Journal of Machine Learning Research 21 (2020) 1-5 Submitted 9/19; Revised 4/20; Published 5/20

# Cornac: A Comparative Framework for Multimodal Recommender Systems

**Aghiles Salah** ASALAH@SMU.EDU.SG
**Quoc-Tuan Truong** QTTRUONG.2017@SMU.EDU.SG
**Hady W. Lauw** HADYWLAUW@SMU.EDU.SG
*Singapore Management University*
*School of Information Systems, 80 Stamford Road, Singapore 178902*

**Editor:** Andreas Mueller

## Abstract

Cornac is an open-source Python framework for multimodal recommender systems. In addition to core utilities for accessing, building, evaluating, and comparing recommender models, Cornac is distinctive in putting emphasis on recommendation models that leverage auxiliary information in the form of a social network, item textual descriptions, product images, etc. Such multimodal auxiliary data supplement user-item interactions (e.g., ratings, clicks), which tend to be sparse in practice. To facilitate broad adoption and community contribution, Cornac is publicly available at `https://github.com/PreferredAI/cornac`, and it can be installed via Anaconda or the Python Package Index (pip). Not only is it well-covered by unit tests to ensure code quality, but it is also accompanied with a detailed documentation[1], tutorials, examples, and several built-in benchmarking data sets.

**Keywords:** comparison, multimodality, recommendation algorithms, software

## 1. Introduction

A longstanding practice in recommender systems is to rely on observed preference data – such as ratings, clicks, purchases – to model and subsequently estimate unseen user-item interactions (Mnih and Salakhutdinov, 2008). One major challenge to this practice is the sparsity of preference data, which poses model estimation and generalization difficulties. In recent times, the community has taken major steps towards the promising direction of alleviating sparsity by leveraging auxiliary data, i.e., information beyond user-item interactions, which can hold additional clues on how users consume items. For instance, an item may have textual content (Wang and Blei, 2011; Wang et al., 2015), descriptive images (He and McAuley, 2016), or other related items (Salah and Lauw, 2018), while users may have social networks (Chaney et al., 2015). Auxiliary data are also referred to as *modalities*. Hereinafter, "multimodal recommender systems" broadly refer to models relying on other modalities – in addition to preference information – to improve recommendations.

Cornac is an open-source Python framework tailored for multimodal recommender systems, whose objectives are two-fold. First, to contribute towards the academic community's effort in advancing further research in this direction, via providing a simple and handy environment for developing, evaluating, and comparing various multimodal recommender al-

1. `https://cornac.readthedocs.io`

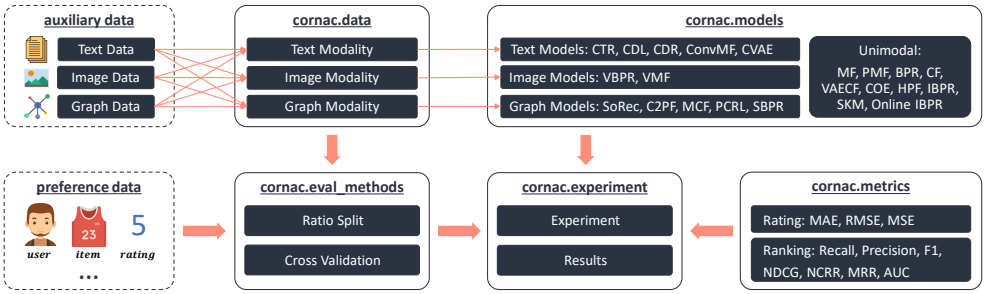


Figure 1: Cornac's infrastructure. Thick arrows represent the interaction flow between the main modules when running an experiment. Thin arrows illustrate cross-modality transformations/utilizations, more discussion on this aspect in the *Multimodality Support* paragraph.

```
import cornac

feedback = cornac.datasets.citeulike.load_feedback()  # preference data
docs, item_ids = cornac.datasets.citeulike.load_text()  # item texts

tm = cornac.data.TextModality(corpus=docs, ids=item_ids, max_vocab=8000)
rs = cornac.eval_methods.RatioSplit(data=feedback, test_size=0.2, item_text=tm)
cdl = cornac.models.CDL(k=50, autoencoder_structure=[200], max_iter=30)
recall = cornac.metrics.Recall(k=[10, 50, 100])

cornac.Experiment(eval_method=rs, models=[cdl], metrics=[recall]).run()
```

Figure 2: An example illustrating how to fit and evaluate the CDL model on CiteULike data set.

gorithms. Second, to facilitate broader adoption of multimodal recommendation algorithms by practitioners in academia and industry, via providing open access to a rich and growing collection of state-of-the-art models[2]. Towards achieving these objectives, we design Cornac based on a number of principles, namely *rich abstraction*, *low coupling*, and *cross-modality*, i.e., transformations across modalities, as well as cross-utilization of models designed for a specific modality (e.g., images) to work with a different one (e.g., text).

## 2. Framework and Key Features

Figure 1 depicts the five main modules of Cornac, while Figure 2 shows a concrete example of how to run an experiment in Cornac.

