Mab2Rec: Contextual Multi-Armed Bandits for Recommender Systems

Serdar Kadıoğlu

Adj. Assoc. Prof., Computer Science, Brown
Group Vice President, AI Center, Fidelity Investments

skadio.github.io

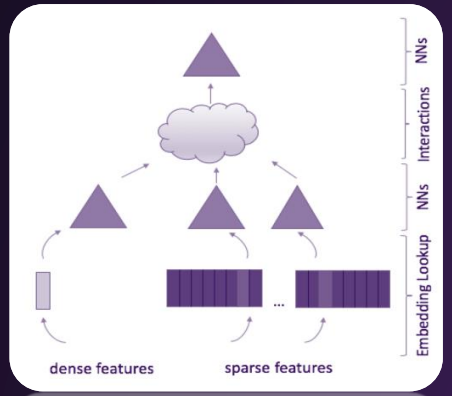
AAAI 2024 | Bridge Program on CP / ML


SOCIAL VIEW
FOOTBALL
DISCOVER
APPS
SEN
RETURN
BACK
DIGITAL/
ANALOG
EXIT
1
4
7
TV

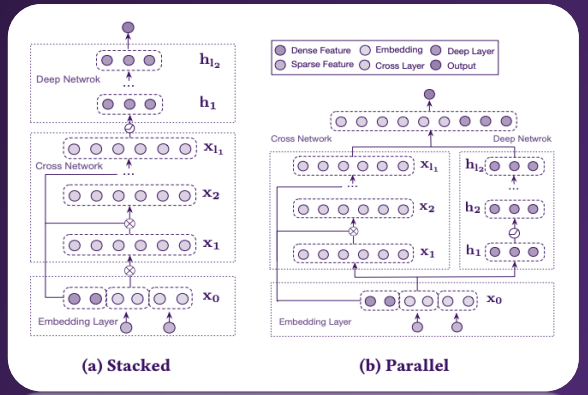
Classical Recommender Systems

Beyond Collaborative Recommenders

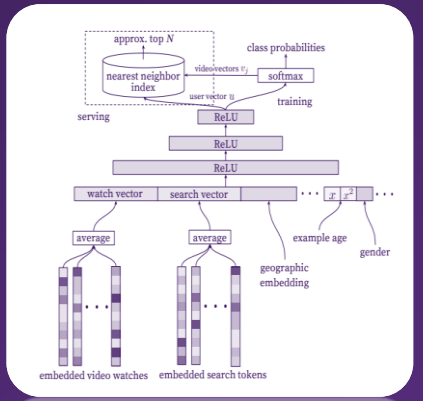
High-Performance Recommender Systems



Meta
DLRM



Google
DCN



YouTube
DNN

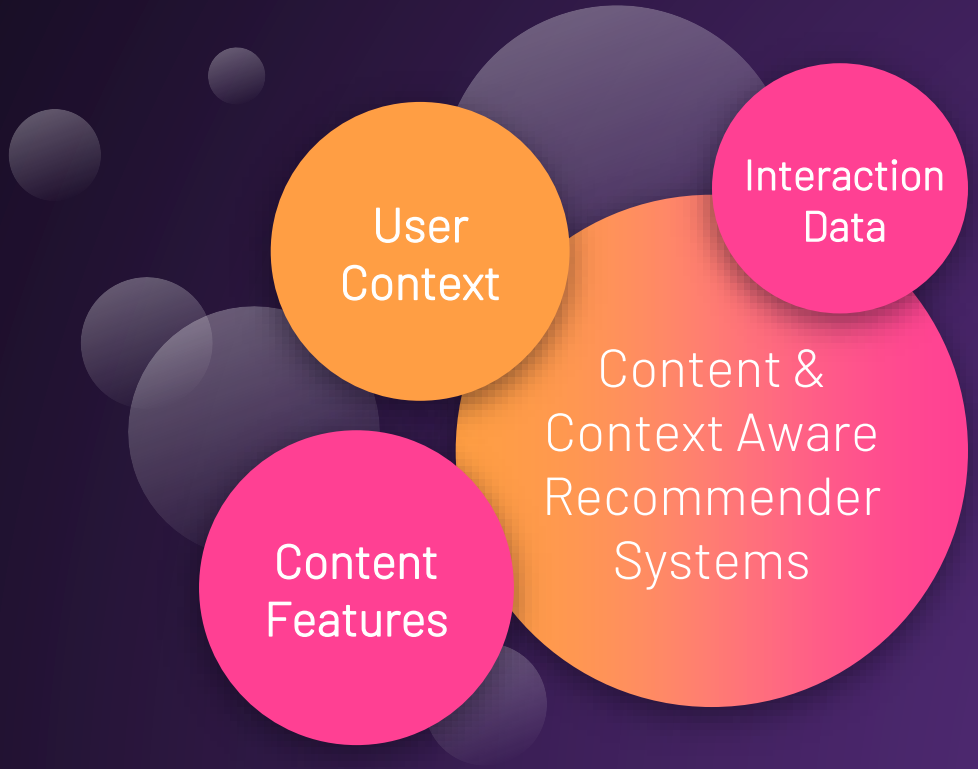
Microsoft Recommenders, NVIDIA Merlin, among others …

