Applying Subsequence Matching to Collaborative Filtering

Extended Abstract

Alejandro Bellogín
Universidad Autónoma de Madrid
Madrid, Spain
alejandro.bellogin@uam.es

Pablo Sánchez
Universidad Autónoma de Madrid
Madrid, Spain
pablo.sanchezp@predoc.uam.es

ABSTRACT

Neighbourhood-based approaches, although they are one of the most popular strategies in the recommender systems area, continue using classic similarities that leave aside the sequential information of the users interactions. In this extended abstract, we summarise the main contributions of our previous work where we proposed to use the Longest Common Subsequence algorithm as a similarity measure between users, by adapting it to the recommender systems context and proposing a mechanism to transform users interactions into sequences. Furthermore, we also introduced some modifications on the original LCS algorithm to allow non-exact matchings between users and to bound the similarities obtained in the [0,1] interval. Our reported results showed that our LCS-based similarity was able to outperform different state-of-the-art recommenders in two datasets in both ranking and novelty and diversity metrics.

ACM Reference Format:
Alejandro Bellogín and Pablo Sánchez. 2018. Applying Subsequence Matching to Collaborative Filtering: Extended Abstract. In *CERI 18: 5th Spanish Conference in Information Retrieval, June 26–27, 2018, Zaragoza, Spain.* ACM, New York, NY, USA, Article 5, 2 pages. https://doi.org/10.1145/3230599.3230605

1 INTRODUCTION

Neighbourhood-based approximations remain one of the most widespread strategies for modelling Recommender Systems due to their simplicity, justifiability, and relatively good performance [6]. The most critical component of this type of recommenders is usually the similarity function to be considered between users or items, where typically the Cosine similarity or the Pearson correlation are used in the literature [1, 2]. Nevertheless these classic similarities work with users as vectors applying mathematical operations between them (so the users with lower distances or higher correlations will be selected as neighbours) leaving aside other important information that can also be derived from the data.

As an alternative, we can further analyse the user profiles and treat them as "sequences" of interactions allowing us to observe the behaviour of the recommender algorithms using different ways to sort the items and capture interaction patterns. With this new definition, we aim to define a new similarity metric to be used on the top of a neighbourhood-based recommender.

In [3], we proposed to use the Longest Common Subsequence (LCS) algorithm – once it is adapted to the Recommenders Systems area – to be used as a similarity metric between users. In that paper we also presented a method to transform the user ratings into sequences of interactions in order to be interpretable for the LCS algorithm. Besides, we proposed a normalisation technique to obtain values in the valid range for a similarity metric, i.e., between 0 and 1, and we also compared our LCS-based similarity metric against other state-of-the art recommenders in two well-known datasets. In this paper we aim to summarise the main contributions of our previous work.

2 APPLYING SUBSEQUENCE MATCHING TO COLLABORATIVE FILTERING

The original Longest Common Subsequence problem is defined as follows: given two sequences x and y with lengths m and n over an alphabet $\Sigma = (\sigma_1, \cdots, \sigma_s)$, the objective is to find a subsequence (a common sequence of x and y obtained by deleting some or no elements of the sequences without changing the order) whose length is the maximum possible. In order to obtain the length of the longest subsequence, the most extended approach is to fill a matrix $C_{m+1 \times n+1}$ with this formula:

$$C[i,j] = \begin{cases} 0 & \text{if i = 0 or j = 0} \\ C[i-1,j-1]+1 & \text{if } i,j>0 \text{ and } x_i = y_j \\ \max(C[i,j-1], C[i-1,j]) & \text{if } i,j>0 \text{ and } x_i \neq y_j \end{cases} \quad (1)$$

Although the standard definition of the LCS problem is only defined between strings or sequences of characters, if we define a set of symbols to represent the user interactions, we may compute the LCS between two users and use this value as their similarity. In order to clarify the full process, let us denote with $\mathcal{U}$ and $\mathcal{I}$ the set of users and items in the system, and $u, v \in \mathcal{U}$ two particular users. We will also use $I(u) = \{(i,r) : (u,i,r) \mid r \neq \emptyset\}$ to indicate the set of items rated by a user, where r denotes the rating the user gave to the item (normally between 1 and 5). Hence, to generate a string/sequence representation of user interactions, it makes sense to use the set $I(u)$ of rated items; nonetheless, there are multiple alternatives to transform $I(u)$ into symbols, depending on the considered alphabet Σ. In [3] we proposed the following transformation functions f:

- Using the item, i.e., $f_i : I(u) \to \Sigma = \mathcal{I}$, $f_i(x) = x(i)$.
- Using the interaction value, i.e., $f_r : I(u) \to \mathcal{R}$, $f_r(x) = x(r)$.
- Using a combination of the item and the interaction value, i.e., $f_{ir} : I(u) \to \mathcal{I} \times \mathcal{R}$, $f_{ir}(x) = (x(i), x(r))$.

where $x(i)$ and $x(r)$ denote the functions that return the id of item i and the rating user u gave to item i. Since typically ratings and item

CERI 18, June 26–27, 2018, Zaragoza, Spain

ACM ISBN 978-1-4503-6543-7/18/06…$15.00
https://doi.org/10.1145/3230599.3230605

ids are numbers, our alphabet Σ will be the set of integers. For the f_{ir} transformation, in order to obtain a unique symbol every pair of item and rating, we propose to use $f_{ir} = x(i) \cdot 10 + x(r)$, assuming $1 \leq x(r) \leq 9$. Even though this is a pure collaborative filtering approach, our framework also opens up the possibility of creating symbols using the features associated for the items (like genres in the case of books or movies) to create a hybrid recommender.

