

# KDD 2020 Notes

Ilnur Shugaepov  
VK.com

August 2020

# Оглавление

1	Tutorials	5
I	Advertising	12
2	Predicting conversions in display advertising based on URL embeddings	13
3	On the Effectiveness of Self-supervised Pre-training for Modeling User Behavior Sequences	16
4	General-Purpose User Embeddings based on Mobile App Usage	19
5	Hubble: an Industrial System for Audience Expansion in Mobile Marketing	22
6	AutoFIS: Automatic Feature Interaction Selection in Factorization Models for CTR Prediction	25
7	Другие работы	27
II	Recommender Systems	29
8	On Sampled Metrics for Item Recommendation	30
9	Temporal-Contextual Recommendation in Real-Time	33
10	SimClusters: Community-Based Representations for Heterogeneous Recs at Twitter	36
11	Learning to Cluster Documents into Workspaces Using Large Scale Activity Logs	37
12	Другие работы	40
III	Deep Learning	44
13	Retrospective Loss: Looking Back to Improve Training of Deep Neural Networks	45
14	Correlation Networks for Extreme Multi-label Text Classification	47
15	Другие работы	49

IV	Big Data Infrastructure	51
16	To Tune or Not to Tune? In Search of Optimal Configurations for Data Analytics	52
17	Lumos: A Library for Diagnosing Metric Regressions in Web-Scale Applications	55

Этот документ содержит мои заметки о некоторых статьях и туториалах с прошедшего KDD. Заметки оформлены в формате конспектов по типу тех, которые можно найти тут <https://vk.com/papersreaders>.

В документе 11 полноценных конспектов и еще 10 мини конспектов.

Для себя я выделил несколько направлений, на работы в которых я смотрел в первую очередь:

- Advertising — все что связано с рекламой
- Recommender Systems — задачи связанные с рекомендательными системами
- Deep Learning

Также на конференции было много статей и туториалов, посвященных обучению на графах.

# Tutorials

---

1.1	[Amazon AWS] Scalable Graph Neural Networks with Deep Graph Library . . . . .	5
1.2	[LinkedIn] Deep Learning for Search and Recommender Systems . . . . .	6
1.3	Graph Representation Learning . . . . .	7
1.4	[Google] Neural Structured Learning . . . . .	7
1.5	Causal Inference Meets Machine Learning . . . . .	7
1.6	[Facebook] Building Recommender Systems with PyTorch . . . . .	8
1.7	[LinkedIn] Image and Video Understanding for Recommendation and Spam Detection Systems . . . . .	9
1.8	[Amazon] Practical AutoML for Tabular, Text, and Image Data . . . . .	10
1.9	[Microsoft] AutoML . . . . .	10
1.10	Reinforcement Learning . . . . .	10
1.11	How to Calibrate Your Neural Network Classifier . . . . .	10
1.12	Robust Deep Learning Methods for Anomaly Detection . . . . .	11

---

Так как конференция проходила в удаленном формате, то для большинства tutorиалов были пред-записаны видео, поэтому получилось посмотреть записи почти всех tutorиалов, которые меня за-интересовали.

Ниже список tutorиалов, которые показались мне наиболее интересными, с кратким описанием.

## 1.1 [Amazon AWS] Scalable Graph Neural Networks with Deep Graph Library

Link: <https://github.com/dglai/KDD20-Hands-on-Tutorial>

Paper: <https://arxiv.org/pdf/1909.01315.pdf>

Очень практический tutorиал про обучение графовых нейросетей.

Речь в основном идет про обучение GraphSAGE<sup>1</sup> с помощью фреймворка DGL<sup>2</sup> на графах разного размера: начиная с маленьких графов и обучения на CPU, заканчивая обучением на огромных графах с использованием нескольких машин.

Tutorиал наглядно показывает, что обучать GNN используя DGL довольно просто.

---

<sup>1</sup>[https://vk.com/papersreaders?w=wall-154085965\\_143](https://vk.com/papersreaders?w=wall-154085965_143)

<sup>2</sup><https://www.dgl.ai>

## Videos

1. Краткий обзор задач и методов в GNNs — <https://www.youtube.com/watch?v=Soa-66W1CEQ>
2. Вводное видео — <https://www.youtube.com/watch?v=JMF4KAk00zo>
3. Обучение GraphSAGE с помощью DGL+PyTorch на задачах node classification и link prediction (на gpu в том числе) — [https://www.youtube.com/watch?v=OEBk\\_HvKNcQ](https://www.youtube.com/watch?v=OEBk_HvKNcQ)
4. Обучение GraphSAGE на больших графах (сэмплирование соседей, dataloader'ы) на node classification и link prediction (single machine - single/multiple gpu(s)) — <https://www.youtube.com/watch?v=2W0UIJX3GV8>

## 1.2 [Linkedin] Deep Learning for Search and Recommender Systems

Link: <https://sites.google.com/view/kdd20tutorial-deepsnr>

Тьюриал от Linkedin про DL методы в поиске и рекомендациях: обзор основных задач, методы решений и Hands On с использованием их фреймворка DeText<sup>3</sup>.