Amatriain, X. et. al. Past, present, and future of recommender systems: An industry perspective. ACM RecSys'16

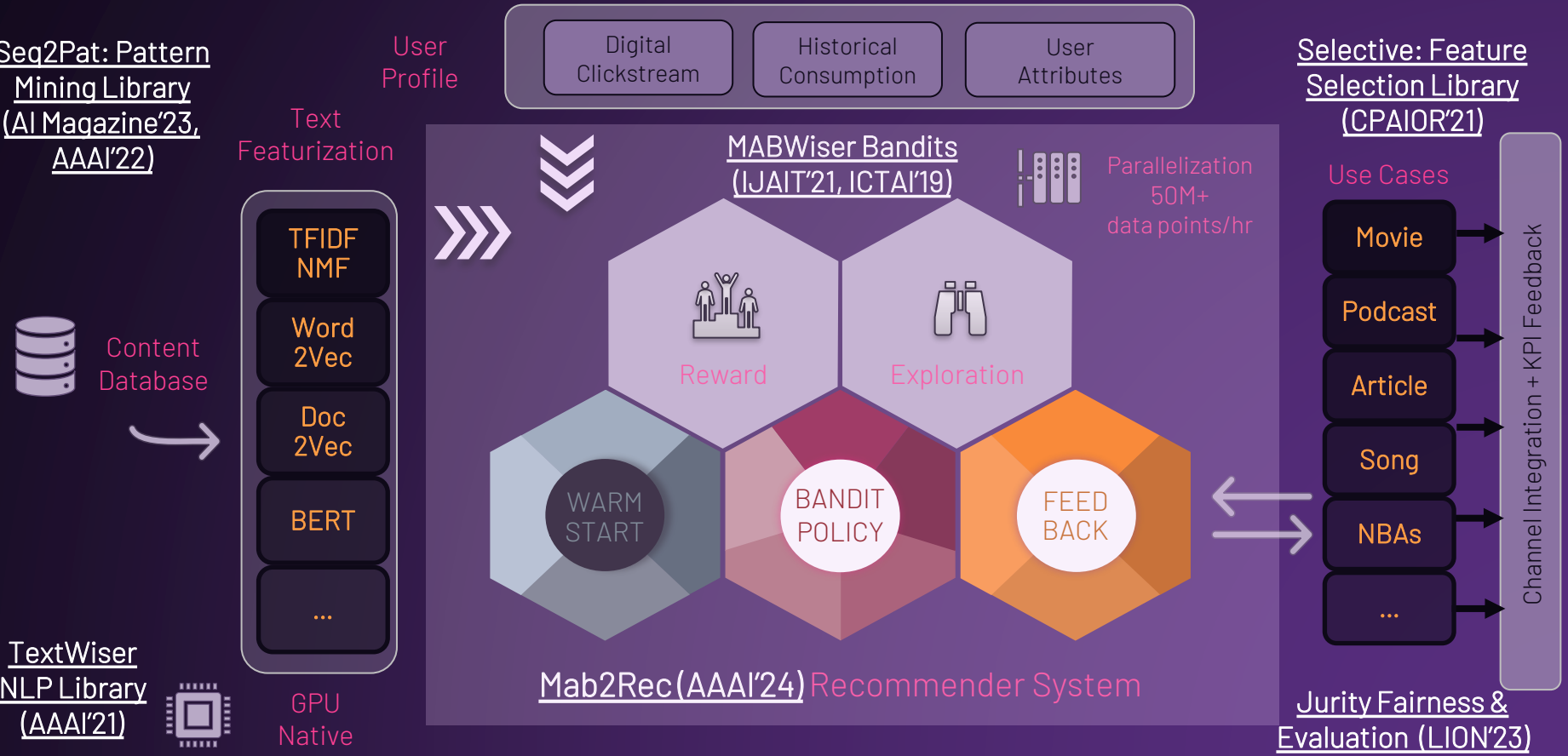
Thousands of items, millions of users, billions of interactions..

