculating the similarity between the items.

Most existing approaches often ignore the dimension of time and assume that the users' and the items' characteristics are static. While such assumption is acceptable for relatively short time periods such as days or weeks, it becomes rather unreasonable for longer time periods during which important factors affecting recommendation decisions such as a users' interests or a movie's popularity can vary significantly.
There are many causes of such causes of such changes. Firstly, a user's interests or tastes often change over time. Secondly, external events such as holidays could lead to abrupt increase in the popularity of certain items such as comedies. Thirdly, as time goes by, the recommender system itself may went through changes such as reorganizing its catalogue or introducing new search or linking features, which may improve the accessibility of some items. Finally, a user's behaviour often exhibit temporal locality. For example, if a person enjoyed a particular movie, he will often try to find related movies by the same directors/actors or of the same genre.

The dataset consisting of information about the rating of an item by a user is explored on the platform to include the temporal information when trying to find users of similar interest so as to depict the recent interests.

To do so, a time factor that can provide weightage to the ratings in the data set according to when they were rated is to be incorporated at the time of calculating similarity between the users. There are various similarity measures available for similarity computation

In order to inculcate all this temporal information, we introduce a time function that gives weightage to the ratings according or in an order they were rated. The expected output is that the new function will perform better and will provide better results when evaluated against the various evaluation metrices. .

There are several functions for computing similarity. This is done by defining an appropriate similarity or distance measure. The simplest and most common example of a distance measure is the Euclidean distance:

$$d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

where n is the number of dimensions (attributes) and x_k and y_k are the *kth* attributes (components) of data objects x and y, respectively.

The Minkowski Distance is a generalization of Euclidean Distance:

$$d(x,y) = (\sum_{k=1}^{n} |x_k - y_k|^r)^{\frac{1}{r}}$$

where *r* is the degree of the distance.
Depending on the value of *r*, the generic Minkowski distance is known with specific names: For *r* = 1, the *city block*, (*Manhattan*, *taxicab* or *L1 norm*) distance; for *r* = 2, the *Euclidean* distance; for $r \to \infty$, the *supremum* (*Lmax norm* or $L\infty$ *norm*) distance, which corresponds to computing the maximum difference between any dimension of the data objects.

The Mahalanobis distance is defined as:

$$d(x,y) = \sqrt{(x-y)\sigma^{-1}(x-y)^T}$$

where σ is the covariance matrix of the data.

Another very common approach is to consider items as document vectors of an n-dimensional space and compute their similarity as the cosine of the angle that they form:

$$cos(x,y) = \frac{(x \bullet y)}{||x||||y||}$$

where • indicates vector dot product and ||*x*|| is the norm of vector *x*.

This similarity is known as the *cosine similarity* or the *L2 Norm.* The similarity between items can also be given by their *correlation* which measures the linear relationship between objects. While there are several correlation coefficients that may be applied, the *Pearson correlation* is the most commonly used.

Given the covariance of data points *x* and *y* Σ, and their standard deviation σ, we compute the Pearson correlation using:

$$Pearson(x,y) = \frac{\Sigma(x,y)}{\sigma_x \times \sigma_y}$$

Out of all these, we have chosen Cosine similarity as our similarity measure that has proved significant results.
The cosine of 0 is 1, and less than 1 for any other angle; the lowest value of the cosine is -1. The cosine of the angle between two vectors thus determines whether two vectors are pointing in roughly the same direction. The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 usually indicating independence, and in-between values indicating intermediate similarity or dissimilarity.

II. LITERATURE REVIEW

Cosine similarity comes from information retrieval research and is used in systems with simple user profile representation [1] [2] [3]. The recommender systems applied to cosine similarity approach treat the index (user profile) and the search query (new item) as n-dimensional vectors. The cosine formula calculates the cosine of the angle between the two vectors. As the cosine approaches 1, the two vectors become coincident. If the two vectors are totally unrelated, the value of the cosine is 0.

