POLITECNICO DI TORINO
Repository ISTITUZIONALE

Sequeval: A framework to assess and benchmark sequence-based recommender systems

Original
Sequeval: A framework to assess and benchmark sequence-based recommender systems / Monti, D.M., Palumbo, E., Rizzo, G., Morisio, M.. - ELETTRONICO. - (2018). (12th ACM Conference on Recommender Systems Vancouver (CA) 2nd-7th October 2018).

Availability:
This version is available at: 11583/2712372 since: 2018-10-15T13:34:57Z

Publisher:
Organizers of REVEAL 2018

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

08 September 2026

Sequeval: A Framework to Assess and Benchmark Sequence-based Recommender Systems

Diego Monti
Politecnico di Torino
diego.monti@polito.it

Enrico Palumbo
ISMB-EURECOM
palumbo@ismb.it

Giuseppe Rizzo
ISMB
giuseppe.rizzo@ismb.it

Maurizio Morisio
Politecnico di Torino
maurizio.morisio@polito.it

ABSTRACT

In this paper, we present `sequeval`, a software tool capable of performing the offline evaluation of a recommender system designed to suggest a sequence of items. A sequence-based recommender is trained considering the sequences already available in the system and its purpose is to generate a personalized sequence starting from an initial seed. This tool automatically evaluates the sequence-based recommender considering a comprehensive set of eight different metrics adapted to the sequential scenario. `sequeval` has been developed following the best practices of software extensibility. For this reason, it is possible to easily integrate and evaluate novel recommendation techniques. `sequeval` is publicly available as an open source tool and it aims to become a focal point for the community to assess sequence-based recommender systems.

1 INTRODUCTION

Traditional Recommender Systems (RSs) usually do not consider the temporal dimension of user preferences when suggesting a set of items. This simplifying hypothesis may represent a limitation in domains characterized by the rapid consumption of different items one after the other. Even if some authors proposed RSs capable of considering training items as sequences [3], the idea of suggesting sequences of items instead of lists ranked by relevance is not widespread. On the other hand, this problem is quite similar to the task of generating a phrase given its initial words [5].

Consider, for example, the context of musical streaming services. Starting from the human-curated playlists already available in the system, a sequence-based RS should be capable of creating a personalized playlist for the target user given an initial seed. The initial seed may be a song, a set of songs, or a genre.

To the best of our knowledge, there is no standardized protocol for performing an offline experiment with sequence-based RSs. The lack of standardized protocols and metrics is a well-known problem inside the RS evaluation community, as it leads to research results that are difficult to compare [4].

Several software tools designed to simplify the development of novel recommendation algorithms are available. Such tools usually also provide a module devoted to their evaluation, but the results obtained are often not comparable because of the differences in how the protocols and the metrics are implemented [6].

In this paper, we present `sequeval`, a Python implementation of an evaluation framework designed for comparing sequence-based RSs. This software package is freely available on GitHub[1] and its usage is described in an introductory video.[2]

[1] https://github.com/D2KLab/sequeval
[2] https://doi.org/10.6084/m9.figshare.6854726

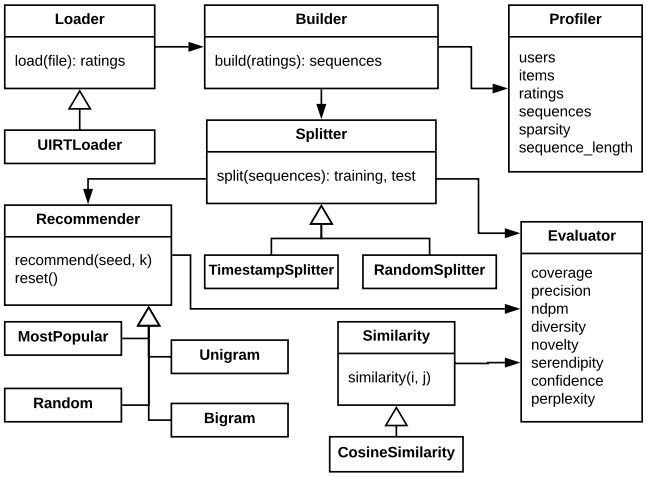

Figure 1: Simplified UML class diagram

2 SYSTEM ARCHITECTURE

`sequeval` is a Python package composed of different modules which are graphically represented in Figure 1. For each component, we defined an abstract class and then we realize one or more possible implementations in order to enable extensibility.

The *loader* module is in charge of reading an input file containing user ratings. We have implemented a concrete *loader* capable of processing a textual file in a MovieLens-like format (UIRT), but the support to other formats can be easily added. It is optionally possible to ignore users or items that do not have a minimum number of ratings, in order to avoid data sparsity issues.

The *builder* module creates the sequences of items from the initial ratings: ratings from the same user that are distant in time less then a threshold are grouped inside the same sequence. Ratings that do not belong to any sequence are discarded.

The *profiler* module computes some statistics about the generated sequences, for example their average length.