## Videos

1. Краткий обзор тьюриала — [https://www.youtube.com/watch?time\\_continue=2&v=Dp2Iwd4FwsU&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=2&v=Dp2Iwd4FwsU&feature=emb_logo)
2. Обзор задач возникающих в поиске, и методов их решения — [https://www.youtube.com/watch?v=LwjIKqNsQjc&feature=emb\\_logo](https://www.youtube.com/watch?v=LwjIKqNsQjc&feature=emb_logo). В основе всего Sequential Modeling. Задачи
  - Query Intent Classification (есть в DeText) - понять к чему относится запрос, поиск людей/вакансий/постов итд.
  - Query Completion (есть в DeText) - автодополнение поискового запроса
  - Query Suggestion - рекомендация запросов
  - Sequence Labeling - выделение в тексте запроса сущностей (NER)
  - Ranking (есть в DeText) - ранжирование отобранных документов-кандидатов
3. Hands On: query intent classification with DeText — [https://www.youtube.com/watch?v=QZ5fb03NA64&feature=emb\\_logo](https://www.youtube.com/watch?v=QZ5fb03NA64&feature=emb_logo)
4. Hands On: query completion with DeText — [https://www.youtube.com/watch?v=0Lfe0cZ7y6A&feature=emb\\_logo](https://www.youtube.com/watch?v=0Lfe0cZ7y6A&feature=emb_logo)
5. Candidate Retrieval: обзор классических (lexical matching) и современных (embeddings based) методов отбора документов-кандидатов — [https://www.youtube.com/watch?v=KECp0syOH28&feature=emb\\_logo](https://www.youtube.com/watch?v=KECp0syOH28&feature=emb_logo)
6. Hands On: candidates retrieval with Elasticsearch — [https://www.youtube.com/watch?v=MXbttPyNON0&feature=emb\\_logo](https://www.youtube.com/watch?v=MXbttPyNON0&feature=emb_logo)
7. Ranking: Metrics (NDCG, MAP, MRR), Learning to Rank (loss functions, algorithms) — [https://www.youtube.com/watch?v=mw\\_zdRLbhJM&feature=emb\\_logo](https://www.youtube.com/watch?v=mw_zdRLbhJM&feature=emb_logo)
8. Hands On: Learning to Rank with DeText — [https://www.youtube.com/watch?v=C-erioRqg4Y&feature=emb\\_logo](https://www.youtube.com/watch?v=C-erioRqg4Y&feature=emb_logo)

---

<sup>3</sup><https://github.com/linkedin/detext>

## 1.3 Graph Representation Learning

Video: [https://www.youtube.com/watch?time\\_continue=27&v=J9AhyrTs2Nk&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=27&v=J9AhyrTs2Nk&feature=emb_logo)

Book: [https://www.cs.mcgill.ca/~wlh/grl\\_book/](https://www.cs.mcgill.ca/~wlh/grl_book/)

Обзорный рассказ об истории развития GRL, о современных методах и приложениях, о текущих челенджах и открытых вопросах.

## 1.4 [Google] Neural Structured Learning

Практический tutorial про регуляризацию обучения с помощью информации представленной в виде графов (реализация Graph-RISE<sup>4</sup>) на TF.

References:

- Neural Graph Learning: Training Neural Networks Using Graphs<sup>5</sup>
- Graph-RISE: Graph-Regularized Image Semantic Embedding<sup>6</sup>

Videos [https://www.youtube.com/watch?v=iyONhiDuCKE&feature=emb\\_logo](https://www.youtube.com/watch?v=iyONhiDuCKE&feature=emb_logo)

[https://www.youtube.com/watch?v=MQ3Ec-19\\_bc](https://www.youtube.com/watch?v=MQ3Ec-19_bc)

[https://www.youtube.com/watch?v=qmY-CANCTdk&feature=emb\\_logo](https://www.youtube.com/watch?v=qmY-CANCTdk&feature=emb_logo)

[https://www.youtube.com/watch?v=h-nTn4nWZo8&feature=emb\\_logo](https://www.youtube.com/watch?v=h-nTn4nWZo8&feature=emb_logo)

[https://www.youtube.com/watch?v=qTC470NN1Z0&feature=emb\\_logo](https://www.youtube.com/watch?v=qTC470NN1Z0&feature=emb_logo)

[https://www.youtube.com/watch?v=9Mwf0VaWGwE&feature=emb\\_logo](https://www.youtube.com/watch?v=9Mwf0VaWGwE&feature=emb_logo)

[https://www.youtube.com/watch?v=FqOWxvvYSA8&feature=emb\\_logo](https://www.youtube.com/watch?v=FqOWxvvYSA8&feature=emb_logo)

[https://www.youtube.com/watch?v=uK4j8CtIXgs&feature=emb\\_logo](https://www.youtube.com/watch?v=uK4j8CtIXgs&feature=emb_logo)

## 1.5 Causal Inference Meets Machine Learning

Link: <http://kdd2020tutorial.thumedia lab.com>

Paper: <https://arxiv.org/abs/2002.02770>

Тutorial про то как оценивать эффект изменений когда либо нет возможности провести АВТ либо когда выборки treatment и control имеют разное распределение.

Как получить несмещенную оценку эффекта и убрать всякие разные bias'ы.