Once we have a user transformed into a series of symbols, we need to specify the order in which those symbols will be arranged. This is a critical step as the LCS algorithm is aware of the order of the two strings. As a first approach, in [3] we sorted each user sequence by the item id, leaving as a future work a temporal ordering of the ratings. Note that building time-based sequences of user preferences is often not a trivial question, as some datasets do not contain meaningful timestamps [4, 5].

In this context, it is important to note that an exact matching is sought between the two strings in the original LCS problem; however, when dealing with user data some level of fuzziness is desired, since it is usually assumed that two users do not need to show an identical behaviour to be considered good predictors of each other. Classical similarity measures like Pearson or Cosine address this issue by considering the distance between the values provided by each user. Because of this, we propose a variation on the LCS algorithm to relax the matching condition, in such a way that a matching threshold δ is used to decide whether two symbols of the sequence are equivalent. Recalling Formula 1, the second expression is modified in the following way:

$$C[i,j] = C[i-1,j-1] + 1 \quad \text{if } i,j > 0 \text{ and } match(x_i, y_j, \delta) \tag{2}$$

where this δ-*matching* is produced when the symbols obtained differ at most by δ units.

After this process, LCS will produce values between 0 and $min(m,n)$, where m and n are the lengths of the sequences we are comparing. However, classic similarity metrics obtain bounded values in ranges $[-1,1]$ or $[0,1]$. Following the same rationale, we propose to normalise the LCS value between two users (considering a transformation function f and matching threshold δ, denoted as LCS_CF(u,v,f,δ)) using the length of the two sequences involved:

$$\text{sim}_1^{f,\delta}(u,v) = \text{LCS\_CF}(u,v,f,\delta) \tag{3}$$

$$\text{sim}_2^{f,\delta}(u,v) = \text{sim}_1^{f,\delta}(u,v)^2 / (|f(u)| \cdot |f(v)|) \tag{4}$$

In this way, $\text{sim}_1^{f,\delta}(u,v) \in [0, |\Sigma|]$, whereas $\text{sim}_2^{f,\delta}(u,v) \in [0,1]$. Furthermore, when $f = f_i$, $\delta = 0$, and we use an ordering based on the item id, Equation 4 is ranking-equivalent to binary Cosine $\cos_b(u,v) = |I(u,v)|/\sqrt{|I(u)| \cdot |I(v)|}$, which demonstrates the generality of our proposed adaptation.

3 EXPERIMENTS

In order to test the performance of our LCS-based recommender, in [3] we performed experiments not only against other similarity metrics in a user-based nearest-neighbour scenario but also against other state-of-the-art recommenders like the popularity recommender, a matrix factorisation technique and other item-based neighbourhood recommenders. The experiments were performed under the RankSys and Mahout frameworks in two different datasets (MovieLens and Lastfm), where we analysed ranking accuracy metrics like Precision, Recall, nDCG, and Mean Average Precision (MAP) and novelty and diversity metrics like EPC, EPD, Aggregate Diversity and Gini.

According to the results obtained, our proposal showed competitive results in terms of precision and novelty and diversity metrics, being able to outperform in both datasets many of the baselines. We also found that applying the normalised version of LCS (Formula 4) produced better result in terms of accuracy metrics.

As a consequence of these results, we have continued working on extending the potential applications of LCS as a similarity metric for recommendation. In particular, we incorporate additional information to be exploited in the user sequences, such as the temporal context (by ordering the sequences by the interaction timestamp) and the item features (where we generate the user sequences also using the content-based information from the items). We found promising results in both cases, not only because our LCS-based approaches were able to obtain better performance than other baselines under different settings, but also because the LCS algorithm proved to be easily adaptable when working with different sources of information.

4 CONCLUSIONS

Research into recommender systems remains active due to their great popularity in a large number of Internet applications. In this extended abstract, we have presented a new similarity metric to be used in neighbourhood-based recommenders based on the LCS algorithm. We have also shown a generic method to transform the users preferences to generate sequences of interactions and also some modifications to the algorithm to allow non-exact matchings (δ), together with a normalisation function to bound the values obtained. Although our results are promising, we believe there is still room to add more modifications to the LCS algorithm, not only in the recommendation step (e.g., an adaptive LCS that gives more weight to more recently scored items or some combinations of LCS with Markov Chains) but also when evaluating the recommendations, as we can use the LCS algorithm to evaluate the quality of recommendations.

ACKNOWLEDGMENTS

This work was supported by the research project TIN2016-80630-P (MINECO).

REFERENCES

[1] Fabio Aiolli. 2013. Efficient top-n recommendation for very large scale binary rated datasets. In *RecSys*. ACM, 273–280.
[2] Alejandro Bellogín and Arjen P. de Vries. 2013. Understanding Similarity Metrics in Neighbour-based Recommender Systems. In *ICTIR*. ACM, 13.
[3] Alejandro Bellogín and Pablo Sánchez. 2017. Collaborative filtering based on subsequence matching: A new approach. *Inf. Sci.* 418 (2017), 432–446.
[4] Pedro G. Campos, Fernando Díez, and Iván Cantador. 2014. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Model. User-Adapt. Interact.* 24, 1-2 (2014), 67–119.
[5] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *TiiS* 5, 4 (2016), 19:1–19:19.
[6] Xia Ning, Christian Desrosiers, and George Karypis. 2015. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In *Recommender Systems Handbook*. Springer, 37–76.