ClayRS: an End-to-End Framework for Reproducible Knowledge-aware Recommender Systems

Pasquale Lops, Marco Polignano, Cataldo Musto, Antonio Silletti, Giovanni Semeraro

Department of Computer Science, University of Bari Aldo Moro, Via E. Orabona 4, 70125 Bari, Apulia, Italy

Abstract

Knowledge-aware recommender systems represent one of the most innovative research directions in the area of recommender systems, aiming at giving *meaning* to information expressed in natural language and obtaining a deeper comprehension of the information conveyed by textual content.

Though rich and constantly evolving, the literature on knowledge-aware recommender systems is particularly scattered when considering software libraries. This makes it difficult to easily exploit advanced content representation and implement *replicable* experimental protocols. Accordingly, this work aims to fill in these gaps by introducing `ClayRS`, an end-to-end framework for replicable knowledge-aware recommender systems. `ClayRS` provides researchers and practitioners with the most recent state-of-the-art methodologies to build knowledge-aware content representations and also includes methods to exploit these representations in content-based recommendation algorithms. Finally, the structure of the framework also allows for building replicable pipelines to push forward the current research in the area and to develop accountable recommender systems.

Keywords: knowledge-aware recommender systems, reproducibility, graph embeddings, BERT embeddings, USE embeddings, deep learning, recommender systems

1. Introduction

Recommender Systems (RS) are personalized systems representing one of the most disruptive technologies that appeared on the scene in the last decades [1]. Such systems support users in several decision-making tasks by acquiring

*Corresponding author: Pasquale Lops

Email addresses: `pasquale.lops@uniba.it` (Pasquale Lops), `marco.polignano@uniba.it` (Marco Polignano), `cataldo.musto@uniba.it` (Cataldo Musto), `a.silletti6@studenti.uniba.it` (Antonio Silletti), `giovanni.semeraro@uniba.it` (Giovanni Semeraro)

information about their needs, interests and preferences, in order to personalize the user experience on the ground of such information. RS have a great influence on consumers' behaviors since many people use these systems to buy products, listen to music, choose restaurants, or even read the posts that Facebook has ranked at the top of our feed. RS are also very important for companies, which report that RS contribute from 10% to 30% of their total revenues [2].

In literature there are different recommendation paradigms, but choosing which is the best one to fit a specific scenario is not trivial. Usually there is not a recommendation paradigm which is universally acknowledged as the *best*. The paradigms that gained more popularity are *Collaborative Filtering* and *Content-based Filtering*. Despite the wide use of collaborative filtering, alone or in combination with other approaches, as it usually happens in deep architectures [3], the adoption of the content-based paradigm has several advantages when compared to the collaborative one:

- User independence: differently from collaborative filtering methods which rely on the ratings of the whole community of users, content-based RS are able to generate recommendations by leveraging only the ratings of the active user, and this helps to mitigate the *data sparsity* issue
- Transparency: Content-based RS rely on the match between content features or descriptions in the user profile and those in the item representations for providing explanations and making the algorithm more transparent. This differs from collaborative filtering algorithms which are mainly black boxes and whose explanations should be based on the user or item similarities or on more complex representations based on latent factors
- New item: Content-based RS do not suffer from the first-rater problem which affects collaborative RS. Indeed, they rely solely on users' preferences to make recommendations.

Nonetheless, content-based RS have several shortcomings:

- Limited content analysis: Shallow representation mechanisms based on keywords, not able to deal with the natural language ambiguity, could lead to content-based RS not able to correctly catch and represent user preferences
- Over-specialization: The matching between user profiles and items in content-based RS could lead to the suggestions of *more of the same* items, i.e. items very similar to those liked in the past, hence with a limited degree of novelty.

Accordingly, it is necessary to improve such representations in order to fully exploit the potential of content-based features and textual data by implementing more advanced models that allow machines to better understand information provided in natural language.

Knowledge-aware recommender systems represent one of the most innovative research directions in the area of recommender systems, aiming at giving *meaning* to information expressed in natural language and obtaining a deeper comprehension of the information conveyed by textual content [4]. The literature on knowledge-aware recommender systems is actually rich, and constantly evolving. Novel research directions in the area of *semantics-aware* content-based recommender systems are described in [5], which integrates and extends the content previously presented in [4], by giving an updated overview of the main techniques to incorporate semantics into items and user profiles. From now on, semantics-aware and knowledge-aware recommender systems will be used interchangeably to refer to the same family of systems.

On the other hand, when considering software libraries, the literature is also very scattered. These techniques are often implemented in different ways, without a comprehensive software library offering a common pipeline to process the textual content and implement knowledge-aware content representations. This means that every researcher needs to continuously recreate a complex pipeline to process and represent content, recommend items by exploiting those representations, and evaluate the related performance. Hence, the experimental workflow related to recommender systems that exploit complex representations for items and users is becoming more and more complex, making the *replicability* of experiments a challenge.

To tackle both these issues, in this paper we present `ClayRS`, a new end-to-end Python framework which will make the entire recommendation pipeline simple, fast, and replicable. In our vision, this work provides a common ground for both researchers and practitioners interested in the latest knowledge-aware content representations to be exploited for user modeling and recommender systems.

The paper is organized as follows: Section 2 gives an overview of (i) the challenges to develop accountable recommender systems through the replicability of the experimental evaluation, with a specific focus on knowledge-aware recommender systems; (ii) the most recent techniques for knowledge-aware content-based recommender systems, covering the approaches classified as *exogenous* and *endogenous*; (iii) a state of the art about the frameworks to implement a complete recommendation pipeline which satisfies all or part of the requirements to facilitate replicability. Section 3 introduces `ClayRS`, a novel end-to-end Python framework covering the whole pipeline for implementing and evaluating knowledge-aware content-based recommender systems. `ClayRS` allows complex representations of users and items to feed knowledge-aware content-based recommendation algorithms, and compute the performance of recommendations making the whole process customizable and replicable. Section 4 gives an overview of using `ClayRS` to configure, execute and evaluate an experimental scenario, Section 5 sketches some open challenges addressed by `ClayRS`, and Section 6 draws some conclusions and gives an outlook to future work.

2. Background and Related Work

2.1. Accountability in Recommender Systems

Although *replicability* is a cornerstone of science and a fundamental requirement for scientific progress, a large amount of published research cannot be replicated. Consequently, it is not always possible to completely trust reported results and progress over state of the art. Unfortunately, even the proposing researchers in some cases are not able to reproduce their own results. Hence the scientific community is in agreement that there is an ongoing reproducibility crisis [6].

Replicability is often confused with other terms, such as *repeatability* and *reproducibility* [7]. In this paper we refer to the ACM definition[1]: *repeatability* is when the same team obtains the same results on the same experimental settings, i.e. researchers able to reliably repeat their own experiment; *reproducibility* is when a different team obtains the same results by using the same experimental settings, i.e. different researchers able to obtain the same result using the authors' own artifacts; *replicability* is when a different team obtains the same results using different experimental settings, i.e. different researchers able to obtain the same result using their own artifacts.

On August 25, 2022 the Executive Office of the US President issued a memorandum for the heads of executive departments and agencies which recommends that federal agencies: i) update their public access policies as soon as possible to make publications and their supporting data resulting from federally funded research publicly accessible without an embargo on their free and public release; ii) establish transparent procedures that ensure scientific and research integrity is maintained in public access policies; iii) coordinate with Office of Science and Technology Policy to ensure equitable delivery of federally funded research results and data.

Replicability issues concern many research areas [8], such as artificial intelligence [9], machine learning [10] and also recommender systems. In [11], an analysis of reproducibility of RS, published at prestigious scientific conferences between 2015 and 2018, revealed that most of the top-n recommendation strategies based on complex neural methods are outperformed even by very simple baselines, e.g. nearest-neighbor heuristic, making most of the claimed improvements simply not real. Indeed, many algorithms simply compare their performance with respect to weak baselines, which have not been properly optimized [12]. Problems concerning replicability were already identified in 2011 as one of the main reasons for the slowdown of recommender systems research [13], and also in 2013 Konstan and Adomavicius considered them as one of the critical issues to achieve progress [14].

In [15], the authors claim that ensuring some levels of replicability in the recommender systems research allows getting *accountable* recommender systems.

[1] `https://www.acm.org/publications/policies/artifact-review-and-badging-current`

To this purpose, a set of requirements to lead to more replicable experimental evaluations has been proposed, i.e., the definition of proper stages in the evaluation. Those stages need to be precisely documented and include *dataset collection and splitting*, *recommendation*, *candidate item filtering*, *evaluation* and *statistical testing*. All these aspects, when designing and implementing a recommender system, may affect its final results and hinder replicability when not explicitly specified. In [16], the authors provide guidelines to support the replicability of experiments involving recommender systems. Referring back to findings outlined in [15], in the following we describe some of the requirements that any evaluation should meet to enable replicability:

- *Dataset collection*: documentation of the way data have been collected and on specific pre-filtering or other modifications performed.
- *Data splitting*: documentation of the data partitioning strategies for obtaining training and test set, considering strategies such as the temporal splitting, or those not taking into account time, e.g. cross validation.
- *Recommendation*: documentation of the recommendation algorithm implementation, along with all the aspects that are not standardized, and all the different parameter settings. There are multiple factors to report to have reproducibility. For collaborative filtering based on kNN, for example, it should be reported whether the method is a user-based or item-based kNN, the number of neighbors, the formula for computing predictions, the similarity metric to find neighbors, and also the tie-breaking strategy when producing a ranking. Similarly, for content-based recommender systems, it is necessary to report all the aspects concerning the representation of items and users.
- *Candidate item filtering*: documentation on the recommended items that will be used to measure the performance of the recommender, i.e. the set of target items the recommender shall rank [17]. There are cheap methodologies considering a minimum set of recommendations not including unrated items, e.g. TestRatings, which usually tend to overestimate the performance, and other methodologies involving a high number of unrated items, e.g. TrainingItems, which better simulate a real system where no test is available [17].
- *Evaluation*: documentation on the performance measurement performed using evaluation metrics, such as error-based (MAE and RMSE), ranking-based ones (precision, recall, nDCG, and MRR), or recently introduced metrics to evaluate fairness. Multiple factors may influence the computation of a specific metric, such as normalizations, the way values are averaged, the cutoff for ranking metrics, or the discounting function adopted. The candidate item filtering may also impact the final value of the metrics.
- *Statistical testing*: documentation on data on which statistical method was computed, along with information such as the test used, the p-value

threshold, any corrections for multiple comparisons, and the confidence interval. As argued in [18], statistical inference is a key component of the evaluation process that has not been given sufficient attention.

Unfortunately, even more variables must be taken into account for implementing replicable content-based or knowledge-aware recommender systems. Indeed, besides the complex recommendation pipeline common to all recommendation strategies, the replicability for content-based and knowledge-aware recommender systems can be ensured by the precise recreation of the representation of items and/or users, which feeds the recommendation algorithm. The problem of processing textual content to obtain a set of features that describe the items (or users) is not trivial, since many techniques to elaborate content-based features exist. A classical Natural Language Processing pipeline identifies, extracts and weights relevant words and phrases from the text (lexical analysis). Moreover, it can also infer some information about the structure of the text in order to identify the role of each word in the whole content (syntactic analysis) [19]. Even simple operations in the pipeline, such as tokenization, removal of stopwords, lemmatization, named entity recognition, just to mention a few, can be performed in a different way and by different libraries. Different weighting strategies for features (e.g. TF-IDF) may return different representations and may affect final results of the evaluation. Similarly, to incorporate semantics into textual content, different strategies exist. Some of them leverage external information sources, namely *exogenous* techniques, others rely on the analysis of large corpora of textual content, i.e. *endogenous* techniques [19]. Several libraries exist for encoding endogenous and exogenous semantics, hence, it becomes more and more important to clearly report all the stages for representing content and all the parameter settings in order to come up with replicable and hence accountable recommender systems.

2.2. Knowledge-aware Content Representations

Semantic representation mechanisms allow to shift *keyword-based* representation of items and user profiles towards *concept-based* ones, since they have the potential to deal with the ambiguity of natural language and to provide a deeper comprehension of the information conveyed by the textual content.