[https://www.youtube.com/watch?v=g\\_2dMQCU0zo&feature=emb\\_logo](https://www.youtube.com/watch?v=g_2dMQCU0zo&feature=emb_logo)

[https://www.youtube.com/watch?v=I6yi\\_wd4r94&feature=emb\\_logo](https://www.youtube.com/watch?v=I6yi_wd4r94&feature=emb_logo)

[https://www.youtube.com/watch?v=f2dvVpDxD3A&feature=emb\\_logo](https://www.youtube.com/watch?v=f2dvVpDxD3A&feature=emb_logo)

[https://www.youtube.com/watch?v=J4B00WZH5A0&feature=emb\\_logo](https://www.youtube.com/watch?v=J4B00WZH5A0&feature=emb_logo)

[https://www.youtube.com/watch?v=\\_DwdpmZkY1M&feature=emb\\_logo](https://www.youtube.com/watch?v=_DwdpmZkY1M&feature=emb_logo)

[https://www.youtube.com/watch?v=roVS6LQS6tk&feature=emb\\_logo](https://www.youtube.com/watch?v=roVS6LQS6tk&feature=emb_logo)

---

<sup>4</sup>[https://vk.com/papersreaders?w=wall-154085965\\_180](https://vk.com/papersreaders?w=wall-154085965_180)

<sup>5</sup><https://storage.googleapis.com/pub-tools-public-publication-data/pdf/bbd774a3c6f13f05bf754e09aa45e7aa6faa08a8.pdf>

<sup>6</sup><https://arxiv.org/pdf/1902.10814.pdf>

## 1.6 [Facebook] Building Recommender Systems with PyTorch

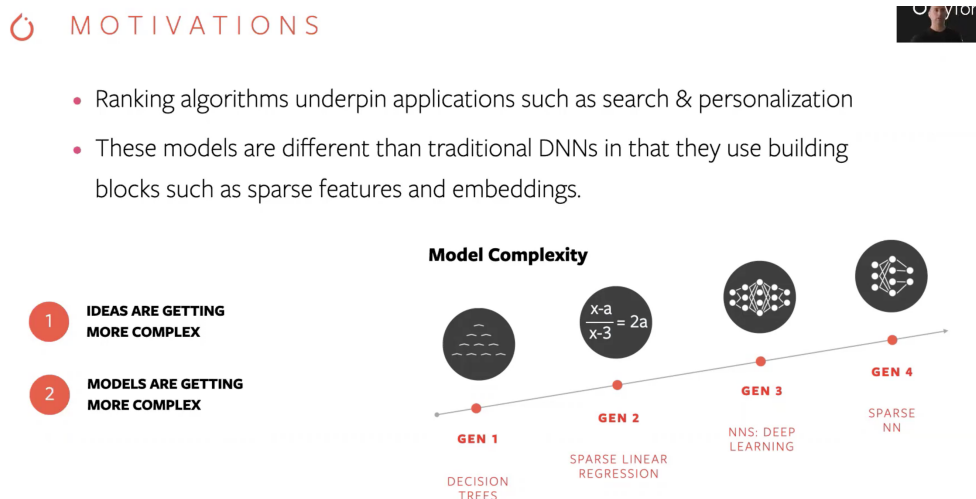
Link: [https://github.com/pytorch/workshops/tree/master/KDD\\_2020](https://github.com/pytorch/workshops/tree/master/KDD_2020)

По большей части обзорный tutorial про обучение рек. систем с помощью PyTorch DLRM<sup>78</sup> (касаются вопросов распределенного обучения).

Немного про интерпретацию нейросетевых PyTorch моделей с помощью библиотеки Captum<sup>9</sup>, и про PyTorch MMF<sup>10</sup> — фреймворк для обучения моделей, которые на вход получают картинки И текст (например Visual Question Answering).

### Videos

1. Введение в PyTorch (design principles, ecosystem), кратко о том почему PyTorch удобен для обучения очень больших моделей (уменьшение размера моделей: pruning, quantization; надежное распределенное обучение) — [https://www.youtube.com/watch?v=AVyJAgSSSQ&feature=emb\\_logo](https://www.youtube.com/watch?v=AVyJAgSSSQ&feature=emb_logo)
2. Введение в DLRMs и распределенное обучение больших моделей (как data так и model parallelism). Более близкий взгляд на саму DLRM модель (прикольные способы сжатия эмбедингов) — [https://www.youtube.com/watch?v=Q-1hVeB2BWo&feature=emb\\_logo](https://www.youtube.com/watch?v=Q-1hVeB2BWo&feature=emb_logo)



Также полезные ссылки: Microsoft Recommenders (<https://arxiv.org/pdf/2008.13528.pdf>), NVIDIA Merling (<https://github.com/NVIDIA/HugeCTR>)

3. Немного про Captum - библиотека для интерпретации самых разных PyTorch моделей (например, выделение важных частей в инпуте) работающих с самыми разными входными данными (табличные, текст, картинки) — [https://www.youtube.com/watch?v=fzrq1QGwsVo&feature=emb\\_logo](https://www.youtube.com/watch?v=fzrq1QGwsVo&feature=emb_logo)

<sup>7</sup><https://arxiv.org/pdf/1906.00091.pdf>

<sup>8</sup><https://github.com/facebookresearch/dlrm>

<sup>9</sup><https://github.com/pytorch/captum>

<sup>10</sup><https://github.com/facebookresearch/mmf>



## WHAT DOES THE CAPTUM LIBRARY OFFER?

Attribution algorithms to interpret:

- + Output predictions with respect to inputs
- + Output predictions with respect to all neurons in the layers
- + Neurons with respect to inputs

Currently we supports gradient and perturbation based algorithms

4. Hands On: MMF фреймворк для обучения моделей, которые на вход получают картинки И текст (например Visual Question Answering) — [https://www.youtube.com/watch?v=quNtfWwNmPw&feature=emb\\_logo](https://www.youtube.com/watch?v=quNtfWwNmPw&feature=emb_logo)

## 1.7 [Linkedin] Image and Video Understanding for Recommendation and Spam Detection Systems

Тьюриал состоит из двух больших частей - вводная часть и приложения в LinkedIn.

Вводную имеет смысл смотреть если хочется получить высокоуровневое представление о задачах и методах, которые возникают в Image and Video Understanding.

Самое интересное - приложения в LinkedIn.