***monolithic** frameworks*

Higher-Order Abstractions from Components



Mab2Rec: Higher-Order Abstractions from Components



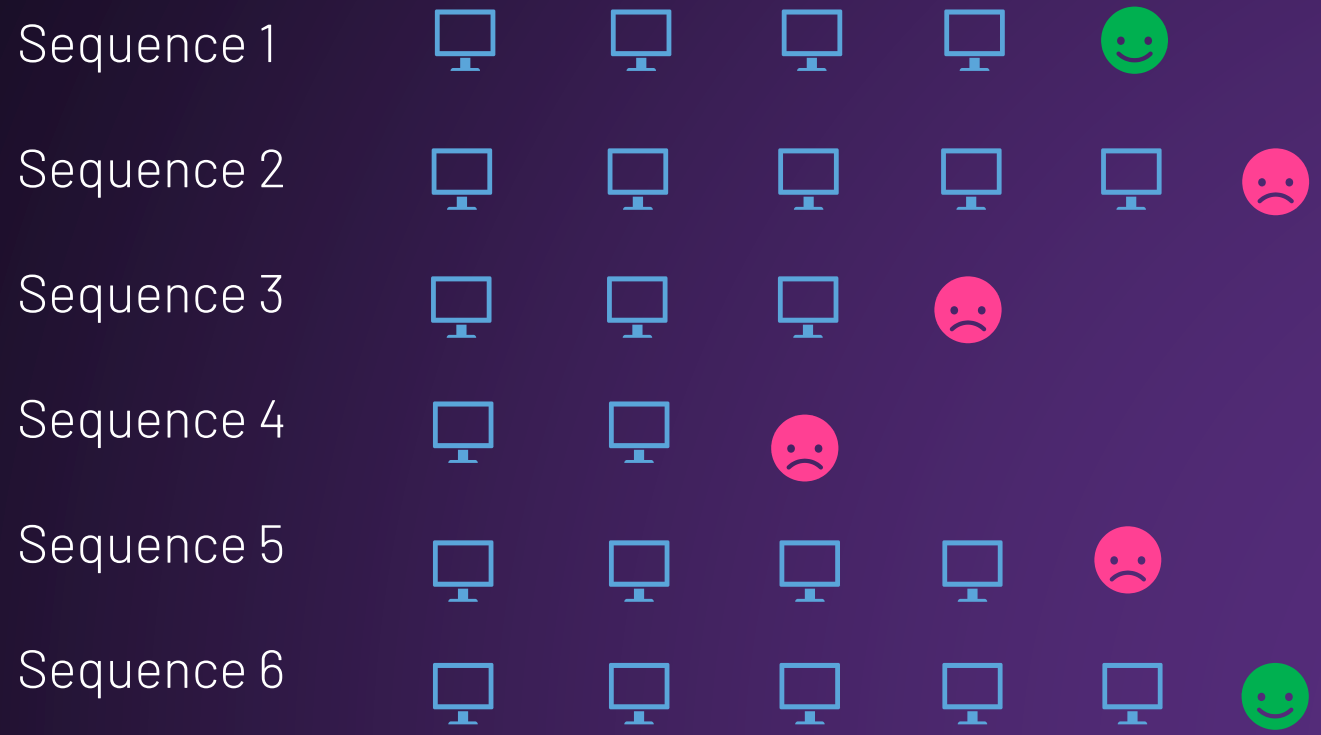
1. User Context Representation

Sequential Pattern Mining

Sequence-to-Pattern Generation

Seq2Pat Library

Interaction Sequence from an Online Bookstore



Goal: Knowledge Extraction

"a logged user who has viewed only traditional, printed books, has been staying in the store from 10 to 25 min, and has opened between 30 and 75 pages, will decide to confirm a purchase with the probability of more than 92 %"

Suchacka, G. and Chodak, G., 2017: Using association rules to assess purchase probability in online stores. Information Systems and e-Business Management, 15(3), pp.751-780.

Seq2Pat Library

Constraint-based Sequential Pattern Mining

Seq2Pat

Sequence to Pattern Generation for Pattern Mining

Discover frequent sequential patterns in large sequence databases

```
# Example to show how to find frequent sequential patterns
# from a given sequence database subject to constraints
from sequential.seq2pat import Seq2Pat, Attribute

# Seq2Pat over 3 sequences
seq2pat = Seq2Pat(sequences=[["A", "A", "B", "A", "D"],
                             ["C", "B", "A"],
                             ["C", "A", "C", "D"]])

# Price attribute corresponding to each item
price = Attribute(values=[[5, 5, 3, 8, 2],
                          [1, 3, 3],
                          [4, 5, 2, 1]])

# Average price constraint
seq2pat.add_constraint(3 <= price.average() <= 4)

# Patterns that occur at least twice (A-D)
patterns = seq2pat.get_patterns(min_frequency=2)
```

pip install seq2pat

AAAI'22, AI Magazine'23, CMU/Fidelity Blog Post

https://github.com/fidelity/seq2pat

Beyond recommenders:

- Dichotomic Pattern Mining (AAAI'22)
- Intent prediction (KDF'22)
- Intrusion detection (Frontiers'22)

2. Content Featurization

Natural Language Processing

Text Featurization

TextWiser Library

Context-Free Grammar for Unification (AAA'21)

TextWiser = Embedding  Transformation(s)

Algorithm 1 Text Embedding Unification Language of TEXTWISER Interface

⟨*start*⟩ ::= ⟨*embed_like*⟩ | ⟨*merge*⟩

⟨*embed_like*⟩ ::= ⟨*embed_option*⟩
| [⟨*embed_option*⟩ , dict]

⟨*embed_option*⟩ ::= ⟨*bow*⟩ | ⟨*doc2vec*⟩ | ⟨*tfidf*⟩ | ⟨*use*⟩

⟨*merge*⟩ ::= { ⟨*transform*⟩ : [⟨*start*⟩ , ⟨*transform_list*⟩]}
| { ⟨*transform*⟩ : [⟨*word_like*⟩ , ⟨*pool_trfm_list*⟩]}
| { ⟨*concatenation*⟩ : [⟨*concat_list*⟩] }

⟨*transform_list*⟩ ::= ⟨*transform_like*⟩
| ⟨*transform_like*⟩ , ⟨*transform_list*⟩

⟨*transform_like*⟩ ::= ⟨*transform_option*⟩
| [⟨*transform_option*⟩ , dict]

⟨*transform_option*⟩ ::= ⟨*lda*⟩ | ⟨*nmf*⟩ | ⟨*svd*⟩ | ⟨*umap*⟩

⟨*word_like*⟩ ::= ⟨*word*⟩ | [⟨*word*⟩ , dict]
| word_option | [word_option , dict]

⟨*word_option*⟩ ::= (see Table 1 word embeddings†)

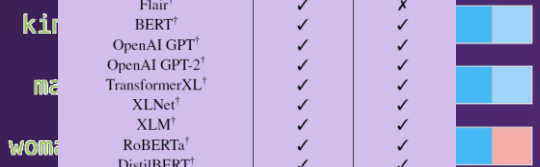
⟨*pool_trfm_list*⟩ ::= ⟨*pool_like*⟩
| ⟨*pool_like*⟩ , ⟨*transform_list*⟩
| ⟨*transform_list*⟩ , ⟨*pool_like*⟩
| ⟨*transform_list*⟩ , ⟨*pool_like*⟩ , ⟨*transform_list*⟩

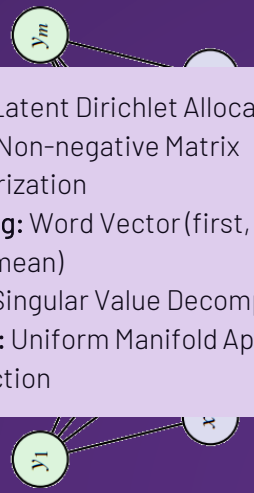
⟨*pool_like*⟩ ::= ⟨*pool*⟩ | [⟨*pool*⟩ , dict]

⟨*concat_list*⟩ ::= ⟨*start*⟩ | ⟨*start*⟩ , ⟨*concat_list*⟩

⟨*foo-bar*⟩ ::= `'foo-bar'` # Omitting trivial terminals

Embeddings	Pre-Trained	Fine-Tuning
Bag of Words (BoW)	✗	✗
TF-IDF	✗	✗
Doc2Vec	✗	✗
Universal Sentence Encoder	✓	✗
Word2Vec†	✓	✓
Character†	✗	✓
BytePair†	✓	✓
ELMo†	✓	✗
Flair†	✓	✗
BERT†	✓	✓
OpenAI GPT†	✓	✓
OpenAI GPT-2†	✓	✓
TransformerXL†	✓	✓
XLNet†	✓	✓
XLM†	✓	✓
RoBERTa†	✓	✓
DistilBERT†	✓	✓
CTRL†	✓	✓
ALBERT†	✓	✓
T5†	✓	✓
XLM-RoBERTa†	✓	✓
BART†	✓	✓
ELECTRA†	✓	✓
DialoGPT†	✓	✓
Longformer†	✓	✓





LDA: Latent Dirichlet Allocation
NMF: Non-negative Matrix Factorization
Pooling: Word Vector (first, last, min, max, mean)
SVD: Singular Value Decomposition
UMAP: Uniform Manifold Approx. and Projection