The availability of several open knowledge sources and knowledge graphs (KG), such as WordNet [20], BabelNet [21], Freebase, WikiData [22], to cite a few, together with the recent spread of word embedding techniques, such as Word2Vec [23], GloVe [24], BERT [25], GPT-3 [26] and so on, have fueled recent progress in the field of content-based recommender systems.

The literature on knowledge-aware representations has been deeply described in [4, 19, 5], where techniques have been classified as *exogenous*, when relying on the integration of external knowledge sources, and *endogenous*, when relying on the implicit semantics learned from the analysis of large corpora of textual content to infer the usage of a word. The following subsections sketch the techniques, and present some recent trends to represent content.

2.2.1. Encoding endogenous semantics

Approaches for *endogenous semantics* representation exploit textual content to produce a vector space representation of the items to be recommended as well as of the users. These approaches fall in the general class of Distributional Semantics Models (DSMs) [27], which rely on the *distributional hypothesis*, which states that *"Words that occur in the same contexts (i.e. they typically co-occur with the same other terms) tend to have similar meanings.* These vector space representations are called *embeddings*. The context is a fragment of text in which a word appears, and may have different granularity. The classical Vector Space Model is an implementation of a DSM, when the whole document is used as *context*, but finer-grained options are possible, e.g. a paragraph, a sentence, a window of surrounding words or even a single word. The finer the granularity, the higher the dimensionality of vectors. For this reason, *word embedding* techniques usually project the original vector space into a smaller *but substantially equivalent* one, thus returning more compact representations.

Word embedding techniques which are included in the `ClayRS` framework cover all the different research *waves* [28] in the area of word embeddings.

In particular, methods available in the framework include early approaches such as Latent Semantic Analysis [29], Random Indexing [30], Latent Dirichlet Allocation and, first-generation word embedding techniques, such as Word2Vec [23], GloVe, and FastText, methods for sentence and documents representation such as Doc2Vec [31] and also more recent methods for *contextual word representations* (CWR), such as BERT [25].

The intuition behind Latent Semantic Analysis is to apply Singular Value Decomposition (SVD) to reduce the overall dimensionality of the input matrix, typically a term-term matrix, to discover *latent factors* that represent the underlying meaning of what is contained in the documents [32]. SVD allows to collapse a usually large matrix of term and document vectors into a smaller-rank approximation, in which highly correlated and co-occurring terms are captured in a single factor.

Similarly to Latent Semantic Analysis, Random Indexing represents terms and documents as points in a *semantic* vector space built according to the distributional hypothesis, but instead of using SVD for dimensionality reduction, Random Indexing adopts Random Projection [33, 34], which does not need to factorize the original matrix, but relies on an incremental and effective method performing the same process with less computational cost.

Latent Dirichlet Allocation is a generative statistical model used to discover the topics that are present in a corpus. It takes as input a term-document matrix containing the count of words in the corpus, and produces two smaller matrices, one document to topic matrix and a word to topic matrix that, when multiplied together, reproduce the original matrix with the lowest approximation error.

Word2Vec exploits *neural networks* to learn a vector space representation of words. In a nutshell, Word2Vec learns (small) word embeddings by exploiting a *two-layer neural network* which is fed with examples gathered from a corpus of textual data to learn the contexts and the linguistic usage of words to generate

the embeddings. Given a corpus of textual data, we define an input layer of size $|V|$, that corresponds to the dimension of the vocabulary V of the terms. An output layer N is created, where N is the size of the embedding we want to obtain at the end of the learning process (N is a parameter of the model and has to be properly tuned). The edges connecting the nodes in the network have different weights, initially randomly set and updated through the training process. The final representation of a term is the set of weights that connects its corresponding node in the input layer to all the nodes in the output layer.

Similarl to Word2Vec, GloVe - Global Vectors for Word Representation - is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space [35].

FastText builds on Word2Vec by learning vector representations for each word and the n-grams found within each word. The values of the representations are then averaged into one vector at each training step. While this adds a lot of additional computation to training, it enables word embeddings to encode sub-word information. FastText vectors have been shown to be more accurate than Word2Vec vectors by a number of different measures.

Doc2Vec is a neural approach that shares the same principles of Word2Vec and focuses on the representation of *documents*. In particular, it aims to generate a single embedding vector for the entire document. For this purpose, as well as Word2Vec, it receives as input the words to be worked on, but in addition a vector providing information regarding the paragraph id of each word is also provided. As well as the Word2Vec model, Doc2Vec can be based on two neural architectures: Distributed Memory Version of Paragraph Vector (PV-DM) and Distributed Bag of Words Version of Paragraph Vector (PV-DBOW).

Finally, the current state of the art in the area is represented by techniques for CWR. As previously stated, the distinctive trait of CWR techniques is the generation of a *context-aware* representation of words, which depends on the other terms that co-occur with the target word *in a particular sentence*. In other terms, the same word, encoded in different sentences (and surrounded by different words), has a different representation.

Generally speaking, these methods put their roots in the area of *sequence modeling*, since each sentence (and its resulting representation) is seen as a sequence of words. Accordingly, early methods such as ElMo [36] embeddings largely use Recurrent Neural Networks and related architecture. However, all these recent techniques gained a lot of interest after the introduction of *attention mechanisms* [37], that allow to generate a more precise representation of word and sentences since they exploit *attention* to carry on much more information about the dependencies of the target word with the other words in the sentence. The current state-of-the-art in the area of CWR is represented by methods based on Tranformers architecture. Indeed, BERT and its derived models are

currently lead several benchmark in the area of Natural Language Processing[2] [38].

2.2.2. Encoding exogenous semantics

Approaches for *exogenous semantics* representations introduce a different vision of the concept of semantics, that is obtained by exploiting data encoded in structured and external knowledge sources, which can be built by experts or constructed collaboratively. Example of such knowledge sources are *WordNet*[3] [20], a lexical database for English, *BabelNet*[4] [21], a large-scale multilingual encyclopedic dictionary and semantic network integrating heterogeneous resources, *Wikidata* [22], a free, collaborative and multilingual database built with the goal of turning Wikipedia into a fully structured resource, and the *The Linked Open Data (LOD) cloud* [39], which refers to the huge number of datasets released through the Linked Open Data initiative, whose *nucleus* is represented by `DBpedia` [40]. Two different strategies can be adopted to build a semantics-aware representation of items by exploiting the data available in the knowledge sources:

1. *linking item features to concepts*, e.g. through Word Sense Disambiguation (WSD) [41], which tries to correctly identify which of the senses of an ambiguous word is invoked in a particular use of the word itself, and Entity Linking (EL) [42], which tries to associate the *mention* of an entity in a text to an entity of the real world stored in a knowledge base
2. *linking items to a knowledge graph*, by directly linking items to nodes in a knowledge graph rather than mapping word forms to word meanings or entities. This process avoids the burden of processing textual content and provides the items with new and descriptive characteristics extracted from a knowledge base, e.g. the properties extracted from `DBpedia`.

`ClayRS` makes available an entity linking algorithm based on the BabelNet 3.0 [43] knowledge base, whose advantage is the unified approach of the two tasks of entity linking and WSD in any of the languages covered by the *native* multilingual semantic network.

As regards the methods to link items to concepts in a knowledge graph, the huge amount of data freely available on the Web thanks to the Linked Open Data initiative allows to improve the effectiveness of existing algorithms in different ways. When an item is linked to the LOD cloud, *descriptive* features of the items (and, in turn, of the profiles of the users) can be collected and exploited, even when no textual content that describes the item is available. Moreover, a more complex data model can be constructed, and this can in turn lead to a more precise representation of the interests and more interesting (and maybe surprising) recommendations. `ClayRS` provides a set of functionalities to deal

[2] `https://gluebenchmark.com/leaderboard`
[3] `http://wordnet.princeton.edu`
[4] `http://babelnet.org`

with complex networks represented in form of *graphs*, in order to fully exploit the potential of this form of representation.

2.3. Related Frameworks

As reported in [15], the use of a recommendation library may help to improve the transparency of the research work, but it could be not enough to provide full accountability, since the library might not be used properly or might not provide support for all the stages to ensure replicability.

Reporting a complete overview of the frameworks in recommender systems research and their support for the requirements to facilitate replicability is out of the scope of the current publication, but we try to give an overall picture of the current state of the art. A clear and updated overview is reported in [15], which describes the capabilities of the most frequently cited libraries in the recommender systems community, such as *Apache Mahout*, *CaseRec* [44], *LensKit* [13, 45], *LibRec* [46], *MyMediaLite* [47], *RankSys* [48], *Surprise* [49], *DaisyRec* [50], *LibRec-auto* [51], and *Rival* [52]. In our analysis we include other recently introduced frameworks, such as *Elliot* [53], *Cornac* [54], *RecBole* [55], *OpenRec* [56], *LightFM* [57], and also our `ClayRS` framework that will be described in detail in the next section.

None of the above mentioned frameworks satisfies all the requirements to facilitate replicability. Most of them supports strategies for data splitting (all but *RankSys* and *OpenRec*), while just a few of them support candidate item filtering (*Lenskit*, *RankSys*, *RiVal*, *RecBole* and `ClayRS`). Error (err) and ranking (rank) metrics are provided by most of the frameworks, with the exception of *RankSys*, *DaisyRec*, *OpenRec* and *LightFM* that only compute ranking metrics, and *Surprise* which only provides error metrics. Fairness metrics (fair) are only supported by *Elliot* and `ClayRS`. Statistical testing is supported only by *Elliot*, *CaseRec*, *LensKit* and `ClayRS`.

In the following we give a detailed description of the recently introduced frameworks, not discussed in [15].

A comprehensive and rigorous framework for reproducible recommender systems evaluation, called *Elliot*, has been introduced in [53]. *Elliot* has been compared with most of the above mentioned frameworks and with others recently introduced, such as *Cornac* [54] and *RecBole* [55]. The comparison has been performed along different dimensions, such as pre-filtering strategies (filter by rating and k-core), splitting strategies (temporal, random and fix), hyperparameter tuning (grid search, simulated annealing, Bayesian Optimization, and Random Search strategies), implemented recommendation models, evaluation metrics (accuracy, error, coverage, novelty, diversity, bias and fairness) and statistical tests (paired t-test and Wilcoxon). *Elliot* is competitive with most of the frameworks for almost all the dimensions, and significantly for pre-filtering strategies and families of metrics supported. It also implements a wide variety of popular recommendation models and two statistical hypothesis tests.

To sum up, it seems there is an ever-growing attention of the recommender systems community to the topic of replicability. Novel recommendation frameworks are continuously developed for rapid prototyping and testing of traditional

recommendation models and new ones based on deep learning, for measuring new performance dimensions such as bias and fairness, and for supporting complex hyperparameter tuning and optimization.

Unfortunately, most of the frameworks focus on the collaborative filtering paradigm, and they do not provide full support for the development of content-based and knowledge-aware recommender systems [5, 19]. Actually, some frameworks provide some basic or advanced functionalities to process text or other types of side information.

Elliot [53] allows to include side information by adopting specific implementations of a `loading` module, able to handle additional data such as visual features [58], and semantic features extracted from knowledge graphs. If needed, it is possible to code a custom loader.

RecBole [55] uses different file types for including different side information in the recommendation models. It adopts a specific file for representing user interactions in terms of ratings, review text, and timestamp; a user feature file to include user side information, such as gender, and occupation; an item feature file to include item characteristics, such as title, release year and genres adopted by MovieLens; a knowledge graph file containing a set of ⟨head, tail, relation⟩ triplets, and a file reporting the correspondence between items and the knowledge graph entities (item-to-entity mapping). Using additional files, *RecBole* also allows to load features from other sources, e.g. pre-trained entity embeddings.

The distinctive feature of *Cornac* [54] is to provide built-in functionalities to process different kinds of raw data to obtain side information, and not only to include them in the recommendation process. *Cornac* can process both image data, and item textual descriptions to obtain basic representations, such as sequences or bag-of-words, or advanced representations, such as graphs.

Similarly to *Cornac*, *OpenRec* [56] is a Python framework able to analyze multi-modal data from users and items, benefiting from advancements in other fields, such as natural language processing and computer vision.