### Вводная часть

1. Introduction to image and video usage in recommendations, search and spam detection at LinkedIn — <https://vimeo.com/445696336>
2. Обзор основных задач связанных с **Image Understanding** - classification, segmentation, objects detection, captioning.  
Обзор методов получения представлений для изображений (с 10х годов и до наших дней) — <https://vimeo.com/445696409>
3. **Image representations** — Кратко про NASNet (поиск архитектуры для ImageNet); EfficientNet (increase the efficiency, while maintaining accuracy) - very good choice if you are looking for a smaller network; Transfer Learning; Deep Metric Learning (siamese nets, contrastive loss); ViLBERT — <https://vimeo.com/445696505>
4. **Video Understanding** — Tasks (classification, segmentation, captioning) — <https://vimeo.com/445696678>
5. Video Understanding - Temporal Deep Learning Methods — <https://vimeo.com/445696755>, <https://vimeo.com/445696881>
6. Video applications for **speech recognition** — обзор современных и не очень методов — <https://vimeo.com/445696966>

### Applications at LinkedIn

1. Кратко про **пайплайны** (оффлайн/онлайн части) связанные с обработкой изображений и видео. **Video Search** - speech-to-text — <https://vimeo.com/445697241>

2. **Feed and Ads Modeling** — чуть подробнее о том, как получают эмбединги для видео (коллаборативные и не очень) — <https://vimeo.com/445697405>
3. **Feed and Ads** — Member Video Affinity (предсказание вероятности взаимодействия пользователя с видео) as a feature in Ranking Model — <https://vimeo.com/445697492>
4. **Spam Detection** — пайплайны и модели — <https://vimeo.com/445697551>

## 1.8 [Amazon] Practical AutoML for Tabular, Text, and Image Data

Link: <https://jwmueller.github.io/KDD20-tutorial/>

Videos: <https://www.youtube.com/playlist?list=PLlG1URKFtW6jfdjxBoZyYrr1Lm0QzeS7h>

AutoGluon: <https://github.com/awsml/autogluon/>

Hands On tutorial по использованию AutoGluon.

## 1.9 [Microsoft] AutoML

Video: [https://www.youtube.com/watch?time\\_continue=537&v=OoNeBuXkLeg&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=537&v=OoNeBuXkLeg&feature=emb_logo)

Microsoft Azure AutoML Platform from a user point of view.

## 1.10 Reinforcement Learning

Всякое разное про современный RL

<https://www.youtube.com/watch?v=-Y6fmyMWIQ0>

[https://www.youtube.com/watch?time\\_continue=3&v=mLT3AgrOSsQ&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=3&v=mLT3AgrOSsQ&feature=emb_logo)

[https://www.youtube.com/watch?time\\_continue=3&v=LIKrJnCo5fY&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=3&v=LIKrJnCo5fY&feature=emb_logo)

## 1.11 How to Calibrate Your Neural Network Classifier

Link: <https://github.com/nplan-io/kdd2020-calibration>

Video: [https://www.youtube.com/watch?v=I4jtkv07DFw&feature=emb\\_logo](https://www.youtube.com/watch?v=I4jtkv07DFw&feature=emb_logo)

Основная статья: <https://arxiv.org/pdf/1706.04599.pdf>

О чем tutorial

- Что такое калибровка и зачем она нужна
- Reliability diagram<sup>11</sup> - как инструмент для проверки калибровки модели; Expected/Maximum Calibration Error - численный способ оценить калибровку модели
- Calibration for multi-class case
- Calibration Methods — isotonic regression, Platt scaling, temperature scaling

Tutorial содержит как теоретическую часть так и code sessions.

---

<sup>11</sup><https://towardsdatascience.com/introduction-to-reliability-diagrams-for-probability-calibration-ed785b3f5d44>

## 1.12 Robust Deep Learning Methods for Anomaly Detection

Link: <https://github.com/raghavchalapathy/KDD-Tutorials-2020-Deep-Robust-Anomaly-Detection>

Использование автоэнкодеров для решения задачи детекции аномалий.

### Videos

1. Настройка окружения в Google Colab — [https://www.youtube.com/watch?time\\_continue=4&v=8TY1UHYcjzk&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=4&v=8TY1UHYcjzk&feature=emb_logo), [https://www.youtube.com/watch?v=H0t09clEkU4&feature=emb\\_logo](https://www.youtube.com/watch?v=H0t09clEkU4&feature=emb_logo)
2. Обзор всяких разных Autoencoder'ов (AE, VAE, Adversarial AE, Wasserstein AE), введение в использование AE для anomaly detection — [https://www.youtube.com/watch?v=UdWTRdhIdQY&feature=emb\\_logo](https://www.youtube.com/watch?v=UdWTRdhIdQY&feature=emb_logo)
3. Hands On: AE for anomaly detection with toy dataset (Keras) — [https://www.youtube.com/watch?v=BF3Plbp08sQ&feature=emb\\_logo](https://www.youtube.com/watch?v=BF3Plbp08sQ&feature=emb_logo)
4. Introduction to Robust (Convolutional) AE — [https://www.youtube.com/watch?v=unkue4h5f4A&feature=emb\\_logo](https://www.youtube.com/watch?v=unkue4h5f4A&feature=emb_logo)
5. Hands On: RAE implementation — [https://www.youtube.com/watch?v=rp1Z7DVt\\_ZA&feature=emb\\_logo](https://www.youtube.com/watch?v=rp1Z7DVt_ZA&feature=emb_logo)

# Часть I

## Advertising

---

2	Predicting conversions in display advertising based on URL embeddings	13
3	On the Effectiveness of Self-supervised Pre-training for Modeling User Behavior Sequences	16
4	General-Purpose User Embeddings based on Mobile App Usage	19
5	Hubble: an Industrial System for Audience Expansion in Mobile Marketing	22
6	AutoFIS: Automatic Feature Interaction Selection in Factorization Models for CTR Prediction	25
7	Другие работы	27

---

# Predicting conversions in display advertising based on URL embeddings

Reference: <http://papers.adkdd.org/2020/papers/adkdd20-qi-u-predicting.pdf>

Keywords: user behavior, embeddings, lookalike

## Какую задачу решают авторы?

Авторы решают задачу предсказания вероятности конверсии пользователя для **конкретного** рекламодателя.