TextWiser

Context-Free Grammar-based NLP Text Featurization

25+ embeddings
5 transformations
100+ pretrained models

```
from textwiser import TextWiser, Embedding, Transformation, WordOptions, PoolOptions

# Data
documents = ["Some document", "More documents. Including multi-sentence documents."]

# Model: TFIDF `min_df` parameter gets passed to sklearn automatically
emb = TextWiser(Embedding.Tfidf(min_df=1))

# Model: TFIDF followed with an NMF + SVD
emb = TextWiser(Embedding.Tfidf(min_df=1), [Transformation.NMF(n_components=30),
                                            Transformation.SVD(n_components=10)])

# Model: Word2Vec
emb = TextWiser(Embedding.Word(WordOption.word2vec))

# Model: BERT with the pretrained bert-base-uncased embedding min pooled
emb = TextWiser(Embedding.Word(word_option=WordOptions.bert),
                Transformation.Pool(pool_option=PoolOptions.min))

# Feature vectors
vectors = emb.fit_transform(documents)
```

pip install textwiser
AAAI 2021
https://github.com/fidelity/textwiser

Beyond recommenders:

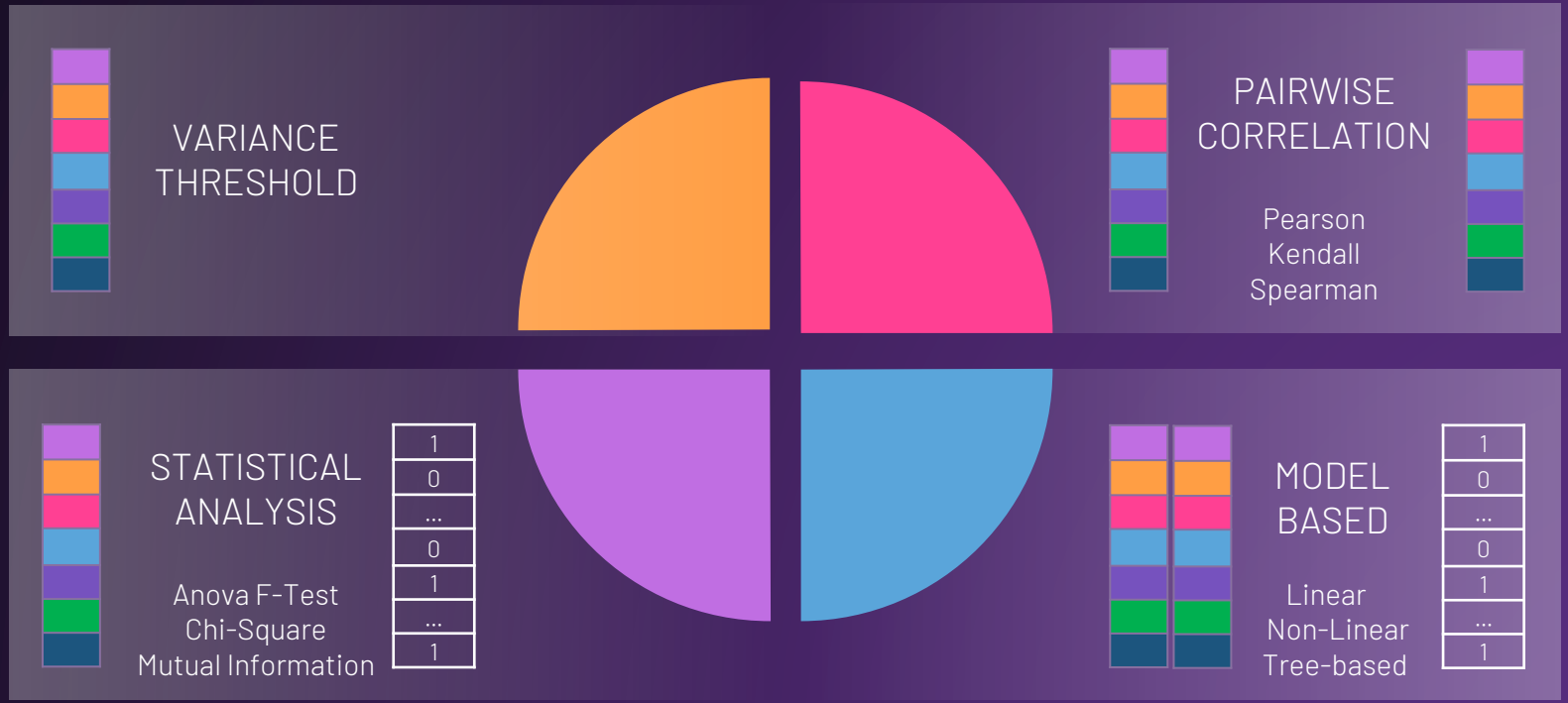
- EaSe: Embeddings-as-a-Service (AAAI'21)

3. Feature Selection

How to eliminate redundant correlated features?