LightFM [57] is a Python implementation of popular recommendation algorithms for implicit and explicit feedback, which also allows to incorporate both item and user metadata. The model learns embeddings for users and items in a way that encodes user preferences over items. The user and item representations are expressed in terms of representations of their features: an embedding is estimated for every feature, and these features are then summed together to come up with representations for users and items.

All the above mentioned frameworks allow to include side information in the recommendation process, and in some cases make available techniques to learn embeddings from the user or item metadata, but there is not a comprehensive set of methods for obtaining complex representations, such as those based on exogenous and endogenous representation techniques [5]. To this purpose, we have developed `ClayRS`, an exhaustive framework to address the need of having a wide variety of knowledge-aware representations, and also able to reproduce the experimental pipeline starting from the content representation, and including the recommendation and the performance evaluation tasks.

Table 1 summarizes the compliance of the different recommendation frameworks to the requirements for facilitating the replicability (see Section 2.1). We have not reported the *dataset collection* and *recommendation* requirements, since they do not concern the functionalities of a framework, but only a documentation to be reported by the researchers about the data collection and filtering, and the recommendation algorithm.

	Splitting strategies	**Candidate item filtering**	**Evaluation Metrics**	**Statistical Testing**
Mahout	•		err, rank	
CaseRec	•		err, rank	•
LensKit	•	•	err, rank	•
LibRec	•		err, rank	
MyMedialite	•		err, rank	
RankSys		•	rank	
Surprise	•		err	
DaisyRec	•		rank	
LibRec-auto	•		err, rank	
RiVal	•	•	err, rank	
Elliot	•		err, rank, fair	•
Cornac	•		err, rank	
RecBole	•	•	err, rank	
OpenRec			rank	
LightFM	•		rank	
ClayRS	•	•	err, rank, fair	•

Table 1: Compliance of recommendation frameworks to the requirements for replicability.

3. ClayRS

`ClayRS` is a new end-to-end framework written in Python, which provides a complete and customizable pipeline for content representation and recommendation [5] [6]. The framework allows handling all the steps of the pipeline, i.e. loading preferences and content information, representing content using several approaches, running recommendation algorithms and evaluating their performance. The ever increasing complexity of pipelines for representing users and items, configuring the recommendation algorithms and computing the evaluation metrics hinder the replicability of the experiments. To this purpose, `ClayRS` tries to fulfill the requirements to satisfy replicability as described in [15] and summarized in Section 2.1.

[5] `https://github.com/swapUniba/ClayRS`
[6] `https://swapuniba.github.io/ClayRS/`

The general architecture of the framework is provided in Figure 1. In a nutshell, the *Content Analyzer* module is responsible for handling and managing user and item representations by implementing most of the knowledge-aware techniques described in [4, 5]. The *Recommender* module provides recommendation algorithms which are fed with ratings and the above mentioned representations. The *Evaluator* module evaluates recommendations according to different families of metrics. The modules are completely independent one from each other. For example, the evaluator can be used to evaluate the recommendations produced by external libraries, e.g. those described in Section 2.3, and the Content Analyzer can export the representations of users and items (e.g., embeddings) for feeding a neural network architecture to generate recommendations.

The following sections briefly describe each module.

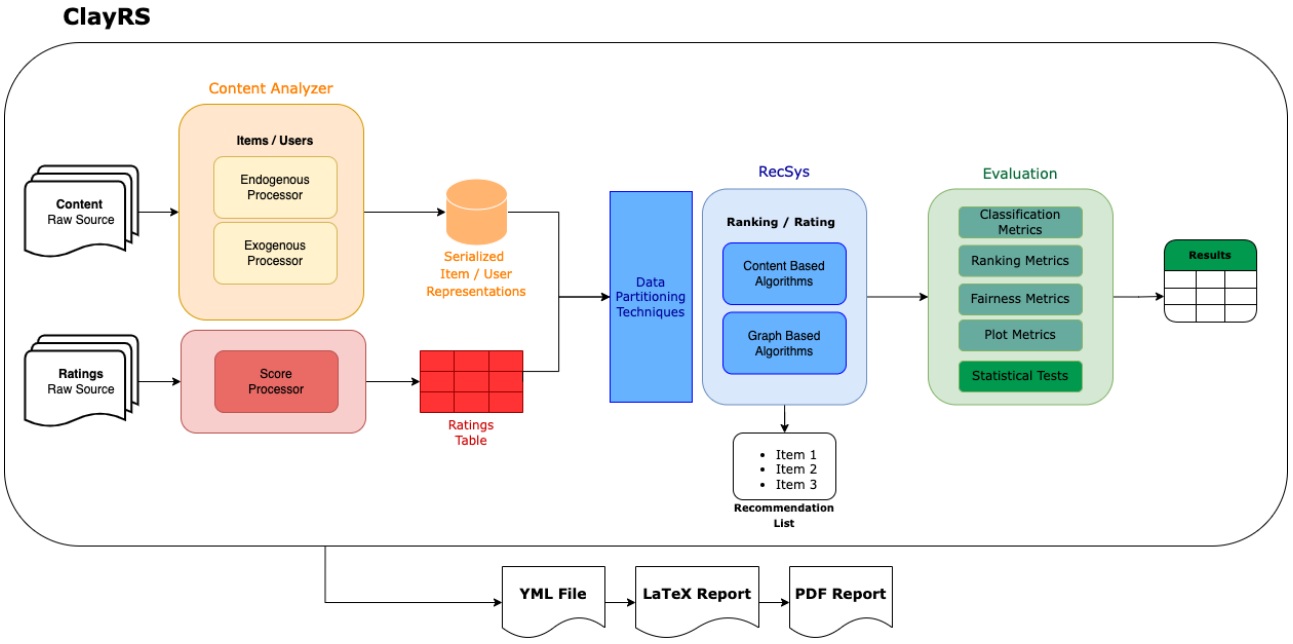

Figure 1: ClayRS Architecture.

3.1. Content Analyzer

The Content Analyzer is the most complex component of the `ClayRS` framework, which makes available several strategies for representing users and items. `ClayRS` allows to provide complex representations, where users and items are represented using different fields, and each field can be represented using one or more strategies at the same time. For example, users can be represented with demographic attributes and personality traits, while items can be represented

using properties extracted from `DBpedia` and textual reviews represented using two different strategies, e.g. TF-IDF vectors and pre-trained embeddings.

`ClayRS` is able to handle and manage the whole pipeline to load raw data (ratings and content) by heterogeneous data sources, e.g. CSV, JSON and DAT files, or SQL databases, pre-process and represent them according to different knowledge-aware techniques.

3.1.1. Pre-processing.

The *pre-processing* step aims at obtaining a set of features that describes the items, starting from the textual content. This is not trivial, since many techniques to elaborate content-based features exist and the identification of the sequence of algorithms that leads to better representation of the content often varies depending on the particular use case. When information has no structure (*e.g., text*, as the content of a news or the plot of a movie), some processing steps are needed to extract relevant information from it. This would allow to represent items such as documents, Web pages, news, product descriptions, user comments or reviews, in a form suitable to be exploited by recommendation algorithms. In other terms, the informative content conveyed by the items must be properly analyzed through particular algorithms in order to shift from the original *unstructured* representation to a *structured* one [19]. As an example, in a movie recommendation scenario, possible features that describe the items are *actors, directors, genres, plot*, which can be of different types, *i.e.* keywords, phrases, entities, or concepts extracted from a dictionary or a knowledge graph.

`ClayRS` integrates a complete pipeline for text processing, which includes the following operations: tokenization, stopwords removal, stemming, lemmatization, named entity recognition, part-of-speech tagging, and URL tagging. Pre-processing operations are provided by different state-of-the-art libraries integrated in the framework, which can be used interchangeably. The framework currently integrates the following libraries for Natural Language Processing, even though it is designed in a way that allows the integration of other libraries:

- *Natural Language Toolkit (NLTK)*[7] [59], an open-source Python library providing interfaces to over 50 corpora and lexical resources, and a suite of text processing libraries.
- *SpaCy*[8] [60], a Python library implementing more sophisticated text processing, e.g. pre-processing text for deep learning. It provides several templates in different languages, including English, Italian, Russian and Japanese, to perform tokenization, semantic tagging, and named entity recognition.
- *EkPhrasis*[9] [61], a collection of lightweight text tools, devoted to the processing of text coming from social networks. *EkPrhasis* provides methods

[7] `https://www.nltk.org/`
[8] `https://spacy.io/`
[9] `https://github.com/cbaziotis/ekphrasis`

for tokenization, word normalization, word segmentation (i.e., hashtags splitting) and spell correction, by using word statistics obtained English Wikipedia, and a large corpora of English tweets. The tokenizer can correctly manage emoticons, emojis and many unstructured expressions like dates, times and more.

3.1.2. Knowledge-aware Content Representations.

The framework currently integrates libraries for encoding exogenous and endogenous semantics, and it is designed in a way that allows the integration of other libraries.

Encoding endogenous semantics is realized through the integration of the following libraries:

- *Gensim*[10] [62], an open source Python library which provides efficient *use* or *creation* of embeddings, including popular models such as Latent Semantic Analysis (LSA) [29], Word2Vec [63], Doc2Vec [31], GloVe[24], FastText[64], Latent Dirichlet Allocation (LDA) [65] and Random Indexing [66, 30].

- *SBERT*[11] [67], a Python framework for state-of-the-art sentence, text and image embeddings. It is based on BERT-like models in order to allow the generation of embeddings where the similarity properties of semantic similar elements are preserved. These embeddings can then be compared e.g. with cosine-similarity to find sentences with a similar meaning. In order to achieve these results, the framework is based on a siamese neural network.

- *Hugging Face*[12] [68], which includes a huge number of *transformers* models. The library provides more than 59,000 models for hundreds of languages, such as BERT, RoBERTa, T5, GPT, BigBird, ELECTRA, etc. The library allows such models to be used as classifiers or as text encoding utilities for obtaining contextualized embeddings that are representative of any individual token, sentence, or document. As an example it is possible to use the last encoding layer of the model or the [CLS] token as embedding representations of a sentence.

- *Scikit-learn* [69] and *Woosh*[13] for representations based on the classical vector space model with the TF-IDF weighting scheme, with the possibility of indexing the content in order to make data searchable.

It is worth to mention that `ClayRS` allows to i) *use* already available models pre-trained on large collections of documents, or ii) *create* them from scratch

[10] `https://radimrehurek.com/gensim/`
[11] `https://www.sbert.net/`
[12] `https://huggingface.co/`
[13] `https://github.com/mchaput/whoosh`

by analyzing specific collection of documents, such as textual descriptions or reviews related to items of the catalogue adopted by the recommender system.

Encoding exogenous semantics is realized through the integration of the following libraries:

- *BabelPy*, the Python entity tagger based on Babelfy [70], a graph-based approach to entity linking, whose output is a bag of concepts and named entities in different languages supported by *BabelNet*
- *NetworkX*, the Python library for the creation and manipulation of complex networks. It is possible to create *bipartite graphs* containing two different types of nodes for representing users, items and their preferences, *tripartite graphs* which also allow to represent properties associated to items, e.g. `DBpedia` properties, and *full graphs* which allow to also enrich users with properties, e.g. personality traits, and allow the creation of new types of nodes for more complex networks. For example, a full graph may be used to create *context nodes*, to model the different contextual situations in which an item can be consumed, as described in [71]. In order to retrieve properties from a `SPARQL` endpoint, a specific wrapper has been provided.

A new version of `ClayRS` has been released, which currently offers methods to also process and include images as side information in the recommendation algorithms, by providing full support to the replicability of the pre-processing steps and the whole recommendation pipeline. More information can be found in [72].

3.2. Recommender

The *Recommender* module provides the functionalities to train a recommendation model by using imported ratings and the representations of users and items obtained by the *Content Analyzer*. The module allows to split ratings according to different strategies, i.e. fix split, bootstrap or k-fold cross validation, and also allows to specify the *candidate item filtering*, according to the following methodologies [17]:

- *TestRatings*, that computes the performance only for items rated by users in the test set
- *TestItems*, that adds to the previous items those not rated (and therefore not relevant) as well, by excluding only ratings in the training set of the active user
- *TrainingItems*, that selects all the items belonging to the training set by all users, except those rated by the target user
- *AllItems*, that selects the whole set of items, except those in the training set of the active user.