Интересной особенностью данной работы является то, что в качестве данных пользователя используется только история его активности в интернете — последовательность URL'ов, которые посетил пользователь.

Решение можно разбить на две части:

- Обучение векторных представлений для URL'ов
- Обучение модели, которая используя представления для URL'ов, предсказывает вероятность конверсии

Так как две части по сути независимы друг от друга, то в работе рассматриваются разные варианты для решения каждой из подзадач, и то какие из комбинаций решений являются наиболее хорошими.

## Как решают?

Рассмотрим сначала решение каждой из подзадач по отдельности, и начнем с того как авторы предлагают получать векторные представления для URL'ов.

### URL Representation Schemes

По аналогии с задачами NLP, авторы предлагают рассматривать последовательность URL'ов как документ, предложениями в котором являются последовательности токенов из URL'ов.

Причем каждый URL состоит максимум из трех токенов, например, [https://en.wikipedia.org/wiki/Main\\_Page/Some\\_Subpage](https://en.wikipedia.org/wiki/Main_Page/Some_Subpage) разбивается на следующие токены [en.wikipedia.org, wiki, Main\_Page].

Все способы, которые рассматриваются в работе можно поделить на две группы i) one-hot encoding ii) обучение эмбеддингов

**One-hot encoding** Вектор для URL'a получается усреднением one-hot векторов токенов, на которые разбит URL.

Так как данный способ не позволяет учитывать семантическую похожесть между URL'ами и размер представления линейно зависит от числа токенов, то авторы рассматривают данный способ в качестве baseline решения.

**Embedding learning** В рамках данного способа авторы предлагают использовать подход похожий на Word2Vec (**Skip-Gram with Negative Sampling**) с той лишь разницей, что обучаются эмбединги не для URL'ов целиком, но для токенов.

В качестве данных для обучения используют истории пользовательской активности. В результате, URL'ы, которые часто оказываются рядом в сессиях находятся близко друг к другу в векторном пространстве.

Для того чтобы из токенов получить векторное представление URL'a, в работе рассматривают следующие варианты

- **Token\_concat** — конкатенация векторов токенов
- **Token\_avg** — усреднение векторов токенов
- **Domain\_only** — используется только вектор домена (первый токен)

## Conversion Prediction

Для того чтобы обучить, например, логистическую регрессию для предсказания вероятности конверсии, нужно истории пользователей превратить в фичи (вектор фиксированной длины).

Авторы статьи предлагают несколько вариантов того как это можно сделать

- Усреднение векторов эмбедингов из истории пользователя
- Однослойная нейросеть поверх усреднения векторов
- Так как предыдущие варианты теряют информацию о хронологии, то в качестве третьего варианта авторы предлагают RNN, на вход которой поступают вектора URL'ов из истории пользователя

Общая архитектура представлена на рисунке 2.1. Итоговая модель обучается на данных о конверсиях пользователей, минимизируя **Cross-Entropy Loss**.

## Experiments

В рамках экспериментов, авторы оценивают качество разных комбинаций получения векторов URL'ов и получения фичей из истории пользователя.

Качество оценивают на задаче предсказания вероятности конверсии для пяти рекламодателей. В качестве метрики для оценки качества используют **ROC AUC**.

По результатам экспериментов лучше всего себя показала модель **Token\_avg/RNN**.