**Main Modules.** First, to handle preference and auxiliary data, `cornac.data` module provides, for different preference data formats as well as multiple types of modalities, reading, formatting, parsing, and transformation utilities. Cornac also offers ready access to popular benchmarks via `cornac.datasets` (Figure 2). Second, to access a collection of recommender models, `cornac.models` module implements both multimodal and unimodal algorithms. The third and fourth modules support the evaluation of recommendation models. `cornac.eval_methods` supports data sampling schemes, such as $k$-fold cross validation and train/validation/test splitting. `cornac.metrics` features various commonly used measures, such as Root Mean Squared Error, Recall, etc. The fifth module `cornac.experiment` brings together other modules to perform an evaluation and organize the results, emphasizing the ease of comparisons across models or settings.

2. `https://github.com/PreferredAI/cornac/blob/master/README.md#models`

**Multimodality Support.** A cornerstone of Cornac's design is to facilitate working with auxiliary data. To this end, the `cornac.data` module provides three main modality classes. `TextModality` implements the necessary routines to process raw texts and output different representations (e.g., sequences, bag-of-words) depending on specific needs. `ImageModality` supports image data (e.g., product images). `GraphModality` concerns data encoding pairwise relations between users/items. These classes encompass various auxiliary data types, e.g., social network, item images, descriptions, and reviews.

As illustrated by the thin arrows in Figure 1, Cornac enables some transformations/utilization across modalities, i.e., from Text, Image to Graph. For example, one can instantiate `GraphModality` using textual information; under the hood, a nearest-neighbor graph for items that encode their textual similarities will be constructed. Moreover, when the model of interest relies on a vector-based representation of auxiliary information, cross-utilizations of models designed for one modality (e.g., images) to work with a different modality (e.g., texts) are supported. Cornac makes this use case convenient by implementing a generic class, namely `FeatureModality`, generalizing the modality classes.

**Scalability.** To ease stochastic optimization, Cornac provides a number of iterators allowing one to perform mini-batch sampling over users, items, (user, item)-pairs, or even (user, item, negative item)-triplets for ranking-based models. Cornac also harnesses the Python ecosystem, e.g., `NumPy` and `Scipy`, for efficient random number generation and fast computations over arrays and sparse matrices, or *Cython* for reaching comparable performance to compiled languages, and seamlessly wrapping existing C/C++ implementations.

**Reproducibility.** Cornac supports reproducible research by offering open-access to existing algorithms and built-in data sets. For comparison at parity, it provides full control over random number generation, allowing one to specify random seed values at different levels – when splitting data sets into training/testing sets, initializing model parameters, or performing mini-batch sampling for stochastic optimization.

## 3. Comparative Review of Recommendation Softwares

To assess the values and competitive advantages offered by Cornac, we conduct a comparative review of existing platforms for recommender models. Notably, Cornac is a systematic and deliberate effort to facilitate the integration of auxiliary data of various modalities.

Table 1 reports some key features of several existing platforms to emphasize the commonalities and differences between them. First, we note that some of the most established open-access software for recommender systems have been written in a language other than Python. *MyMediaLite*, written in C#, includes implementations of existing algorithms and

| Frameworks | Language | Feedback Type | | Evaluation Metrics | | Multimodal Algorithms | | | Multimodality Support | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Explicit | Implicit | Rating | Ranking | Text | Graph | Image | Text | Graph | Image |
| MyMediaLite (Gantner et al., 2011) | C# | ✓ | ✓ | ✓ | ✓ | | | | | | |
| Recommenderlab (Hahsler, 2015) | R | ✓ | | ✓ | ✓ | | | | | | |
| LibRec (Guo et al., 2015) | Java | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| LightFM (Kula, 2015) | Python | ✓ | ✓ | ✓ | ✓ | ✓ | | | | | |
| TensorRec (James, 2017) | | ✓ | ✓ | ✓ | ✓ | ✓ | | | | | |
| Surprise (Hug, 2017) | | ✓ | | ✓ | ✓ | | | | | | |
| Implicit (Ben, 2016) | | | ✓ | | ✓ | | | | | | |
| Spotlight (Kula, 2017) | | ✓ | ✓ | ✓ | ✓ | | | | | | |
| OpenRec (Yang et al., 2018) | | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | | | |
| Cornac [this paper] | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 1: Qualitative comparison of recommender system frameworks

| Model | Library | AUC | Time (s) CPU | Time (s) GPU |
|---|---|---|---|---|
| BPR | LibRec | – | – | – |
| | OpenRec | 0.641 | 115.4 | 24.8 |
| | Cornac | 0.697 | 9.0 | – |
| VBPR | LibRec | – | – | – |
| | OpenRec | 0.806 | 4805.0 | 2225.4 |
| | Cornac | 0.791 | 4080.2 | 1762.0 |

| Model | Library | R@100 | Time (s) CPU | Time (s) GPU |
|---|---|---|---|---|
| WMF | LibRec | 0.343 | 10.3 | – |
| | OpenRec | 0.351 | 71.6 | 42.6 |
| | Cornac | 0.364 | 18.9 | 16.5 |
| CDL | LibRec | – | – | – |
| | OpenRec | 0.415 | 1727.7 | 1193.6 |
| | Cornac | 0.405 | 257.1 | 238.6 |

| Model | Library | RMSE | Time (s) CPU | Time (s) GPU |
|---|---|---|---|---|
| PMF | LibRec | 0.888 | 0.5 | – |
| | OpenRec | 0.954 | 23.4 | 25.0 |
| | Cornac | 0.838 | 0.8 | – |
| SoRec | LibRec | 0.820 | 1.0 | – |
| | OpenRec | – | – | – |
| | Cornac | 0.818 | 0.9 | – |