Actually the framework implements the following recommendation algorithms:

- *CentroidVector*, which computes the centroid of the items that the user liked, and then computes the similarity between that centroid and all the items to rank, by returning the most similar ones. It is possible to select the item representation, e.g. the TF-IDF representation of the plot or the Word2Vec representation for the plot and Doc2Vec for the director field, the similarity measure to adopt, e.g. cosine similarity, and the threshold to deem an item as relevant, e.g. a fixed score or the average user rating

- *IndexQuery*, which builds a query using the representation(s) of the items the user liked, and then searches the items more similar to the query, by ranking them according to the similarity score. A textual representation of items is necessary to build a significant query and perform a significant search. *Woosh* is used to index the content and perform the query, by using classical similarity or BM25

- *Classifiers*, which implements recommendation through classification algorithms. The framework provides the functionalities to train a classifier by selecting specific features to represent each item, the threshold to deem an item as relevant, and the specific classification algorithm, among those made available in the *Scikit-learn* library, e.g. Decision Trees, k-Nearest Neighbor, Random Forests, Logistic Regression, Gaussian Process, Support Vector Classifier. The framework also integrates *Regressors* to predict ratings of unseen items, instead of classifying them as relevant or not. Algorithms provided by the *Scikit-learn* library are available, such as Linear Regression, Ridge and Bayesian Ridge Regression, Stochastic Gradient Descent Regressor, and Huber Regressor to cite a few

- *PageRank*, which can be performed on graph-based representations. The PageRank can be *personalized*, which means that it will be calculated with Priors, considering the user profile as personalization vector. The framework also allows to perform feature selection using several strategies, in order to prune the graph before calculating predictions.

The framework also integrates the *distex* library [14] to offer a distributed process pool to utilize multiple CPUs or machines for enhancing the computation of recommendations.

3.3. Evaluator

This module allows to evaluate the performance of recommendations through the use of specific metrics and statistical tests. Recommendations computed by the Recommender module or computed by other libraries and imported in

[14] `https://pypi.org/project/distex/`

the framework, are provided to the Evaluator along with the ground truth. `ClayRS` provides different evaluation metrics, partitioned in different families, even though new metrics can be added:

- *Error*, containing Mean Absoulte Error (MAE), Mean Square Error (MSE) and Root Mean Square Error (RMSE)
- *Classification*, containing Precision, Recall, F-measure, R-precision
- *Ranking*, containing normalized Discounted Cumulative Gain (nDCG), Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR), and correlation coefficients, i.e. Pearson, Kendall Tau and Spearman
- *Fairness*, containing Gini index (unbalancement in terms of frequency of the distribution of the recommendations to all the users), catalogue coverage (amount of items in the catalogue which are recommended to at least one user) and ΔGAP (which shows how the popularity of the recommended items differs from the expected popularity of the items in the user profiles)
- *Plot*, reporting plots showing the long tail distribution, the profile vs recommendations popularity, and the correlation between popular or niche items and how many times are being recommended.

Metrics such as Precision, Recall, and F-measure can be computed using micro or macro-average and, similarly to nDCG, they can be computed at different cutoff values. Their values can vary according to the candidate item filtering strategy. Two statistical hypothesis tests can be computed, i.e. *Wilcoxon* and *Paired t-test*.

3.4. Ensuring Replicability with ClayRS

According to the guidelines described in [15], the stages in recommender systems evaluation should be properly documented for an accountable recommender system evaluation. All the parameters and alternatives in all the steps of the recommendation pipeline must be explicitly specified.

To this purpose `ClayRS` automatically generates a set of YAML files, one for each module of the architecture, which contain all the parameters adopted to represent users and items, to configure the recommendation algorithm and compute the metrics. Figure 2 shows the YAML file generated by the Content Analyzer, which reports the representation for the items, having two fields, i.e. description and tags, represented using the pre-trained `word2vec-google-news-300` model available in `GenSim`, and the Natural Language Processing pipeline performed on the text, using the `NLTK` library.

Figure 3 reports the YAML file generated by the Recommender System, which includes the dataset statistics, the partitioning strategy to obtain training and test set, and the parameters to configure the recommendation algorithm. In the example, a holdout partitioning strategy is used with 80% of ratings in the

training set and the rest in the test set. In order to reproduce exactly the same split, the random seed is reported. The recommendation algorithm adopted in the experiment is the Support Vector Classifier (SVC) of the `ScikitLearn` library, trained using the Word2Vec representations of description and tags. The experiment is configured to report the top-20 recommendations using the *TestRatings* candidate item filtering methodology.

```
source_file: movies_all_contents.csv
id_each_content:
- item
exogenous_representations: null
field_representations:
 description_0:
  WordEmbeddingTechnique:
   embedding_source: Gensim word2vec-google-news-300
  preprocessing:
   NLTK:
    strip_multiple_whitespace: true
    remove_punctuation: true
    stopwords_removal: true
    url_tagging: false
    lemmatization: true
    stemming: false
    pos_tag: false
    lang: english
 tags_0:
  WordEmbeddingTechnique:
   embedding_source: Gensim word2vec-google-news-300
  preprocessing:
   NLTK:
    strip_multiple_whitespace: true
    remove_punctuation: true
    stopwords_removal: true
    url_tagging: false
    lemmatization: true
    stemming: false
    pos_tag: false
    lang: english
```

Figure 2: YAML file for the Content Analyzer.

Finally, figure 4 contains all the data produced by the Evaluator. The file reports the list of metrics computed at a specific cutoff, and depending on the metrics, also some parameters, such as the threshold to deem an item as relevant, the way of computing the average, i.e. micro or macro-average, the configurations of the groups for the ΔGAP metric. The YAML file also contains the list of items in the catalogue, the computed metrics for each fold, and the overall system results.

Starting from those files, `ClayRS` also builds a report of the experiment, both in LaTeX and PDF, to help researchers to write the experimental section of their papers more rapidly. They can run the whole experimental pipeline, produce results, and have the draft of the experimental evaluation with the description of the dataset, metrics, algorithms and results to work on.

```
interactions:
 n_users: 6040
 n_items: 3706
 total_interactions: 1000209
 sparsity: 0.9553163743776871
 min_score: 1.0
 max_score: 5.0
 mean_score: 3.5815624534472295
partitioning:
 HoldOutPartitioning:
  train_set_size: 0.8
  shuffle: false
  random_state: 42
  skip_user_error: true
recsys:
 ContentBasedRS:
  algorithm:
   ClassifierRecommender:
    item_field:
     description:
     - word2vec
     tags:
     - word2vec
    classifier: SkSVC
    threshold: null
    embedding_combiner: Centroid
  mode: rank
  n_recs: '20'
  methodology:
   TestRatingsMethodology:
    only_greater_eq: null
```

Figure 3: YAML file for the Recommender System.

Such a report fulfils all the requirements for ensuring the accountability of recommender systems indicated in [15], and also helps the researchers to describe in a standard way the experimental protocol and results of their evaluation. An excerpt of the report is depicted in Figure 5.

4. ClayRS in Practice: Experimental Scenarios

In this section we illustrate how to use `ClayRS` to configure a complete recommendation pipeline for performing experiments. We performed experiments on three different datasets and using different techniques for representing content, in order to show how the framework easily supports the creation of knowledge-aware representations.

4.1. Datasets

We perform evaluation on three datasets frequently used in the recent literature, namely *MovieLens-1M (ML-1M)*, *GoodBooks* and *MovieLens-100k (ML-100k)*. Statistics are reported in Table 2.

```
n_split: 1
metrics:
 PrecisionAtK:
  k: 10
  relevant_threshold: 3
  sys_average: macro
  precision: <class 'numpy.float64'>
 RecallAtK:
  k: 10
  relevant_threshold: 3
  sys_average: macro
  precision: <class 'numpy.float64'>
 FMeasureAtK:
  k: 10
  beta: 1
  relevant_threshold: null
  sys_average: macro
  precision: <class 'numpy.float64'>
 NDCGAtK:
  k: 10
 MRRAtK:
  relevant_threshold: null
 GiniIndex:
  top_n: 10
 DeltaGap:
  user_groups:
   bb-focused: 0.2
   diverse: 0.6
   niche: 0.2
  top_n: 10
  pop_percentage: 0.2
 CatalogCoverage:
  catalog:
  - '1709'
  - '1901'
...
  - '1493'
  top_n: 10
  k: null
```

```
sys_results:
 sys - fold1:
  Precision@10 - macro: 0.9113232335187597
  Recall@10 - macro: 0.5828403461728942
  F1@10 - macro: 0.5525344967620525
  NDCG@10: 0.8885825917094407
  MRR@10: 0.8125599607840045
  Gini - Top 10: 0.795620471423874
  DeltaGap - Top 10 | bb-focused: 0.040754321323404966
  DeltaGap - Top 10 | diverse: 0.1939485859424278
  DeltaGap - Top 10 | niche: 0.32708137940578486
  CatalogCoverage - Top 10: 41.93
 sys - mean:
  Precision@10 - macro: 0.9113232335187597
  Recall@10 - macro: 0.5828403461728942
  F1@10 - macro: 0.5525344967620525
  NDCG@10: 0.8885825917094407
  MRR@10: 0.8125599607840045
  Gini - Top 10: 0.795620471423874
  DeltaGap - Top 10 | bb-focused: 0.040754321323404966
  DeltaGap - Top 10 | diverse: 0.1939485859424278
  DeltaGap - Top 10 | niche: 0.32708137940578486
  CatalogCoverage - Top 10: 41.93
```

Figure 4: YAML file for the Evaluator.

	Users	**Items**	**Ratings**	**Sparsity**	**Avg. rating**
ML-1M	6,040	3,706	1,000.209	95.53%	3.58
GoodBooks	53,424	10,000	5,976,479	98.88%	3.92
ML-100k	943	1,682	100,000	93.69%	3.52

Table 2: Statistics of the datasets

`DBpedia` was used as Knowledge Graph (KG) to map items of ML-100k and perform recommendations using the Personalized PageRank algorithm. We populate the KG using users and items along with `DBpedia` properties for items and gender and occupation extracted from the dataset for users. In total the KG contains $11,343$ nodes, 120,469 edges (links) and 10.62 average links per node.

4.2. Data Splitting and Evaluation Protocol

Datasets are split using the *HoldOut* methodology, holding 80% of ratings for training and 20% for testing, and we considered as *positive* only the ratings greater than 3. The predictive accuracy of the algorithms was evaluated on top-10 and top-20 recommendation lists, calculated by following the *TestRatings* candidate item filtering strategy [17].

1.4 Recommendation Algorithm

In this experiment a Graph Based Recommender Algorithm has been used. Specifically the NXPageRank algorithm, a Page Rank algorithm based on the networkx implementation The PageRank can be 'personalized', in this case the PageRank will be calculated with Priors. The alpha value used was: 0.85

1.5 Metrics

The Precision metric is calculated as such for the **single user**:

$$Precision_u = \frac{tp_u}{tp_u + fp_u}$$

Where:
- tp_u is the number of items which are in the recommendation list of the user and have a rating $\geq$ **3.52986**
- fp_u is the number of items which are in the recommendation list of the user and have a rating $<$ **3.52986**

Precision needs those parameters:
the **relevant_threshold**, is a parameter needed to discern relevant items and non-relevant items for every user. If not specified, the mean rating score of every user will be used, in this experiment it has been set to **None**.

Overall Precision is computed using **macro**.