## Преимущества подхода

- Довольно прост в реализации

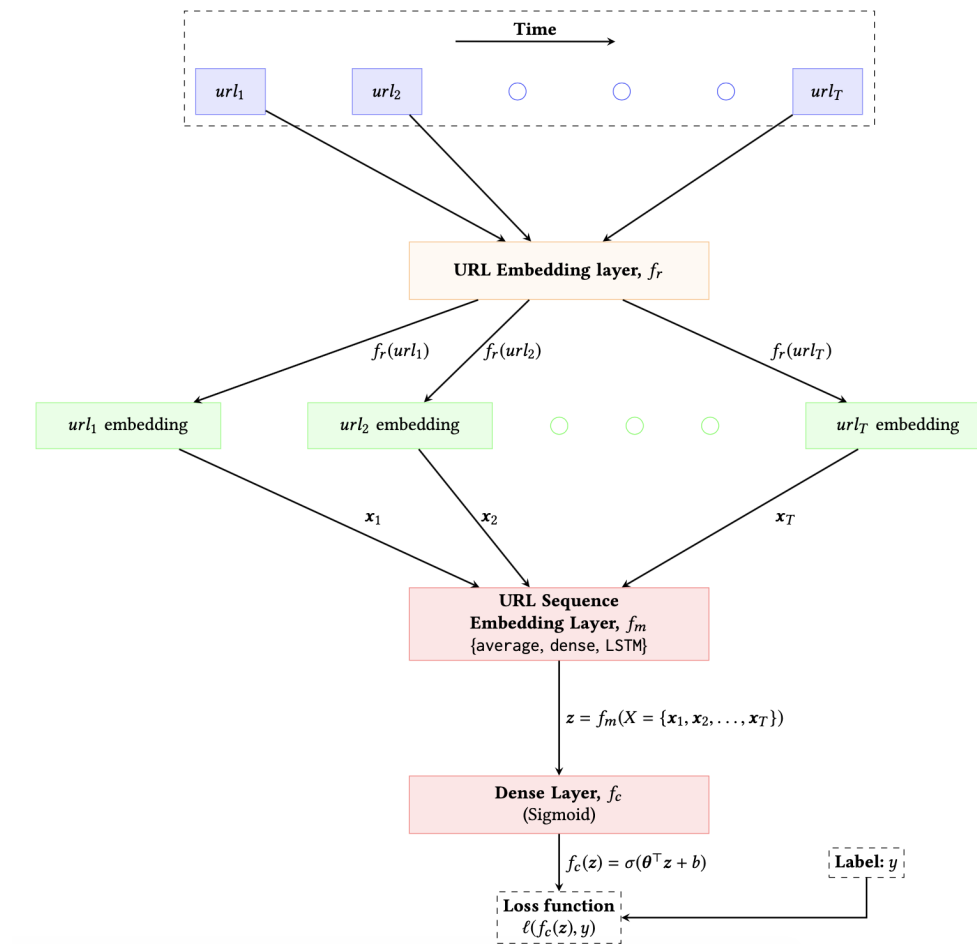


Рис. 2.1: Общая архитектура

## Мое мнение

- Хотелось бы увидеть оценку качества на большем числе датасетов. Кажется, что сравнение на пяти рекламодателях это довольно мало для надежной оценки.
- Не смотря на то, что **Token\_avg/RNN** оказался лучшим по результатам оффлайн экспериментов, по метрикам видно, что baseline **One\_hot/LR** не так уж и сильно отстает по метрикам. Интересно было бы увидеть результаты сравнения данных моделей в онлайн
- Интересно почему авторы не используют дополнительную информацию о пользователях, а ограничились историей серфинга в интернете
- То что предлагают в статье очень похоже на то как я сейчас пробую сделать Lookalike

### Key Takeaways:

1. Word2Vec все еще сила в 2020 году

# On the Effectiveness of Self-supervised Pre-training for Modeling User Behavior Sequences

Unity

Best paper award from AdKDD

Reference: <http://papers.adkdd.org/2020/papers/adkdd20-liao-effectiveness.pdf>

Keywords: user behavior, embeddings, CTR prediction

## Какую задачу решают авторы?

История активности пользователя (какие сайты посетил, на какие объявления кликал) несет в себе полезные сигналы в том числе о коммерческих интересах пользователя.

Использование данной информации может улучшить качество предсказания вероятности клика пользователя по объявлению (CTR prediction), так как позволяет своевременно реагировать на недавние действия пользователя.

Кроме того, в задаче CTR prediction'a очень важно постоянно обновлять модель [3], чтобы можно было реагировать на изменения трендов в данных.

Большинство state-of-the-art моделей на текущий момент - это довольно массивные нейросетевые архитектуры [2, 10, 7], которые довольно тяжело постоянно обновлять, так как это требует много времени и ресурсов, и лишь некоторые модели [10] используют информацию об активности пользователя.

В рамках статьи авторы предлагают нейросетевую архитектуру (см. Рисунок 3.1), состоящую из двух моделей

- **dense network** — обрабатывает категориальные и численные признаки связанные с пользователем и объявлением
- **sequential network** — обрабатывает активность пользователя

Архитектура позволяет предобучить **sequential network** в self-supervised режиме, что позволяет как быстрее переобучать модель для CTR prediction'a, так и добиваться более хороших результатов.

## Как решают?

Рассмотрим детальнее что из себя представляет каждая из сетей.