SIFT [3] is the representative of the cosine similarity class of content-based filtering approaches in filtering USENET Netnews. It represents a user profile and an article as a weighted vector of keywords. Matching of the profile and an article occurs with respect to the cosine similarity of the vector space model.
In the approach [3], items are thought of as vectors in the m dimensional user-space where the dimension is the attribute by which the item is rated. The cosine of the angle between the vectors that represent two items is their similarity (see Figure 2). We know from calculus the dot product formula:

$$\vec{i} \cdot \vec{j} = ||\vec{i}|| \cdot ||\vec{j}|| \cdot \cos\Theta$$

$$\Longrightarrow sim(i,j) = \cos\Theta = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}|| \cdot ||\vec{j}||}$$

Similarity (closeness) is defined by data analysis in a term of a distance. The first step in neighbourhood based prediction collaborative filtering algorithm is to weight all users with respect to similarity with the active user. Say in the case of application of collaborative filtering in recommender system when you are given recommendations for movies or books, etc. You are likely to trust those that come from people who have historically proven themselves as providers of accurate and relevant recommendations. Several different similarity weighing measures have been used. The vector similarity 'cosine'measure, has proven successful in information retrieval [4].

Work by Lathia, N. et al [5] gives an outline of a method of how to depict user-similarity over time. In order to incorporate time factor, the user rating is sorted according to when they were input and then simulate a system that iteratively updates (every μ days). Beginning at time (t=ε) all ratings before ε are used to train the algorithm and test on all ratings input before the next update, at time (ε+μ). This process is repeated for each time t, incrementing by μ at each step. At each step, what was previously tested on becomes incorporated into the training set and simulated on the system.

Time-changing baseline predictors was given by Koren, Y [6] in which it is proposed to include the temporal variability within the baseline predictors through two major temporal effects. First is addressing the fact that an item's popularity is changing over time. For example, movies can go in and out of popularity as triggered by external events such as the appearance of an actor in a new movie. This is manifested in our models by the fact that item bias bi will not be a constant but a function that changes over time. The second major temporal effect is related to user biases - users change their baseline ratings over time. For example, a user who tended to rate an average movie "4 stars", may now rate such a movie "3 stars", for various reasons explained earlier. Hence, in our models we would like to take the parameter b_u as a function of time. This induces the following template for a time sensitive baseline predictor:

$$b_u i(t) = \mu + b_u(t) + b_i(t)$$

The function $b_{ui}(t)$ represents the baseline estimate for u's rating of i at day t. Here, $b_u(t)$ and $b_i(t)$ are real valued functions that change over time.
More work by Koren, Y., Bell, R., [7] which gives a more detailed approach of capturing temporal dynamics with the baseline predictors and also states more prediction rules.
In another work by Lathia, N., [9] a new approach is proposed. They say that by minimizing the mean error produced when predicting hidden user ratings and also if we adopt an approach of adaptive neighbourhoods [9] then root mean square error is considered to be a criterion for including temporal factor.
Another approach of implicit feedback is given by Lee, T.Q., Park, T., [10] which proposes to give the pseudo ratings matrix an entry '1' as a rating value when a user u purchases. A Time-based Pseudo Rating Matrix is generated where two kinds of temporal information are incorporated - the time when the item was launched and the time when the user purchased an item - into the simple pseudo rating matrix. Two observations are taken:
• More recent purchases better reflect a user's current preference.
• Recently launched items appeal more to users.
Based on these observations, they define a rating function *w* that computes rating values (rather than simply assigning 1) as follows:

w (pi, lj) = The rating value when an item with launch time lj was purchased at time pi.

In the work [11], we show that temporal diversity is an important facet of recommender systems, by showing how CF data changes over time and performing a user survey. We then evaluate three CF algorithms from the point of view of the diversity in the sequence of recommendation lists they produce over time. We examine how a number of characteristics of user rating patterns (including profile size and time between ratings) affect diversity. We then propose and evaluate set methods that maximise temporal recommendation diversity without extensively penalising accuracy.
[11], Time-aware recommender systems (TARS) are systems that take into account a time factor - the age of the user data. There are three approaches for using a time factor: (1) the user data may be given different weights by their age, (2) it may be treated as a step in a biological process and (3) it may be compared in different time frames to find a significant pattern. This research deals with the latter approach. When dividing the data into several time frames, matching users becomes more difficult – similarity between users that was once identified in the total time frame may disappear when trying to match between them in smaller time frames.
The user matching problem is largely affected by the sparsity problem, which is well known in the recommender system literature. Sparsity occurs where the actual interactions between users and data items is much smaller in comparison to the entire collection of possible interactions. The sparsity grows as the data is

split into several time frames for comparison. As sparsity grows, matching similar users in different time frames becomes harder, increasing the need for finding relevant neighbouring users. The research suggests a flexible solution for dealing with the similarity limitation of current methods. To overcome the similarity problem, we suggest dividing items into multiple features. Using these features we extract several user interests, which can be compared among users.