Precision at k is the proportion of recommended items in the top-k set that are relevant. The Precision@K metric is calculated as such for the **single user**:

$$Precision@K_u = \frac{tp@K_u}{tp@K_u + fp@K_u}$$

Where:
- $tp@K_u$ is the number of items which are in the recommendation list of the user **cutoff to the first K items** and have a rating $\geq$ **3.52986**
- $tp@K_u$ is the number of items which are in the recommendation list of the user **cutoff to the first K items** and have a rating $<$ **3.52986**

In this experiment the value **k is 2**, the sys_average is **macro**

The FMeasure@K metric combines Precision@K and Recall@K into a single metric. It is calculated as such for the **single user**:

$$FMeasure_u = (1 + \beta^2) \cdot \frac{P@K_u \cdot R@K_u}{(\beta^2 \cdot P@K_u) + R@K_u}$$

Where:
- $P@K_u$ is the Precision at K calculated for the user **u** - $R@K_u$ is the Recall at K calculated for the user **u** - β is a real factor which could weight differently Recall or Precision based on its value:
- $\beta = 1$: Equally weight Precision and Recall - $\beta > 1$: Weight Recall more - $\beta < 1$: Weight Precision more

If $\beta = 1$, which basically is the harmonic mean of recall and precision:

$$F1@K_u = \frac{2 \cdot P@K_u \cdot R@K_u}{P@K_u + R@K_u}$$

In this experiment $\mathbf{k = 1}$, $\boldsymbol{\beta = 1}$ and **sys_average** is **macro**.

1.6 Results

In the following table, we present the results of the evaluation 1.6

Metric	Value
Precision - macro	0.55697
Precision@2 - macro	0.6527
NDCG	0.93529
MRR	0.79602
F1@1 - macro	0.04221

Table 2: Table of the results

Figure 5: PDF report generated by ClayRS.

For *ML-1M* and *GoodBooks* datasets we used *Tags* (T) and *Description* (D) fields, and their combination (T+D), using the following representation techniques:

1. **TF-IDF:** we used the representation provided by *Scikit-learn*, where the specific configuration for the sake of replicability is contained in the YAML file produced by the Content Analyzer. The file gives the details about the use of IDF, the use of a classical term frequency instead of the sub-linear scaling, the pre-processing pipeline consisting of punctuation and stopwords removal and lemmatization, performed using the *NLTK* library
2. **Word2Vec:** we used the pre-trained model `word2vec-google-news-300` from GenSim, with the same pre-processing pipeline described for TF-IDF
3. **LSA:** we used the *Latent Semantic Analysis* algorithm from *GenSim* with the same processing pipeline as before
4. **GloVe:** we used the pre-trained model `glove-twitter-50` from *GenSim* with the same processing pipeline as before

5. **FastText:** we used the pre-trained model `fasttext-wiki-news-subwords-300` from *GenSim* with the same processing pipeline as before
6. **Doc2Vec:** we used the *Doc2Vec* algorithm from *GenSim* with the same processing pipeline as before and we used a combiner based on the centroid.

4.3. Recommendation Algorithms

As recommendation algorithms we exploited:

1. *CentroidVector*, which computes the centroid of the items liked by the user, i.e. user profile, and returns the most similar items to the user profile using the cosine similarity computed on the specific tested representations
2. *Support Vector Classifier* available in *Scikit-learn*, trained with the specific tested representations
3. *Personalized PageRank*, configured using a damping factor equal to 0.85, a maximum number of iterations equal to 100 for the power method eigenvalue solver, and $1.0e-06$ as error tolerance used to check convergence in power method solver.

4.4. Evaluation Metrics and Statistical Testing

Metrics were calculated on the top-10 and top-20 recommendation lists returned by each algorithm for each user, and finally averaged over all the users (macro-average). As evaluation metrics, we adopted standard methods used to evaluate the accuracy and fairness of the algorithms. In particular, we adopted:

1. *F1* - harmonic mean of precision and recall, *nDCG* - normalized discounted cumulated gain and *MRR* - mean reciprocal rank, to assess the quality of the ranking
2. *Gini Index* to measure how unbalanced (in terms of frequency) is the distribution of the recommendations to all the users. This metric assumes values in the range [0,1], where 0 indicates a balanced (and more *fair*) distribution of the recommendations, while 1 represents the worst value (not balanced recommendations), i.e. recommendations concentrated on a single item
3. *Catalogue Coverage* measures the amount of items in the catalogue which are recommended to at least one user, and it is obtained by merging all the recommendation lists produced for all the users by an algorithm and by counting the amount of different items contained in the merged list. Of course, the higher the coverage, the higher the *fairness of the algorithm.*, since a larger number of the items available in the catalogue are included in the recommendation lists
4. The Group Average Popularity (GAP) measures the average popularity of the items in a certain group. In our case, we define $GAP(g)_p$, which measures the average popularity of the items in the user profiles p of a specific group g and $GAP(g)_r$, which measures the average popularity of the items in the recommendation list r of a specific group g. Popularity is

calculated as the amount of ratings expressed by the users on a particular item. Based on the protocol presented in [73], three different groups of users are defined: *blockbuster* (whether they majority of the items liked by the user are in the top-20% most rated items), *niche* users (majority of liked items in the less-20% most rated items) and *diverse* users (the remaining).
For each algorithm and user group, we are interested in the change in GAP (*i.e.*, ΔGAP), which shows how the popularity of the recommended items differs from the expected popularity of the items in the user profiles. Formally:

$$\Delta GAP(g) = \frac{GAP(g)_r - GAP(g)_p}{GAP(g)_p} \tag{1}$$

The interpretation of such metric is straightforward. $\Delta GAP = 0$ would indicate fair recommendations in terms of item popularity, where fair means that the average popularity of the recommendations a user receives matches the average popularity in the user's profile. Conversely, if ΔGAP is higher than 0, the algorithm *overestimates* the popularity *required* by the user, based on her previous likes. Conversely, if ΔGAP is lower than 0 an underestimation occurs.

4.5. Replicability of the experiments

For the sake of replicability the reader can access the notebooks of three experiments at the following URLs:

1. Experiment 1 on MovieLens-1M dataset:
   `https://colab.research.google.com/drive/1j_wNb947jmPW1EK7tbsrkoRWA3CdQ5oL?usp=sharing`
2. Experiment 2 on GoodBooks dataset:
   `https://colab.research.google.com/drive/1_HpRg_CURN-2_92ROxPSjZtQzI0baZ0l?usp=sharing`
3. Experiment 3 on MovieLens-100k dataset:
   `https://colab.research.google.com/drive/1idIxvhjs_LONryt90BGgSNxaeUA9CgY5?usp=sharing`

For each experiment, the notebook allows to access:

- the dataset used in terms of ratings and side information
- the knowledge-aware representations computed by the Content Analyzer, along with all the pre-processing steps performed on the content
- the complete configuration of the recommendation algorithms
- the evaluation of the models in terms of specific evaluation metrics at different cutoff values

- the comparison of the metrics computed by `ClayRS` with those computed by `RecMetrics`[15], a Python library of evalulation metrics and diagnostic tools for recommender systems, in order to identify possible discrepancies between the metric values
- the statistical tests performed using `ClayRS` to compare metric results for each user, and the same tests performed using the `SciPy` library, in order to identify possible discrepancies
- the YAML files produced by the Content Analyzer, the Recommender and the Evaluator modules, containing all the necessary parameters to replicate the experiments.

4.6. Results

The results for *ML-1M* dataset for top-10 and top-20 recommendation lists are shown in Table 3. We mark the best-performing configuration for each algorithm and metric in bold font, while the second best result is underlined. The following main observations can be made:

- *Accuracy:* content-based recommendations based on *tags* have the best performance with respect to those using *descriptions* or a combination of *tags* and *descriptions*, regardless the recommendation algorithm and the representation technique adopted. This is valid for recommendation lists of 10 or 20 items, with only few exceptions for the TF-IDF representation. The value of F1 is penalized by the low recall, while high values for nDCG and MRR show the ability of the recommender to rank items. Hence, the system is able to recommend and well rank good movies. Using only *descriptions* leads to a worsening of accuracy, and this is probably due to the fact that *tags* allow to more clearly define user profiles, differently from *descriptions* which could introduce some noise in the user profiling process. As regards the representation techniques, the best results are obtained using TF-IDF representation with the centroid vector recommendation algorithm. This means that more complex semantic representations do not allow improving accuracy. Using the SVC classifier, the best results are obtained using the Latent Semantics Analysis representation strategy.
- *Fairness:* content-based recommendations based only on *descriptions* have always better performance in terms of coverage, concentration of recommendation measured through the Gini index and also in terms of ΔGAP with respect to using tags or tags combined with descriptions. This means that *descriptions* are a valuable source of information for recommending more items from the catalogue, even though less than half of the catalogue is recommended to users. The results of the Gini index highlight a concentration of recommendations and not properly an even distribution

[15] `https://github.com/statisticianinstilettos/recmetrics`

of the items. Combining *tags* with *descriptions* hurts the performance for fairness metrics with performance which becomes similar to using only *tags*. The best Gini index is always obtained by the Doc2Vec strategy.

The results for *GoodBooks* dataset are shown in Table 4, and the following observations can be made:

- *Accuracy:* similarly to the results obtained for ML-1M, content-based recommendations based on *tags* have the best performance with respect to those using *descriptions* or a combination of *tags* and *descriptions*, regardless the recommendation algorithm, the representation technique adopted, and the size of recommendation lists. It is worth to notice that the difference in performance is not so high as in the ML-1M dataset. Similarly to the ML-1M dataset, the F1 score is penalized by the low recall, while high values for nDCG and MRR show the ability of the recommender to rank books. As regards the representation techniques, the best results are obtained using TF-IDF representation regardless of the recommendation algorithm, showing the ability of a very simple representation strategy to provide accurate recommendations.

- *Fairness:* the coverage for this dataset is very high and over 90% for all the configurations and most of the times content-based recommendations based only on *descriptions* have the best performance in terms of concentration of recommendations measured through the Gini index and also in terms of ΔGAP. Similarly to the results obtained for ML-1M, the Doc2Vec strategy performs very well in terms of Gini index, with results which are better than those obtained for ML-1M.

In order to also test `ClayRS` with graph-based recommendation algorithms, we performed experiments using *ML-100k* dataset and the Personalized PageRank recommendation algorithm. Results are shown in Table 5. We can observe that the algorithm is able to recommend less than half of the items in the catalogue, with a concentration of recommendations towards popular items, as highlighted by the high Gini index and by the ΔGAP values.