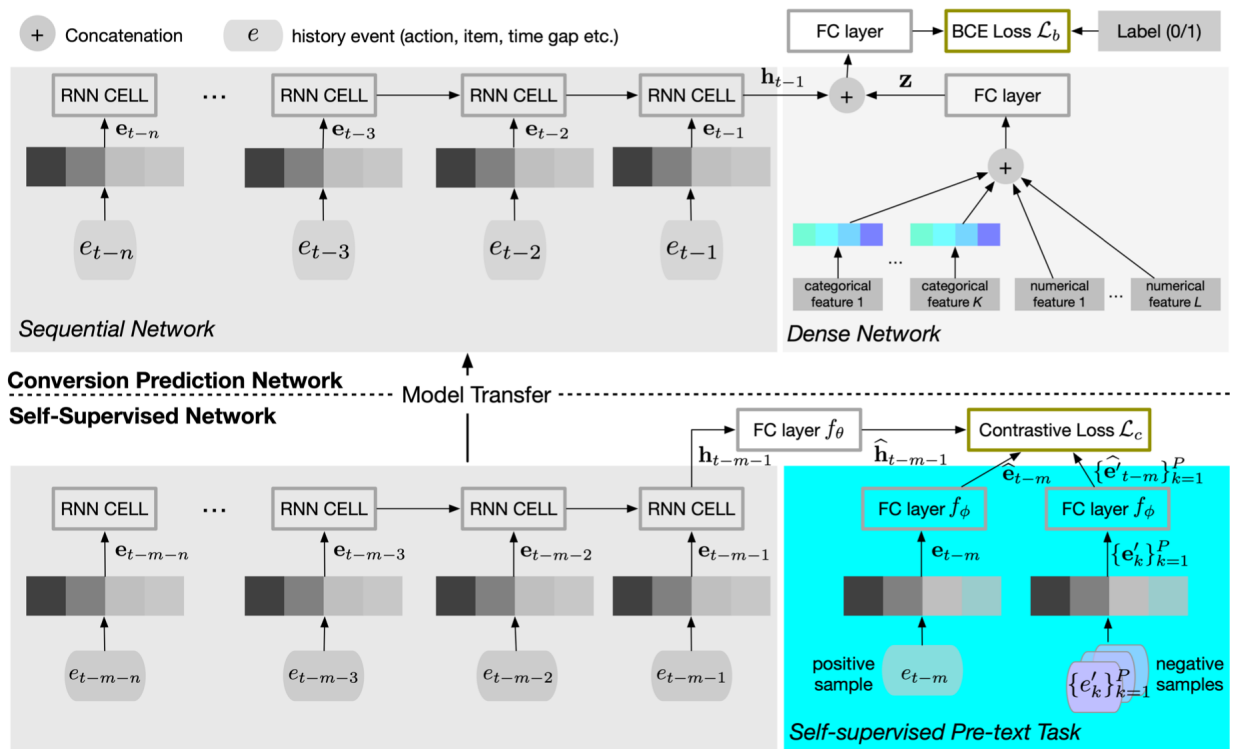


Рис. 3.1: The overview of the proposed conversion prediction and the self-supervised networks. The former consists of the sequential and the dense networks. The latter is learned by the guidance of the proposed pre-text task and it serves as the pre-trained model for the sequential network.

## Dense Network

**Dense network** имеет уже довольно привычный для задачи CTR prediction'a вид (см. Рисунок 3.1).

Для категориальных фичей обучаются эмбединги, которые конкатенируются с числовыми признаками и подаются на вход нескольким полносвязным слоям.

**Remark:** в рамках работы в **dense network** подают на вход только ранжируемое объявление.

## Sequential Network

Для того чтобы ускорить переобучение модели CTR prediction'a, авторы предлагают предобучать **sequential network** в self-supervised режиме на задаче предсказания следующего события в сессии пользователя (см. Рисунок 3.1 нижняя часть).

Важно заметить, что модель обучают не напрямую предсказывать следующее событие, а обучают различать релевантные события от нерелевантных, то есть для обучения используют **contrastive loss** (Очень похоже на разницу между CBOW и Skip-Gram w/ Negative Sampling моделями для Word2Vec).

Использование **contrastive loss** имеет следующие преимущества в сравнении с задачей много-классовой классификации

- Обучение становится существенно быстрее, когда классов очень много. На практике, если

пробовать предсказать следующий посещенный сайт, то классов могут быть миллионы. Ускорение достигается за счет того, что нет необходимости вычислять softmax для расчета итоговых вероятностей.

- Модель более устойчива к шуму в данных для обучения

## Итоговая модель

Промежуточные представления полученные с помощью **dense network** и **sequential network** конкатенируются и передаются на вход в полносвязный слой.

Модель обучается оптимизируя классический для задачи CTR-prediction'a лосс - **Binary Cross-Entropy loss**.

За счет того что **sequential network** уже была предобучена, то для обучения итоговой модели нужно как меньше данных, так и меньше времени.

## Преимущества подхода

На мой взгляд, самое важное преимущество, о котором почти не написано в статье, заключается в следующем:

Так как модель декомпозирована на **dense network** и **sequential network**, то на этапе ранжирования нет необходимости в том, чтобы запрашивать недавнюю активность пользователя и пропускать ее через **sequential network**.

Промежуточное представление получаемое с помощью **sequential network** можно предпосчитывать и обновлять для пользователя независимо всякий раз, когда мы получаем информацию о новых действиях пользователя, что случается гораздо реже чем ранжирование. И на этапе ранжирования достаточно для пользователя запросить уже посчитанное промежуточное представление.

Это существенно уменьшает время необходимое для ранжирования объявлений, так как фактически на данном этапе используется только **dense network**.

В статье [6] пытались добиться такой же декомпозиции, но итоговое решение получилось очень сложным.

## Мое мнение

- Очень хотелось бы в качестве одного из baseline'ов увидеть решение, где в качестве **sequential network** используют просто усреднение вектора объектов из активности пользователя (вектора можно было бы обучить с помощью Word2Vec).
- Также было бы интересно увидеть в качестве **sequential network** и другие архитектуры, например, Bert4Rec.
- Жаль, что совсем не сравнивают свое решение с другими архитектурами на публичных датасетах.

# General-Purpose User Embeddings based on Mobile App Usage

Tencent

Reference: [9]

Keywords: user behavior, embeddings, deep learning

## Какую задачу решают авторы?

Данные об использовании пользователем мобильных приложений несут в себе много информации о его интересах (см. Рисунок 4.1).

Список установленных приложений, установки, удаления — все это хорошие индикаторы как долгосрочных так и кратковременных интересов пользователя.

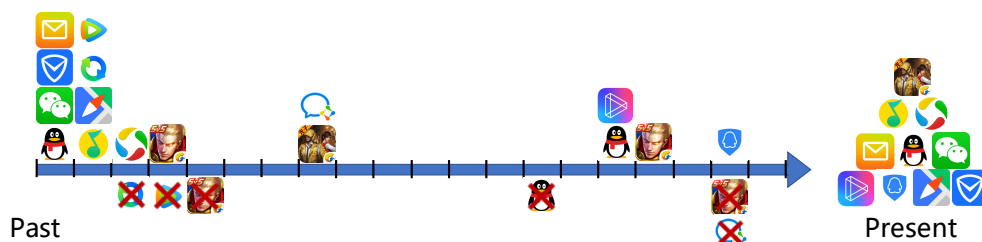


Рис. 4.1: Illustration of retention, installation, and uninstallation. Operations of (un)installation are low-frequency and unevenly distributed over time.