Selective Library

Feature Selection Techniques



Selective

White-box Feature Selection

25+ supervised and unsupervised techniques for classification regression and text-based tasks

```python
from sklearn.datasets import load_boston
from feature.utils import get_data_label
from feature.selector import Selective, SelectionMethod

# Data
data, label = get_data_label(load_boston())

# Feature selectors from simple to more complex
selector = Selective(SelectionMethod.Variance(threshold=0.0))
selector = Selective(SelectionMethod.Correlation(threshold=0.5, method="pearson"))
selector = Selective(SelectionMethod.Statistical(num_features=3, method="anova"))
selector = Selective(SelectionMethod.Linear(num_features=3, regularization="none"))
selector = Selective(SelectionMethod.TreeBased(num_features=3))

# Feature reduction
subset = selector.fit_transform(data, label)
print("Reduction:", list(subset.columns))
print("Scores:", list(selector.get_absolute_scores()))
```

pip install selective
CPAIOR 2021, DSO@IJCAI'22
https://github.com/fidelity/selective

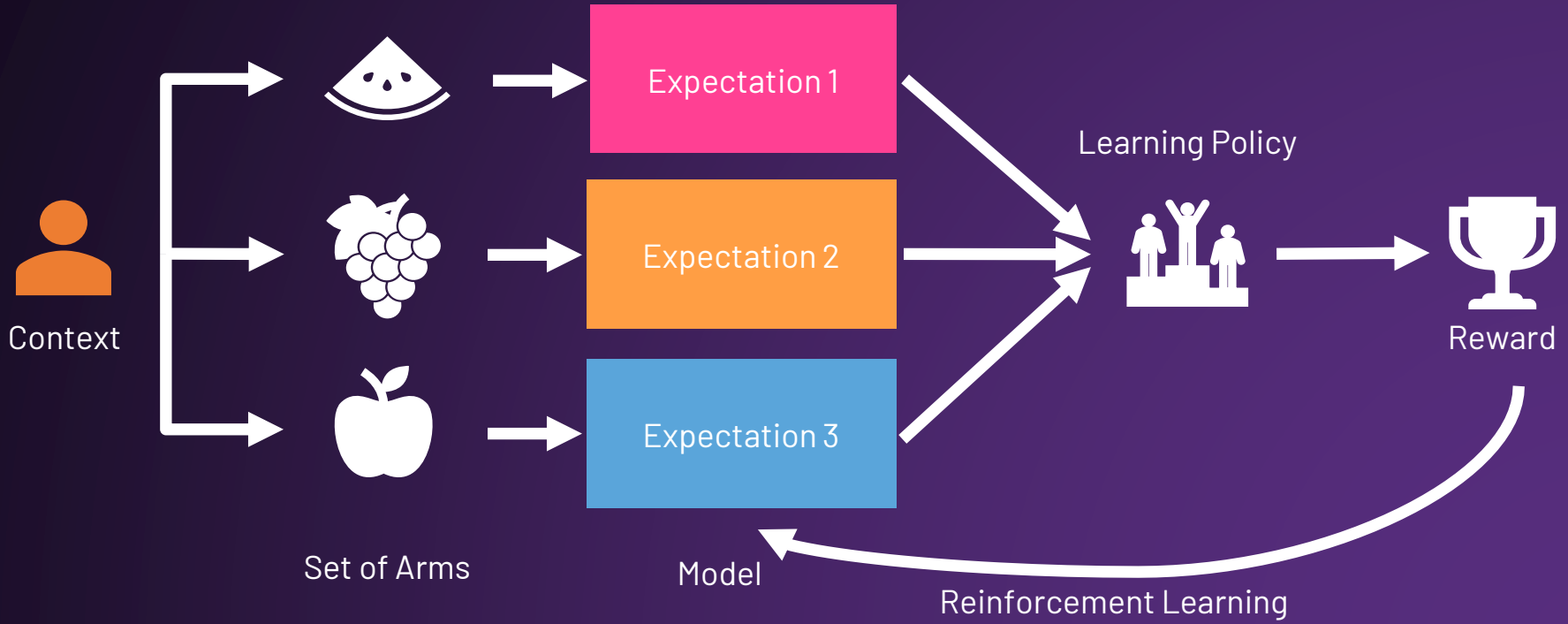
Beyond recommenders:
- Experiment design (CPAIOR'21)
- Active learning (DSO@IJCAI'22)

4. Recommendation Algorithm

Multi-armed bandit framework

MABWiser Library

Multi-Armed Bandits



MAB Applications

Domain	Arms	Reward	Reference
Advertisement Optimization	Ads	Revenue	Chapelle & Li. NeurIPS, 2011.
Movie Artwork Personalization	Artwork	Play Movie	Chandrashekar et al. Netflix Technology Blog, 2017.
News Article Recommendation	Articles	Click	Li, Chu, Langford & Schapiro. WWW, 2010.
Clinical Trial	Trial Group	Patient Outcome	Berry. Nature Reviews Drug Discovery, 2006.
Hyper-Parameter Tuning	Values	Model Performance	Yelp MOE, 2014.