Alg.	Repr.	Content	Top-10								Top-20							
			F1	nDCG	MRR	Gini	ΔGap (bb/diverse/niche)			Cov.	F1	nDCG	MRR	Gini	ΔGap (bb/diverse/niche)			Cov.
Centroid	TF-IDF	T	0.5667	**0.9045**	**0.8844**	0.8440	0.1280	0.4230	0.7695	38.32%	0.6472	**0.9508**	**0.8844**	0.8303	0.0298	0.2520	0.5641	44.09%
		D	0.5072	0.8595	0.7728	0.8038	0.0013	0.0641	0.1219	**42.69%**	0.6031	0.9266	0.7729	0.7961	0.0016	0.0691	0.1193	**47.14%**
		T+D	**0.5670**	0.9042	0.8841	0.8434	0.1277	0.4155	0.7519	38.34%	**0.6473**	0.9506	0.8841	0.8305	0.0298	0.2496	0.5557	44.23%
	Word2Vec	T	0.5612	0.9009	0.8677	0.8462	0.1289	0.4031	0.7018	38.07%	0.6449	0.9493	0.8677	0.8308	0.0312	0.2495	0.5467	43.93%
		D	0.4984	0.8570	0.7583	0.8083	-0.0142	0.0387	0.1436	41.23%	0.5987	0.9255	0.7583	0.7976	-0.0006	0.0614	0.1312	46.17%
		T+D	0.5506	0.8935	0.8442	0.8400	0.0959	0.2981	0.5103	38.05%	0.6368	0.9454	0.8442	0.8278	0.0269	0.2082	0.4354	43.82%
	LSA	T	0.5536	0.8937	0.8616	0.8169	0.0944	0.2725	0.3610	40.85%	0.6391	0.9453	0.8616	0.8117	0.0245	0.1887	0.3299	45.60%
		D	0.5054	0.8601	0.7615	0.7991	-0.0145	0.0615	0.1541	42.15%	0.6038	0.9270	0.7616	0.7943	-0.0025	0.0651	0.1460	46.71%
		T+D	0.5468	0.8885	0.8388	0.8084	0.0760	0.2101	0.2486	41.12%	0.6344	0.9425	0.8388	0.8067	0.0220	0.1626	0.2501	45.76%
	GloVE	T	0.5544	0.8954	0.8635	0.8497	0.1279	0.4320	0.7979	38.37%	0.6373	0.9461	0.8635	0.8329	0.0285	0.2529	0.5764	43.93%
		D	0.5037	0.8583	0.7562	0.8113	0.0092	0.0953	0.1956	41.31%	0.6009	0.9259	0.7562	0.8006	-0.0008	0.0886	0.1745	46.44%
		T+D	0.5384	0.8850	0.8351	0.8403	0.0724	0.3249	0.6527	38.02%	0.6271	0.9402	0.8351	0.8264	0.0143	0.1987	0.4802	43.93%
	FastText	T	0.5598	0.9000	0.8693	0.8518	0.1351	0.4440	0.8175	37.64%	0.6433	0.9487	0.8693	0.8352	0.0318	0.2642	0.6007	43.63%
		D	0.4953	0.8537	0.7501	0.8091	-0.0378	0.0196	0.1486	41.15%	0.5962	0.9231	0.7502	0.7989	-0.0112	0.0431	0.1349	46.14%
		T+D	0.5408	0.8877	0.8376	0.8384	0.0480	0.2515	0.5058	37.91%	0.6298	0.9415	0.8376	0.8266	0.0148	0.1743	0.4132	43.85%
	Doc2Vec	T	0.5637	0.9021	0.8766	0.8268	0.0964	0.3361	0.6152	39.58%	0.6456	0.9499	0.8766	0.8167	0.0232	0.2030	0.4416	45.06%
		D	0.4956	0.8525	0.7362	**0.7989**	-0.0167	0.0125	0.0450	42.12%	0.5969	0.9231	0.7363	**0.7922**	-0.0001	0.0508	0.0873	46.52%
		T+D	0.5622	0.9005	0.8723	0.8273	0.0936	0.3256	0.5876	39.50%	0.6450	0.9492	0.8723	0.8171	0.0223	0.1989	0.4287	45.06%
SVC	TF-IDF	T	0.5681	0.8985	0.8449	0.8116	0.0651	0.2782	0.5006	41.31%	0.6547	0.9442	0.8449	0.8089	0.0189	0.1839	0.3936	45.82%
		D	0.4935	0.8518	0.7218	0.7634	-0.0010	0.0915	0.1950	44.95%	0.5986	0.9221	0.7220	0.7741	-0.0022	0.0796	0.1708	48.27%
		T+D	0.5641	0.8930	0.8161	0.8123	0.0652	0.2723	0.5044	41.47%	0.6549	0.9402	0.8162	0.8109	0.0185	0.1858	0.4016	45.74%
	Word2Vec	T	0.5605	0.8944	0.8320	0.8038	0.0572	0.2187	0.3543	41.50%	0.6509	0.9432	0.8321	0.8073	0.0196	0.1674	0.3220	45.76%
		D	0.5062	0.8585	0.7460	0.7615	0.0017	0.0829	0.1573	44.95%	0.6077	0.9255	0.7461	0.7734	0.0001	0.0803	0.1523	48.11%
		T+D	0.5525	0.8885	0.8125	0.7956	0.0407	0.1939	0.3270	41.93%	0.6464	0.9396	0.8125	0.8009	0.0147	0.1517	0.2971	45.71%
	LSA	T	**0.5731**	**0.9034**	**0.8593**	0.8204	0.0878	0.2971	0.4649	40.34%	**0.6580**	**0.9479**	**0.8593**	0.8174	0.0247	0.2024	0.3980	45.28%
		D	0.5016	0.8558	0.7398	0.7625	0.0038	0.0809	0.1677	44.98%	0.6029	0.9242	0.7398	0.7736	0.0013	0.0768	0.1605	48.03%
		T+D	0.5704	0.9003	0.8458	0.8179	0.0901	0.2932	0.4688	40.15%	0.6566	0.9457	0.8458	0.8168	0.0249	0.2019	0.3940	45.25%
	GloVe	T	0.5469	0.8849	0.7918	0.7899	0.0330	0.1633	0.2569	42.63%	0.6429	0.9381	0.7919	0.7980	0.0152	0.1408	0.2598	46.01%
		D	0.4998	0.8541	0.7328	0.7623	0.0002	0.0722	0.1477	**45.63%**	0.6000	0.9233	0.7329	0.7723	2.90e-06	0.0722	0.1432	48.43%
		T+D	0.5327	0.8751	0.7710	0.7792	0.0162	0.1225	0.2113	43.66%	0.6331	0.9334	0.7710	0.7899	0.0078	0.1175	0.2195	46.71%
	FastText	T	0.5524	0.8885	0.7999	0.7912	0.0398	0.1806	0.2922	42.01%	0.6463	0.9400	0.8000	0.7998	0.0161	0.1466	0.2844	45.98%
		D	0.5031	0.8563	0.7463	0.7605	-0.0015	0.0831	0.1668	45.25%	0.6031	0.9243	0.7464	0.7732	-0.0016	0.0763	0.1523	**48.60%**
		T+D	0.5380	0.8784	0.7782	0.7804	0.0213	0.1400	0.2394	43.04%	0.6396	0.9348	0.7783	0.7911	0.0096	0.1249	0.2420	46.33%
	Doc2Vec	T	0.5596	0.8942	0.8221	0.8010	0.0461	0.2193	0.3954	42.04%	0.6500	0.9434	0.8221	0.8035	0.0154	0.1619	0.3426	45.95%
		D	0.4979	0.8542	0.7376	**0.7602**	-0.0015	0.0735	0.1475	44.98%	0.5981	0.9237	0.7377	**0.7718**	-0.0002	0.0757	0.1421	48.49%
		T+D	0.5583	0.8935	0.8170	0.7994	0.0442	0.2088	0.3829	42.26%	0.6493	0.9429	0.8170	0.8035	0.0158	0.1584	0.3309	46.22%

Table 3: Accuracy and fairness for ML-1M dataset.

Alg.	Repr.	Content	Top-10							Top-20						
			F1	nDCG	MRR	Gini	ΔGap (bb/diverse/niche)			Cov.	F1	nDCG	MRR	Gini	ΔGap (bb/diverse/niche)			Cov.
Centroid	TF-IDF	T	0.5202	0.8673	0.7843	0.7008	-0.2746	-0.1275	0.0123	96.50%	0.6729	0.9439	0.7847	0.5931	-0.4025	-0.2565	0.0035	99.98%
		D	0.5091	0.8616	0.7722	0.6479	-0.4489	-0.3031	-0.0664	93.68%	0.6701	0.9415	0.7725	0.5717	-0.4545	-0.3021	-0.0120	100%
		T+D	**0.5210**	**0.8676**	**0.7866**	0.6959	-0.2855	-0.1418	-0.0033	97.11%	**0.6734**	**0.9440**	**0.7870**	0.5912	-0.4053	-0.2588	-0.0015	99.99%
	Word2Vec	T	0.5053	0.8609	0.7587	0.6849	-0.3373	-0.1868	0.0084	96.67%	0.6674	0.9411	0.7591	0.5813	-0.4165	-0.2690	0.0003	99.99%
		D	0.4939	0.8551	0.7434	0.6717	-.0.4431	-0.2860	-0.0355	88.72%	0.6649	0.9387	0.7437	0.5808	-0.4442	-0.2909	-0.0016	100%
		T+D	0.4984	0.8573	0.7503	0.6850	-0.3781	-0.2239	-0.0062	90.18%	0.6658	0.9395	0.7506	0.5862	-0.4240	-0.2746	0.0029	99.99%
	LSA	T	0.5154	0.8655	0.7773	0.6824	-0.3327	-0.2006	-0.0741	97.60%	0.6709	0.9430	0.7777	0.5823	-0.4193	-0.2760	-0.0173	99.99%
		D	0.4975	0.8566	0.7432	0.6539	-0.4623	-0.3168	-0.0689	92.31%	0.6669	0.9393	0.7435	0.5745	-0.4511	-0.2965	-0.0070	100%
		T+D	0.5109	0.8632	0.7740	0.6870	-0.3451	-0.2088	-0.0560	94.88%	0.6700	0.9421	0.7743	0.5822	-0.4251	-0.2789	-0.0128	99.98%
	GloVE	T	0.5051	0.8604	0.7565	0.6775	-0.3692	-0.2111	-0.0055	96.06%	0.6691	0.9410	0.7568	0.5787	-0.4301	-0.2776	-0.0027	99.98%
		D	0.4954	0.8561	0.7425	0.6671	-0.4395	-0.2905	-0.0542	89.15%	0.6659	0.9391	0.7428	0.5779	-0.4466	-0.2954	-0.0085	100%
		T+D	0.5004	0.8583	0.7523	0.6768	-0.4016	-0.2484	-0.0255	90.44%	0.6678	0.9400	0.7525	0.5826	-0.4353	-0.2845	-0.0028	99.99%
	FastText	T	0.5023	0.8591	0.7565	0.6736	-0.3877	-0.2338	-0.0254	95.67%	0.6679	0.9404	0.7568	0.5779	-0.4375	-0.2846	-0.0034	99.99%
		D	0.4943	0.8554	0.7377	0.6813	-0.4274	-0.2632	0.0209	86.75%	0.6664	0.9388	0.7380	0.5785	-0.4445	-0.2883	0.0065	100%
		T+D	0.4992	0.8578	0.7504	0.6839	-0.4090	-0.2421	0.0129	88.62%	0.6674	0.9398	0.7507	0.5818	-0.4381	-0.2825	0.0074	100%
	Doc2Vec	T	0.5062	0.8618	0.7591	**0.6326**	-0.4078	-0.2822	-0.1195	**99.15%**	0.6685	0.9415	0.7595	**0.5660**	-0.4420	-0.2978	-0.0272	100%
		D	0.4995	0.8573	0.7464	0.6459	-0.4551	-0.3053	-0.0540	92.78%	0.6670	0.9396	0.7467	0.5716	-0.4565	-0.3043	-0.0121	100%
		T+D	0.5047	0.8606	0.7567	0.6388	-0.4187	-0.2903	-0.1154	96.39%	0.6685	0.9410	0.7570	0.5715	-0.4426	-0.2965	-0.0243	100%
SVC	TF-IDF	T	**0.5484**	**0.8775**	**0.8137**	0.5773	-0.4079	-0.2620	0.0004	99.99%	**0.6855**	**0.9482**	**0.8139**	0.5517	-0.4573	-0.3021	-0.0031	100%
		D	0.5028	0.8577	0.7488	0.5519	-0.4536	-0.3040	-0.0140	100%	0.6684	0.9397	0.7492	0.5529	-0.4602	-0.3022	-7.9978e-05	100%
		T+D	0.5424	0.8746	0.8029	0.5695	-0.4209	-0.2715	-0.0058	100%	0.6838	0.9469	0.8032	0.5523	-0.4582	-0.3008	-0.0022	100%
	Word2Vec	T	0.5239	0.8674	0.7679	0.5554	-0.4454	-0.2927	-0.0116	99.99%	0.6774	0.9439	0.7683	0.5535	-0.4533	-0.2986	-0.0048	100%
		D	0.5015	0.8580	0.7461	0.5503	-0.4601	-0.3059	-0.0094	100%	0.6681	0.9399	0.7464	0.5528	-0.4577	-0.3019	-0.0038	100%
		T+D	0.5173	0.8648	0.7650	0.5517	-0.4584	-0.3011	-0.0160	100%	0.6746	0.9428	0.7654	0.5529	-0.4568	-0.3003	-0.0054	100%
	LSA	T	0.5466	0.8768	0.8071	0.5741	-0.4153	-0.2569	0.0115	100%	**0.6855**	0.9479	0.8073	0.5545	-0.4519	-0.2954	-0.0028	100%
		D	0.4966	0.8559	0.7428	0.5496	-0.4607	-0.3024	-0.0107	100%	0.6661	0.9390	0.7431	0.5519	-0.4569	-0.3020	-0.0026	100%
		T+D	0.5375	0.8727	0.7978	0.5670	-0.4255	-0.2693	0.0033	99.99%	0.6820	0.9462	0.7980	0.5537	-0.4532	-0.2976	-0.0027	100%
	GloVe	T	0.5073	0.8605	0.7433	0.5498	-0.4597	-0.3050	-0.0091	100%	0.6720	0.9409	0.7437	0.5539	-0.4548	-0.2992	-0.0035	100%
		D	0.4919	0.8541	0.7299	0.5498	-0.4564	-0.3038	-0.0110	100%	0.6648	0.9382	0.7302	0.5529	-0.4563	-0.3030	-0.0056	100%
		T+D	0.5000	0.8576	0.7373	0.5492	-0.4577	-0.3039	-0.0093	100%	0.6685	0.9397	0.7377	0.5537	-0.4576	-0.3014	-0.0053	100%
	FastText	T	0.5137	0.8631	0.7530	0.5546	-0.4465	-0.2907	-0.0068	100%	0.6745	0.9420	0.7534	0.5539	-0.4540	-0.2994	-0.0037	100%
		D	0.4943	0.8549	0.7345	**0.5485**	-0.4631	-0.3049	-0.0101	100%	0.6651	0.9385	0.7348	0.5528	-0.4573	-0.3016	-0.0006	100%
		T+D	0.5065	0.8601	0.7473	0.5505	-0.4570	-0.2980	-0.0168	100%	0.6704	0.9408	0.7477	0.5532	-0.4565	-0.3008	-0.0029	100%
	Doc2Vec	T	0.5255	0.8682	0.7706	0.5598	-0.4403	-0.2908	-0.0025	100%	0.6783	0.9442	0.7709	0.5526	-0.4574	-0.3016	-0.0088	100%
		D	0.4992	0.8570	0.7400	0.5490	-0.4622	-0.3084	-0.0070	100%	0.6675	0.9395	0.7403	**0.5511**	-0.4596	-0.3050	-0.0059	100%
		T+D	0.5225	0.8668	0.7693	0.5564	-0.4458	-0.2948	-0.0149	99.98%	0.6768	0.9436	0.7696	0.5521	-0.4589	-0.3027	-0.0082	100%