Table 2: Comparison of model implementations in terms of various metrics on relevant data sets. Higher AUC on Tradesy (left), higher recall on Citeulike (middle), and lower RMSE on FilmTrust (right) are better. Less time implies more efficient learning.

basic routines to measure recommendation performance. In turn, *LibRec* is a Java library implementing a rich collection of algorithms, several metrics, and data splitting schemes. *Recommenderlab* is an R package for developing and testing recommender algorithms.

With the rise of Python for data science applications, there is a timely need for Python-based frameworks for recommender systems. Aside from Cornac, such frameworks include *LightFM* inspired by factorization machine to model interactions between user and item features, *TensorRec* enabling non-linear interactions (e.g., using deep neural nets), *Surprise* focusing on explicit feedback (rating prediction) such as matrix factorization techniques. Analogously, *Implicit* pays attention to algorithms for implicit feedback (e.g., clicks). *Spotlight* (built on PyTorch) and *OpenRec* (built on TensorFlow) provide a collection of building blocks for loss functions and representations for fast prototyping of recommender models.

As evident from the right half of Table 1, a pivotal differentiator of Cornac is its *multimodal* support. This is expressed in two ways. For one, it features a number of multimodal algorithms covering all the three modalities of concern: *text*, *graph*, and *visual*. For another, it offers native support for this family of models. Some frameworks (e.g., *LibRec*), though not designed for multimodality specifically, include several social and text-based recommender models. Nevertheless, they leave the burden of handling auxiliary data to model-level implementations. As a limitation, their data processing pipelines are not as standardized as in Cornac. This standardization is crucial to Cornac's flexibility in enabling fast development/integration of new multimodal recommender systems, cross-modal utilization of these models, as well as objective comparisons across models and modalities.

**Quantitative Comparisons.** Table 2 reports the performance of several models, including Bayesian Personalised Ranking (BPR) and its visual extension VBPR, Weighted Matrix Factorization for implicit feedback (WMF) and its textual extension CDL, Probabilistic Matrix Factorization (PMF) and its social network extension SoRec. We compare the Cornac implementations with those from other libraries LibRec and OpenRec. For each model, we retain the hyperparameters used in its corresponding research paper, while selecting the learning rate and number of the iterations based on held-out validation set due to the difference in terms of optimization procedures. Search spaces for the latter parameters are respectively $\{1e^{-3}, 5e^{-3}, \ldots, 1e^{-1}\}$ and $\{50, 100, \ldots, 500\}$. In addition to accuracy measures, we also report the training time on both CPU and GPU environments (Intel Xeon E5-2650 v4 and Nvidia Tesla P100). Missing results (−) are due to non-availability of implementation at the point of writing, except for BPR in LibRec where the AUC evaluation did not scale (out-of-memory) to the Tradesy data set. Overall, Cornac implementations offer competitive performance while requiring less training time in most cases. Cornac also has better coverage and support in terms of different types of auxiliary data.

## Acknowledgments

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its NRF Fellowship Programme (Award No. NRF-NRFF2016-07).

## References

Frederickson Ben. Collaborative filtering for implicit datasets. `https://github.com/benfred/implicit`, 2016.

Allison JB Chaney, David M Blei, and Tina Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In *RecSys*, pages 43–50, 2015.

Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In *ACM RecSys*, 2011.

Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. Librec: A java library for recommender systems. In *UMAP Workshops*, volume 4, 2015.

Michael Hahsler. recommenderlab: A framework for developing and testing recommendation algorithms. Technical report, 2015.

Ruining He and Julian McAuley. Vbpr: Visual bayesian personalized ranking from implicit feedback. In *AAAI*, pages 144–150, 2016.

Nicolas Hug. Surprise, a Python library for recommender systems. `http://surpriselib.com`, 2017.

Kirk James. A tensorflow recommendation algorithm and framework in python. `https://github.com/jfkirk/tensorrec`, 2017.

Maciej Kula. Metadata embeddings for user and item cold-start recommendations. In Toine Bogers and Marijn Koolen, editors, *ACM RecSys Workshop.*, pages 14–21, 2015.

Maciej Kula. Spotlight. `https://github.com/maciejkula/spotlight`, 2017.

Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2008.

Aghiles Salah and Hady W. Lauw. A bayesian latent variable model of user preferences with item context. In *IJCAI*, pages 2667–2674, 2018.

Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456, 2011.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD*, pages 1235–1244, 2015.

Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh, and Deborah Estrin. Openrec: A modular framework for extensible and adaptable recommendation algorithms. In *ACM WSDM*, pages 664–672, 2018.