MABWiser

Parallelizable Contextual Multi-Armed Bandits

8 learning policies
5 neighborhood policies, and their combination

```python
from mabwiser.mab import MAB, LearningPolicy

# Data
arms = ['Movie1', 'Movie2']
decisions = ['Movie1', 'Movie1', 'Movie2', 'Movie1']
rewards = [20, 17, 25, 9]
contexts = [ [..], [..], [..], [..] ]

# Model
mab = MAB(arms, LearningPolicy.UCB1())

# Train
mab.fit(decisions, rewards, contexts)

# Test
recommendation = mab.predict(test_context=[..])
```

pip install mabwiser
TMLR'22, IJAIT'21, ICTAI'19
https://github.com/fidelity/mabwiser

Beyond recommenders:
- ALNS (Wouda et. al. 2023)
- GOLEM (Nikitin et al. 2021)

5. Fairness & Performance Evaluation

Fairness evaluation & bias mitigation

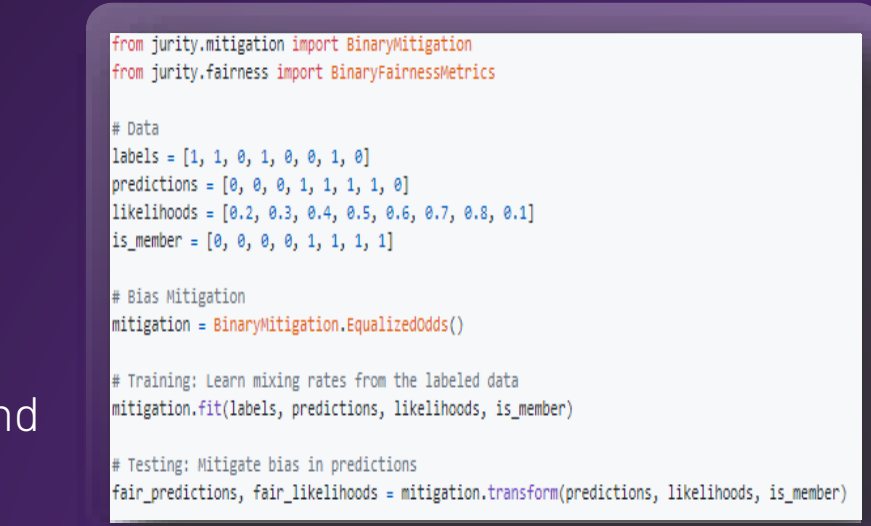
recommendation metrics

Jurity Library

Jurity

Fairness & Performance

Binary & multi-class fairness evaluation, bias mitigation, classification, and recommender system metrics