Table 4: Accuracy and fairness for GoodBooks dataset.

Alg.	Top-10							Top-20						
Personalized PageRank	F1	nDCG	MRR	Gini	ΔGAP (bb/diverse/niche)		Cov.	F1	nDCG	MRR	Gini	ΔGAP (bb/diverse/niche)		Cov.
	0.5894	0.8831	0.7922	0.6948	0.0495	0.3501	0.7532	41.50%	0.6649	0.9326	0.7922	0.6577	-0.0201	0.1467	0.4465	56.54%

Table 5: Accuracy and fairness for MovieLens100k dataset.

5. Open Challenges

In the last decade, recommendation pipelines have become increasingly complex due to the use of sophisticated algorithms based on neural networks or to the ever increasing adoption of side information involving not only text, but images, audio, or video as well. This makes the replicability of experiments more and more challenging. For these reasons, the use of recommendation libraries may help to promote a fair evaluation of new algorithms and approaches with state-of-the-art baselines and allow other researchers to correctly reproduce the results presented in scientific papers.

As good practice, libraries should be designed and implemented in a way that they can provide support to all the stages for a full accountability. `ClayRS` deals with this open challenge by *mandatorily* requiring the implementation of the replicability issues for the whole recommendation pipeline. This is a distinctive aspect of `ClayRS` with respect to other frameworks.

Indeed, each new technique implemented by the *Content Analyzer* must provide all the details to ensure the replicability. This means that each preprocessing operation along with its parameters must be properly listed and documented in the YAML file: this would allow to obtain exactly the same representation since, it has been shown that, even very simple preprocessing operations have an impact on the overall performance of recommendations [72]. Similarly, each new recommendation algorithm implemented by the *Recommender* module or each new evaluation metrics implemented by the *Evaluator* requires the proper documentation to be included in the YAML files for the sake of accountability.

The second open challenge that could be tackled using `ClayRS` is the support to the creation and *hybridization* of different types of knowledge representations, which currently represent the state of the art for several recommendation scenarios [74, 75]. Differently from other recommendation frameworks, `ClayRS` provides methods for generating and combining complex representations using a variety of techniques, in order to feed internal recommendation algorithms implemented in the framework, or external systems, e.g. implementing complex neural architectures. The possibility of managing different representations inside the same system allows to have more control on the entire process and to foster the replicability through the support provided by the framework. `ClayRS` allows to represent items in a complex way using different fields, and each field may be processed in a different way and represented using different knowledge-aware techniques. Those complex representations may be used singularly or combined through specific methods, such as the concatenation, sum or other pre-fusion strategies that could be implemented. With the new version of `ClayRS` which

also supports the use of visual features extracted from images [72], more powerful hybridization techniques could be implemented inside the framework, combining both text and images, without losing the advantage of a process which results completely replicable. This is one of the most important distinctive aspects of `ClayRS` with respect to other state of the art recommendation libraries: the recommendation pipeline makes available not only the recommendation algorithms and the evaluation module, but it provides a *shared interface* to use a very wide set of knowledge-aware representations for side information, implemented in a myriad of specific libraries, without the burden of dealing with the specific implementations of each library.

To sum up, the design choices behind the development of the `ClayRS` framework might drive the development of other specific libraries, by following our inspiring principles: (i) full support to replicability for each functionality implemented by the library; (ii) shared interface towards different methods to deal with content. This would allow to foster replicability, making fairer the comparisons among results coming from different systems.

6. Conclusions and Future Work

In this paper, we have introduced `ClayRS`, an end-to-end Python framework that allows to build *replicable* recommendation pipelines based on *knowledge-aware* algorithms. The framework fills in a significant gap in the current research in the area, since it provides researchers and practitioners with an easy-to-use and extensible tool to build state-of-the-art implementations of a knowledge-aware recommender system. In this first implementation, the framework covers the most important content representations currently presented in literature and allows the construction of very sophisticated recommendation pipelines based on the combination of different and heterogeneous data representations and data sources. The validity of the framework is also demonstrated by providing a complete and replicable example of a different recommendation pipeline based on three popular state-of-the-art datasets.

As future work, we plan to extend the representation mechanisms for content. In particular we plan to integrate *knowledge graph embedding* techniques [76], whose aim is to represent *entities* and *relations* in a knowledge graph as *dense vectors* by projecting them in a vector space. The task is challenging since each technique aims to preserve the original structure of the graph (*i.e.,* nodes having similar neighborhood or nodes playing a similar role in the network have a similar representation, too) [76]. Such techniques obtained very good performance in a broad range of scenarios where data can be modeled as a graph, such as biology and social networks [77], and recommender systems are no exception [78, 79, 80].

We also plan to integrate algorithms for explaining recommendations. Indeed, recommender systems need to progressively evolve from simple black boxes to systems having the ability to properly explain or justify their own behavior. We are currently integrating ExpLOD [81], an algorithm-agnostic framework able to generate natural language explanations for recommendations provided

by a generic algorithm. ExpLOD leverages the information available in the Linked Open Data cloud, a huge set of interconnected semantic datasets which encode in RDF format the information covering many topical domains.

Acknowledgments

This research is partially funded by PNRR project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI (CUP H97G22000210007) under the NRRP MUR program funded by the NextGenerationEU.

References

[1] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, Recommender systems: an introduction, Cambridge University Press, 2010.

[2] J. Grau, Personalized product recommendations: Predicting shoppers' needs (2009).

[3] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Computing Surveys (CSUR) 52 (1) (2019) 1–38.

[4] M. de Gemmis, P. Lops, C. Musto, F. Narducci, G. Semeraro, Semantics-aware content-based recommender systems, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 119–159.

[5] C. Musto, M. de Gemmis, P. Lops, F. Narducci, G. Semeraro, Semantics and content-based recommendations, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2022, pp. 251–2987.

[6] M. Baker, 1,500 scientists lift the lid on reproducibility, Nature 533 (7604) (2016) 452–454, http://dx.doi.org/10.10138/533452a.

[7] H. E. Plesser, Reproducibility vs. replicability: A brief history of a confused terminology, Frontiers Neuroinformatics 11 (2017) 76. doi:10.3389/fninf.2017.00076.
URL https://doi.org/10.3389/fninf.2017.00076

[8] C. S. Collberg, T. A. Proebsting, Repeatability in computer systems research, Commun. ACM 59 (3) (2016) 62–69. doi:10.1145/2812803.

[9] O. E. Gundersen, S. Kjensmo, State of the art: Reproducibility in artificial intelligence, in: S. A. McIlraith, K. Q. Weinberger (Eds.), Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, AAAI Press, 2018, pp. 1644–1651.

[10] O. E. Gundersen, S. Shamsaliei, R. Isdahl, Do machine learning platforms provide out-of-the-box reproducibility?, Future Gener. Comput. Syst. 126 (2022) 34–47.
URL https://doi.org/10.1016/j.future.2021.06.014

[11] M. F. Dacrema, S. Boglio, P. Cremonesi, D. Jannach, A troubling analysis of reproducibility and progress in recommender systems research, ACM Trans. Inf. Syst. 39 (2) (2021) 20:1–20:49.

[12] S. Rendle, L. Zhang, Y. Koren, On the difficulty of evaluating baselines: A study on recommender systems, CoRR abs/1905.01395. arXiv:1905.01395.
URL http://arxiv.org/abs/1905.01395

[13] M. D. Ekstrand, M. Ludwig, J. A. Konstan, J. Riedl, Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit, in: B. Mobasher, R. D. Burke, D. Jannach, G. Adomavicius (Eds.), Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011, ACM, 2011, pp. 133–140. doi:10.1145/2043932.2043958.
URL https://doi.org/10.1145/2043932.2043958

[14] J. A. Konstan, G. Adomavicius, Toward identification and adoption of best practices in algorithmic recommender systems research, in: A. Bellogín, P. Castells, A. Said, D. Tikk (Eds.), Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, RepSys 2013, Hong Kong, China, October 12, 2013, ACM, 2013, pp. 23–28. doi:10.1145/2532508.2532513.
URL https://doi.org/10.1145/2532508.2532513

[15] A. Bellogín, A. Said, Improving accountability in recommender systems research through reproducibility, User Model. User Adapt. Interact. 31 (5) (2021) 941–977. doi:10.1007/s11257-021-09302-x.
URL https://doi.org/10.1007/s11257-021-09302-x

[16] N. Polatidis, E. Pimenidis, A. Fish, S. Kapetanakis, A guideline-based approach for assisting with the reproducibility of experiments in recommender systems evaluation, Int. J. Artif. Intell. Tools 28 (8) (2019) 1960011:1–1960011:16. doi:10.1142/S021821301960011X.
URL https://doi.org/10.1142/S021821301960011X

[17] A. Bellogín, P. Castells, I. Cantador, Precision-oriented evaluation of recommender systems: an algorithmic comparison, in: B. Mobasher, R. D. Burke, D. Jannach, G. Adomavicius (Eds.), Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011, ACM, 2011, pp. 333–336. doi:10.1145/2043932.2043996.
URL https://doi.org/10.1145/2043932.2043996

[18] N. Ihemelandu, M. D. Ekstrand, Statistical inference: The missing piece of recsys experiment reliability discourse, in: E. Zangerle, C. Bauer, A. Said (Eds.), Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop 2021 co-located with the 15th ACM Conference on Recommender Systems (RecSys 2021), Amsterdam, The Netherlands, September 25, 2021, Vol. 2955 of CEUR Workshop Proceedings, CEUR-WS.org, 2021.
URL http://ceur-ws.org/Vol-2955/paper9.pdf

[19] P. Lops, C. Musto, F. Narducci, G. Semeraro, Semantics in Adaptive and Personalised Systems - Methods, Tools and Applications, Springer, 2019. doi:10.1007/978-3-030-05618-6.
URL https://doi.org/10.1007/978-3-030-05618-6

[20] G. Miller, WordNet: An On-Line Lexical Database, International Journal of Lexicography 3 (4), (Special Issue).

[21] R. Navigli, S. P. Ponzetto, Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network, Artif. Intell. 193 (2012) 217–250. doi:10.1016/j.artint.2012.07.001.

[22] D. Vrandecic, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Commun. ACM 57 (10) (2014) 78–85. doi:10.1145/2629489.