The *splitter* module assigns the sequences created by the *builder* to the training and the test sets, according to a random or to a more realistic timestamp-based strategy. It is up to the experimenter deciding the percentage of sequences in the test set.

The *recommender* module includes an abstract class that needs to be implemented by any recommender that relies on this framework. For demonstrative purposes, we have implemented four different baseline recommenders, described in Section 2.1.

The purpose of the abstract class in the *similarity* module is to compute a content-based similarity metric between two items; we have chosen to implement it as a generic cosine-based similarity.

Finally, the *evaluator* module, given the test set and a recommender, computes the measures. We have decided to rely on a comprehensive set of eight metrics, which are detailed in Section 2.2.

2.1 Recommenders

We have included in `sequeval` four baseline recommenders, which represent an adaptation of classical non-personalized baselines to the sequence-based scenario.

Most Popular The most popular recommender only considers the popularity of the items in the sequences available in the training set in order to create the recommended sequence, which is always a ranked list of the most popular items.

Random The random recommender simply creates sequences that include items randomly chosen from the ones available in the training sequences.

Unigram The unigram recommender generates sequences that contain items sampled with a probability proportional to the number of times they were observed in the training set.

Bigram The bigram recommender estimates the 1-*st* order transition probabilities among all possible pair of items available in the training sequences and it recommends sequences according to those probabilities.

2.2 Evaluation Metrics

In order to provide a comprehensive analysis of the recommended sequences, we have included in the *evaluator* module eight metrics. We decided to consider traditional metrics like coverage and precision, and also less common ones like novelty, diversity, and serendipity. We have also introduced the metric of perplexity, because it was created for evaluating sequences [1].

For each sequence in the test set, a sequence of a certain length is generated by the chosen recommender, considering the same target user and the first item of the test sequence as the initial seed. The length of the sequences is a parameter of the experiment.

In the following, we briefly describe the metrics available.

Coverage It expresses the items that the sequence-based RS is capable of suggesting when generating sequences similar to the ones available in the test set.

Precision It represents the fraction of recommended items that are included in the corresponding test sequence.

nDPM It indicates if the items in the recommended sequences are in the same order of the ones in the test sequences.

Diversity This metric exploits the cosine similarity in order to check if the items in the recommended sequences are enough diverse among them.

Novelty It rewards algorithms that are capable of suggesting sequences that contain items that are probably unknown to the user because they are not popular.

Serendipity It is similar to the metric of precision, but it is computed always considering popular items as irrelevant.

Confidence It expresses how sure the recommender is of the generated sequences and it is computed as the average probability that the suggested items are correct.

Perplexity This metric can be interpreted as the number of items from which a random recommender should choose in order to obtain a similar sequence.

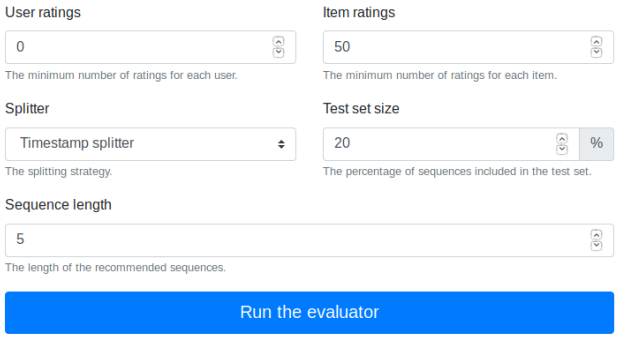

Figure 2: Screenshot of the web-based interface

3 DEMONSTRATION AND USAGE

In order to exploit the proposed framework, it is necessary to create an implementation of the abstract recommender that must be capable, given the user and the current item of the sequence, of predicting the probabilities for all the possible items of being the next one inside the recommended sequence. It is then possible to write an evaluation script that relies on this library.

For demonstrative purposes, we have included in `sequeval` a down-sampled version of the playlists dataset originally collected by Shuo Chen from the *Yes.com* website [2].

We also provide a simple evaluation script which can be used to perform different experiments with the baseline recommenders. This script can be easily modified in order to accommodate novel recommandation techniques.

Furthermore, we have realized a web-based version of the same evaluation script in order to provide a more convenient graphical interface to perform the experiments, as shown in Figure 2.

4 CONCLUSION AND FUTURE WORK

In this paper, we presented `sequeval`, a software tool for evaluating sequence-based recommender systems. The strategic choice of publishing it as an open source project fosters reuse and extension. As future work, we plan to include in `sequeval` further datasets and more complex recommendation techniques.

REFERENCES

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* 3 (2003), 1137–1155. http://www.jmlr.org/papers/v3/bengio03a.html

[2] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 714–722. https://doi.org/10.1145/2339530.2339643

[3] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM Press, 161–169. https://doi.org/10.1145/3109859.3109882

[4] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. 2015. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction* 25, 5 (2015), 427–491. https://doi.org/10.1007/s11257-015-9165-3

[5] Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing*. Prentice Hall.

[6] Alan Said and Alejandro Bellogín. 2014. Comparative recommender system evaluation. In *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM Press, 129–136. https://doi.org/10.1145/2645710.2645746