III. COSINE SIMILARITY FUNCTION FOR TEMPORAL DYNAMICS OF WEB DATA

Most existing approaches often ignore the dimension of time and assume that the users' and the items' characteristics are static. While such assumption is acceptable for relatively short time periods such as days or weeks, it becomes rather unreasonable for longer time periods during which important factors affecting recommendation decisions such as a users' interests or a movie's popularity can vary significantly

Temporal information may be included in the form of time weightage function when calculating the similarity among the users-items in the process of similarity computation. Hence; we wish to explore this area so we can obtain an algorithm involving a new similarity function to handle the dynamics of web data.

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

Where t is the time factor

As the users' feedback in the form of ratings, clicks, etc. is collected over a long period of time and it is very common for a users' interest or an items' popularity to change over such a long period of time. By including the time factor in the form of weightage to the ratings, we can distinguish between the current and the older data. More recent data is given more weightage in the form of a time-factor't'. Smaller value of't' suggests more weightage. By doing so, we are able to include more recent data when calculating the similarity rather than very old data. So the recent trends are reflected better and the accuracy of the cosine similarity function is improved.

IV. CONCLUSION

Cosine similarity computation function is quite popular for the web-based applications and has also been proven to be successful when handling the web data. Yet, scope of further research is left as the data is highly dynamic in nature as well as its volume is increasing explosively.

Most existing approaches often ignore the dimension of time and assume that the users' and the items' characteristics are static. Temporal information may be included in the form of time weightage function when calculating the similarity among the users-items in the process of CF. Hence a new cosine similarity function is obtained to handle the dynamics of web data.

As the users' feedback in the form of ratings, clicks, etc. is collected over a long period of time and it is very common for a users' interest or an items' popularity to change over such a long period of time. By including the time factor in the form of weightage to the ratings, we can distinguish between the current and the older data. More recent data is given more weightage in the form of a time-factor't'. By doing so, we are able to include more recent data when calculating the similarity rather than very old data. So the recent trends are reflected better and the accuracy of the cosine similarity function is improved.

REFERENCES

[1] Ungar, L. H., and Foster, D. P. Clustering Methods for Collaborative Filtering. In Workshop on Recommender Systems at the 15th National Conference on Arti_cial Intelligence. 295, 1998

[2] P. Melville, R. J. Mooney, and R. Nagarajan, "Contentboosted collaborative filtering for improved recommendations," in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02)*, pp. 187–192, Edmonton,Canada, 2002.

[3] Dhoha Almazro, A Survey Paper on Recommender Systems, 2010.

[4] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pages 230–237, New York, NY, USA, 1999. ACM Press.

[5] Spertus, E., Sahami, M., and Buyukkokten, O., Evaluating similarity measures: A large-scale study in the orkut social network. In *Proceedings of the 2005 International Conference on Knowledge Discovery and Data Mining (KDD-05)*, 2005.

[6] G.Salton and C. Buckley.Term weighting approaches in automatic text retrieval. Information Processing and management, 24(5):513-523, 1988.

[7] Basu, C., Hirsh, H., and Cohen, W. Recommendation as Classi_cation: Using Social and Content-based Information in Recommendation. In Recommender System Workshop'98. pp. 11-15. (1998).

[8] Nathan N Liu, Min Zhao, Evan Xiang, Qiang Yang, Online Evolutionary Collaborative Filtering ACM, **Pages:** 95-102,2010

[9] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, 2003

[10] G. Shani, D. Heckerman, and R. I. Brafman, "AnMDP-based recommender system," *Journal ofMachine Learning Research*, vol. 6, pp. 1265–1295, 2005.

[11] Lathia N., Hailes S., Capra L., Amatriain X., Temporal Diversity in Recommender Systems, SIGIR May, 2010.