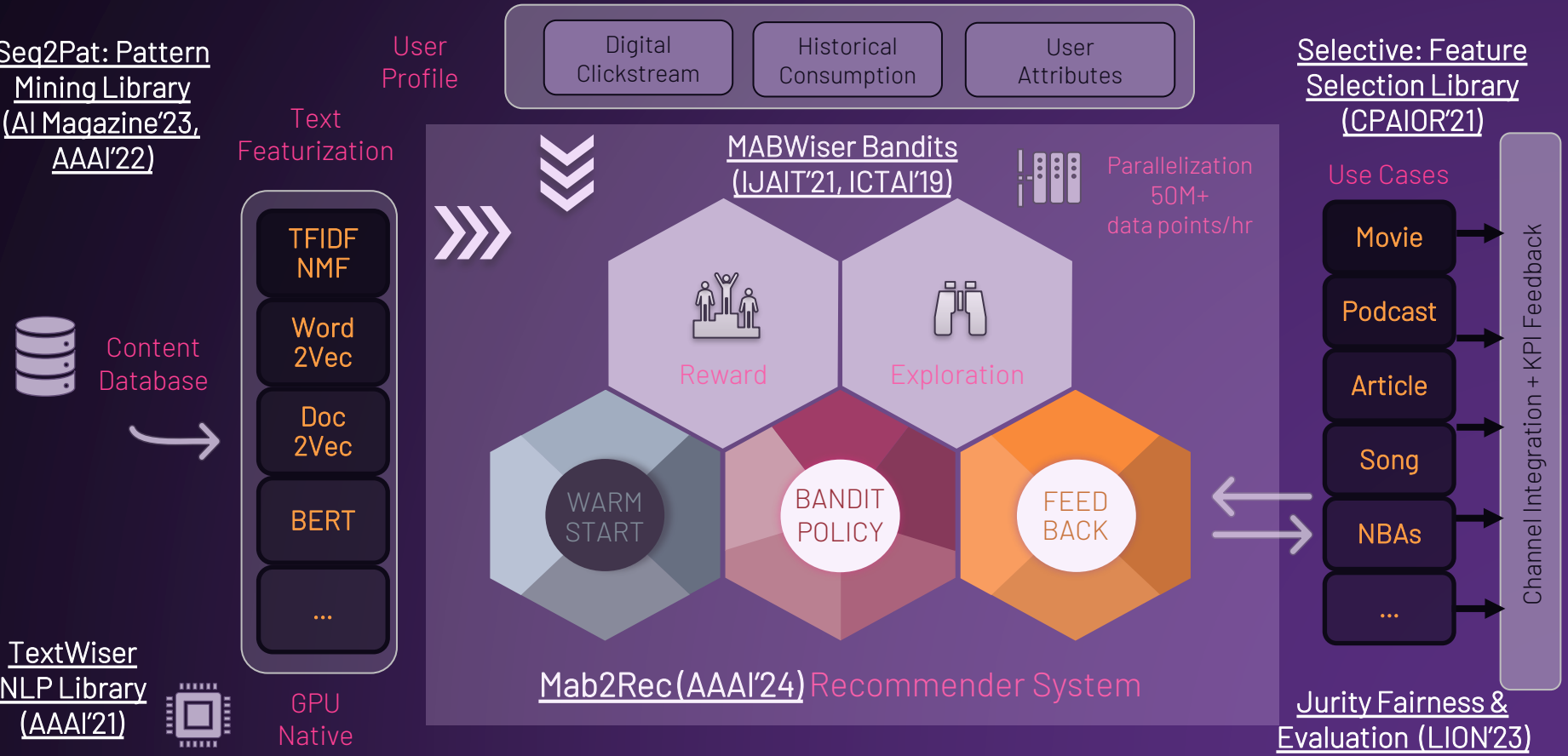
```python
from jurity.mitigation import BinaryMitigation
from jurity.fairness import BinaryFairnessMetrics

# Data
labels = [1, 1, 0, 1, 0, 0, 1, 0]
predictions = [0, 0, 0, 1, 1, 1, 1, 0]
likelihoods = [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.1]
is_member = [0, 0, 0, 0, 1, 1, 1, 1]

# Bias Mitigation
mitigation = BinaryMitigation.EqualizedOdds()

# Training: Learn mixing rates from the labeled data
mitigation.fit(labels, predictions, likelihoods, is_member)

# Testing: Mitigate bias in predictions
fair_predictions, fair_likelihoods = mitigation.transform(predictions, likelihoods, is_member)
```

pip install jurity
LION'23, CIKM'22, ICMLA'21
https://github.com/fidelity/jurity

Beyond recommenders:

- Absence of Ground Truth (ICMLA'21)
- Absence of protected membership (LION'23)
- Diversity via bias mitigation (CIKM'22)

Mab2Rec: Higher-Order Abstractions from Components



pip install mab2rec

docker pull ghcr.io/skadio/atlas docker:latest

Tutorial Notebooks

Star our library and stay up to date! **500K+ downloads**

Special thanks to my collaborators!

[AAAI'24] Building higher order abstractions from recommender components
[AI Magazine'23] Seq2Pat: Bridging pattern mining and machine learning
[IJCAI'22] Active learning meets optimized item selection
[CPAIOR'21] Optimized item selection to boost exploration for recommender systems
[AAAI'21] Representing the unification of text featurization using a context-free grammar
[AAAI'22] Seq2Pat: Sequence-to-Pattern generation
[AAAI'22] Dichotomic pattern mining for prediction from clickstream datasets
[ICMLA'21] Surrogate ground truth to enhance binary fairness in uplift modelling
[IJAIT'21] Parallelizable contextual multi-armed bandits
[JDSA'21] Modeling uncertainty to improve personalized recommendations via Bayesian DL
[ICTAI'19] Bayesian DL-based exploration-exploitation for personalized recommendations

Recommenders	Mab2Rec:	https://github.com/fidelity/mab2rec
Multi-armed Bandits	MABWiser:	https://github.com/fidelity/mabwiser
NLP/Embeddings	TextWiser:	https://github.com/fidelity/textwiser
Pattern Mining	Seq2Pat:	https://github.com/fidelity/seq2pat
Feature Selection	Selective:	https://github.com/fidelity/selective
AI Fairness & Bias	Jurity:	https://github.com/fidelity/jurity

skadio.github.io

Doruk Kilitcioglu | Bernard Kleynhans | Pablo Colunga

Filip Michalsky | Du Cheng | Xin Wang

Amin Hosseininasab | Willem-Jan van Hoeve | Irmak Bukey