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[24] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[25] J. D. M.-W. C. Kenton, L. K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of NAACL-HLT, 2019, pp. 4171–4186.

[26] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.

[27] Z. S. Harris, Mathematical Structures of Language, Interscience, New York,, 1968.

[28] P. Lops, C. Musto, M. Polignano, Semantics-aware content representations for reproducible recommender systems (score), in: Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization, 2022, pp. 354–356.

[29] T. K. Landauer, P. W. Foltz, D. Laham, An introduction to latent semantic analysis, Discourse processes 25 (2-3) (1998) 259–284.

[30] M. Sahlgren, The Word-Space Model: Using Distributional Analysis to Represent Syntagmatic and Paradigmatic Relations between Words in High-dimensional Vector Spaces, Ph.D. thesis, Stockholm University (2006).

[31] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International conference on machine learning, 2014, pp. 1188–1196.

[32] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman, Indexing by Latent Semantic Analysis, Journal of the American Society for Information Science 41 (6) (1990) 391–407.

[33] C. H. Papadimitriou, P. Raghavan, H. Tamaki, S. Vempala, Latent semantic indexing: A probabilistic analysis, in: PODS, ACM Press, 1998, pp. 159–168.

[34] S. S. Vempala, The Random Projection Method, Vol. 65, American Mathematical Society, 2004.

[35] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.

[36] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, CoRR abs/1802.05365. `arXiv:1802.05365`.
URL `http://arxiv.org/abs/1802.05365`

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30.

[38] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S. Bowman, Glue: A multi-task benchmark and analysis platform for natural language understanding, in: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, 2018, pp. 353–355.

[39] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann, DBpedia-a crystallization point for the Web of Data, Journal of web semantics 7 (3) (2009) 154–165.

[40] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A nucleus for a web of open data, Springer, 2007.

[41] C. D. Manning, H. Schütze, Foundations of statistical natural language processing, Vol. 999, MIT Press, 1999.

[42] D. Rao, P. McNamee, M. Dredze, Entity linking: Finding extracted entities in a knowledge base, in: Multi-source, multilingual information extraction and summarization, Springer, 2013, pp. 93–115.

[43] R. Navigli, S. P. Ponzetto, BabelRelate! a joint multilingual approach to computing semantic relatedness, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12), Toronto, Canada, 2012.

[44] A. da Costa, E. Fressato, F. Neto, M. Manzato, R. Campello, Case recommender: A flexible and extensible python framework for recommender systems, in: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18, ACM, New York, NY, USA, 2018, pp. 494–495. doi:10.1145/3240323.3241611.
URL http://doi.acm.org/10.1145/3240323.3241611

[45] M. D. Ekstrand, Lenskit for python: Next-generation software for recommender systems experiments, in: M. d'Aquin, S. Dietze, C. Hauff, E. Curry, P. Cudré-Mauroux (Eds.), CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020, ACM, 2020, pp. 2999–3006. doi:10.1145/3340531.3412778.
URL https://doi.org/10.1145/3340531.3412778

[46] G. Guo, J. Zhang, Z. Sun, N. Yorke-Smith, Librec: A java library for recommender systems, in: A. I. Cristea, J. Masthoff, A. Said, N. Tintarev (Eds.), Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation, and Personalization (UMAP 2015), Dublin, Ireland, June 29 - July 3, 2015, Vol. 1388 of CEUR Workshop Proceedings, CEUR-WS.org, 2015.
URL http://ceur-ws.org/Vol-1388/demo_paper1.pdf

[47] Z. Gantner, S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Mymedialite: a free recommender system library, in: B. Mobasher, R. D. Burke, D. Jannach, G. Adomavicius (Eds.), Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011, ACM, 2011, pp. 305–308. doi:10.1145/2043932.2043989.
URL https://doi.org/10.1145/2043932.2043989

[48] S. Vargas, Novelty and diversity enhancement and evaluation in recommender systems and information retrieval, in: S. Geva, A. Trotman, P. Bruza, C. L. A. Clarke, K. Järvelin (Eds.), The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014, ACM, 2014, p. 1281. doi:10.1145/2600428.2610382.
URL https://doi.org/10.1145/2600428.2610382

[49] N. Hug, Surprise: A python library for recommender systems, J. Open Source Softw. 5 (52) (2020) 2174. doi:10.21105/joss.02174.
URL https://doi.org/10.21105/joss.02174

[50] Z. Sun, D. Yu, H. Fang, J. Yang, X. Qu, J. Zhang, C. Geng, Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison, in: R. L. T. Santos, L. B. Marinho, E. M. Daly, L. Chen, K. Falk, N. Koenigstein, E. S. de Moura (Eds.), RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020, ACM, 2020, pp. 23–32. doi:10.1145/3383313.3412489.
URL https://doi.org/10.1145/3383313.3412489

[51] N. Sonboli, M. Mansoury, Z. Guo, S. Kadekodi, W. Liu, Z. Liu, A. Schwartz, R. Burke, librec-auto: A tool for recommender systems experimentation, in: G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, H. Tong (Eds.), CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021, ACM, 2021, pp. 4584–4593. doi:10.1145/3459637.3482006.
URL https://doi.org/10.1145/3459637.3482006

[52] A. Said, A. Bellogín, Rival: a toolkit to foster reproducibility in recommender system evaluation, in: A. Kobsa, M. X. Zhou, M. Ester, Y. Koren (Eds.), Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014, ACM, 2014, pp. 371–372. doi:10.1145/2645710.2645712.
URL https://doi.org/10.1145/2645710.2645712

[53] V. W. Anelli, A. Bellogín, A. Ferrara, D. Malitesta, F. A. Merra, C. Pomo, F. M. Donini, T. D. Noia, Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation, in: F. Diaz, C. Shah, T. Suel, P. Castells, R. Jones, T. Sakai (Eds.), SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021, ACM, 2021, pp. 2405–2414. doi:10.1145/3404835.3463245.
URL https://doi.org/10.1145/3404835.3463245

[54] A. Salah, Q. Truong, H. W. Lauw, Cornac: A comparative framework for multimodal recommender systems, J. Mach. Learn. Res. 21 (2020) 95:1–95:5.
URL http://jmlr.org/papers/v21/19-805.html

[55] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian, Y. Min, Z. Feng, X. Fan, X. Chen, P. Wang, W. Ji, Y. Li, X. Wang, J. Wen, Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms, in: G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, H. Tong (Eds.), CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event,

Queensland, Australia, November 1 - 5, 2021, ACM, 2021, pp. 4653–4664. doi:10.1145/3459637.3482016.
URL https://doi.org/10.1145/3459637.3482016

[56] L. Yang, E. Bagdasaryan, J. Gruenstein, C. Hsieh, D. Estrin, Openrec: A modular framework for extensible and adaptable recommendation algorithms, in: Y. Chang, C. Zhai, Y. Liu, Y. Maarek (Eds.), Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018, ACM, 2018, pp. 664–672. doi:10.1145/3159652.3159681.
URL https://doi.org/10.1145/3159652.3159681

[57] M. Kula, Metadata embeddings for user and item cold-start recommendations, in: T. Bogers, M. Koolen (Eds.), Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015., Vol. 1448 of CEUR Workshop Proceedings, CEUR-WS.org, 2015, pp. 14–21.
URL http://ceur-ws.org/Vol-1448/paper4.pdf

[58] C. Ardito, T. Colafiglio, T. Di Noia, E. D. Sciascio, Brain computer interface, visual tracker and artificial intelligence for a music polyphony generation system, in: C. Ardito, R. Lanzilotti, A. Malizia, H. Petrie, A. Piccinno, G. Desolda, K. Inkpen (Eds.), Human-Computer Interaction - INTERACT 2021 - 18th IFIP TC 13 International Conference, Bari, Italy, August 30 - September 3, 2021, Proceedings, Part V, Vol. 12936 of Lecture Notes in Computer Science, Springer, 2021, pp. 368–371. doi:10.1007/978-3-030-85607-6_39.
URL https://doi.org/10.1007/978-3-030-85607-6_39

[59] E. Loper, S. Bird, Nltk: The natural language toolkit, in: Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, 2002, pp. 63–70.

[60] Y. Vasiliev, Natural Language Processing with Python and SpaCy: A Practical Introduction, No Starch Press, 2020.

[61] C. Baziotis, N. Pelekis, C. Doulkeridis, Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 747–754.

[62] R. Řehůřek, P. Sojka, Software Framework for Topic Modelling with Large Corpora, in: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, Valletta, Malta, 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[63] T. Mikolov, W.-t. Yih, G. Zweig, Linguistic regularities in continuous space word representations, in: Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies, 2013, pp. 746–751.

[64] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146.

[65] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
URL http://jmlr.org/papers/v3/blei03a.html

[66] M. Sahlgren, An Introduction to Random Indexing, in: Proc. of the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE, 2005.

[67] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3982–3992.

[68] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Brew, Huggingface's transformers: State-of-the-art natural language processing, CoRR abs/1910.03771. arXiv:1910.03771.
URL http://arxiv.org/abs/1910.03771

[69] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.

[70] A. Moro, A. Raganato, R. Navigli, Entity Linking meets Word Sense Disambiguation: a Unified Approach, Transactions of the Association for Computational Linguistics 2 (2014) 231–244.

[71] C. Musto, P. Lops, M. de Gemmis, G. Semeraro, Context-aware graph-based recommendations exploiting personalized pagerank, Knowl. Based Syst. 216 (2021) 106806. doi:10.1016/j.knosys.2021.106806.
URL https://doi.org/10.1016/j.knosys.2021.106806

[72] P. Lops, E. Musacchio, C. Musto, M. Polignano, A. Silletti, G. Semeraro, Reproducibility analysis of recommender systems relying on visual features: traps, pitfalls, and countermeasures, in: Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, ACM, 2023. doi:3604915.3609492.
URL https://doi.org/10.1145/3604915.3609492

[73] H. Abdollahpouri, M. Mansoury, R. Burke, B. Mobasher, The unfairness of popularity bias in recommendation, arXiv preprint arXiv:1907.13286.

[74] G. Spillo, C. Musto, M. Polignano, P. Lops, M. de Gemmis, G. Semeraro, Combining graph neural networks and sentence encoders for knowledge-aware recommendations, in: Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2023, Limassol, Cyprus, June 26-29, 2023, ACM, 2023, pp. 1–12. doi:10.1145/3565472.3592965.
URL https://doi.org/10.1145/3565472.3592965

[75] M. Polignano, C. Musto, M. de Gemmis, P. Lops, G. Semeraro, Together is better: Hybrid recommendations combining graph embeddings and contextualized word representations, in: H. J. C. Pampín, M. A. Larson, M. C. Willemsen, J. A. Konstan, J. J. McAuley, J. Garcia-Gathright, B. Huurnink, E. Oldridge (Eds.), RecSys '21: Fifteenth ACM Conference on Recommender Systems, Amsterdam, The Netherlands, 27 September 2021 - 1 October 2021, ACM, 2021, pp. 187–198. doi:10.1145/3460231.3474272.
URL https://doi.org/10.1145/3460231.3474272

[76] H. Cai, V. W. Zheng, K. C.-C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Transactions on Knowledge and Data Engineering 30 (9) (2018) 1616–1637.

[77] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge-Based Systems 151 (2018) 78–94.

[78] E. Palumbo, G. Rizzo, R. Troncy, E. Baralis, M. Osella, E. Ferro, Translational models for item recommendation, in: European Semantic Web Conference, Springer, 2018, pp. 478–490.

[79] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, C. Xu, Recurrent knowledge graph embedding for effective recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 297–305.

[80] W. Song, Z. Duan, Z. Yang, H. Zhu, M. Zhang, J. Tang, Explainable knowledge graph-based recommendation via deep reinforcement learning, arXiv preprint arXiv:1906.09506.

[81] C. Musto, F. Narducci, P. Lops, M. de Gemmis, G. Semeraro, Linked open data-based explanations for transparent recommender systems, Int. J. Hum. Comput. Stud. 121 (2019) 93–107. doi:10.1016/j.ijhcs.2018.03.003.
URL https://doi.org/10.1016/j.ijhcs.2018.03.003