При решении прикладных задач, таких как рекомендации, предсказание вероятности клика (CTR prediction), поиск релевантной аудитории (lookalike), можно получить дополнительный прирост качества при использовании информации об использовании пользователем мобильных приложений.

Как правило, данную информацию добавляют в модели путем feature engineering'a, что может быть довольно сложно.

В работе представлена модель, которая по данным об использовании приложений, позволяет построить General-Purpose эмбединг для пользователя.

Полученные эмбединги можно легко использовать для решения практических задач, избегая сложного feature engineering'a.

## Как решают?

Модель описанная в работе довольно сложная, поэтому мне бы хотелось рассмотреть один из бэйслайнов, приведенных в работе, который, не смотря на свою простоту, позволяет получать очень хорошие результаты.

**Основная идея** Эмбединги для пользователя можно получить если скормить список установленных приложений в DAE (см. Рисунок 4.2).

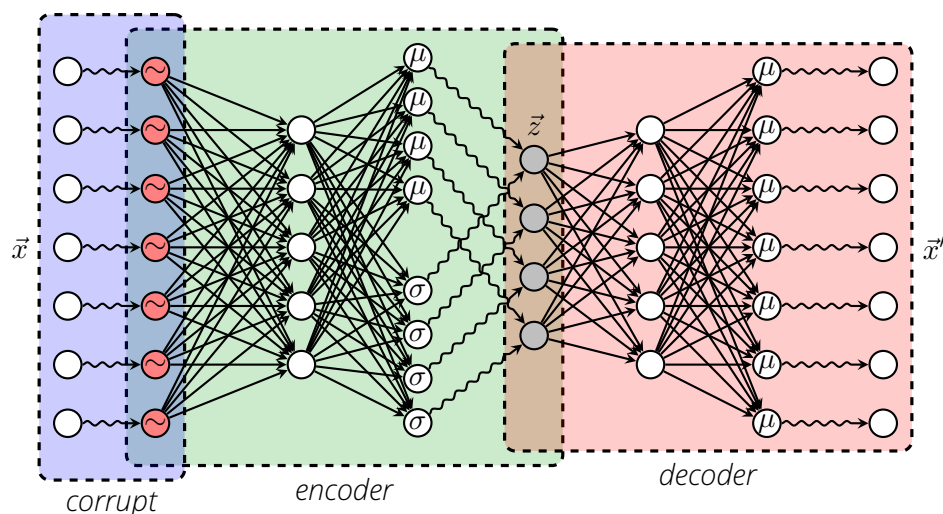


Рис. 4.2: Denoising Autoencoder (DAE)

**Data Preprocessing** Прежде обучать модель, нужно выполнить предварительную чистку данных, чтобы эмбединги пользователей лучше отражали их интересы:

- Игнорируются наиболее популярные приложения, которые есть практически на всех смартфонах, так как факт их наличия на телефоне мало что говорит об интересах пользователя
- Игнорируются предустановленные на телефон приложения
- Также исключаются совсем редкие приложения, которые установлены у маленького числа пользователей

## Experiments

В статье представлен ряд экспериментов, подтверждающих полезность полученных эмбедингов пользователей.

Рассмотрим два эксперимента

**Test #1: Next Week's Installation Prediction** В рамках первого теста, эмбединги пользователя используют для предсказания того какие приложения пользователь установит в течении следующей недели.

Для решения задачи обучают нейросеть, состоящую из нескольких полносвязных слоев, на эмбедингах пользователей.

**Test #2: Look-alike Audience Extension** В рамках второго теста, эмбединги используются для расширения аудитории.

Данные для обучения — аудитория из миллиона пользователей, примерно 10% из которых - seed (пользователи, которые используют некоторое нишевое приложение).

Задача состоит в том, чтобы научиться искать пользователей, похожих на seed.

Для этого на эмбедингах пользователей обучается классификатор (XGBoost), который предсказывает вероятность того, что пользователь относится к seed пользователям.

В качестве метрики для оценки качества использовали ROC AUC.

Не смотря на простоту подхода, использование DAE показывает результаты, которые не намного хуже в сравнении с моделью описанной в статье.

## Преимущества подхода

- Простой в реализации подход
- Не требует частого переобучения модели целиком, так как новые приложения появляются не так часто
- Позволяет практически в онлайн обновлять эмбединги для пользователей при установке/удаление приложений, что позволяет лучше реагировать на изменение интересов пользователя

# Hubble: an Industrial System for Audience Expansion in Mobile Marketing

Reference: <https://dl.acm.org/doi/pdf/10.1145/3394486.3403295>

Keywords: embeddings, lookalike, audience expansion, graphs, knowledge distillation

## Какую задачу решают авторы?

Авторы решают довольно популярную в области интернет-рекламы задачу — расширение аудитории.

Часто, у рекламодателя может быть список пользователей, которые положительно отреагировали на предыдущую рекламную кампанию (seed множество).

Для того чтобы показывать новое объявление пользователям, которые с большей вероятностью совершат целевое действие, рекламная система может помочь рекламодателю и найти пользователей, которые похожи на seed множество.

На практике при решении данной задачи разработчики сталкиваются со следующим основным челленджем:

- система должна хорошо масштабироваться на случай большого числа кампаний и пользователей, и выдавать расширенную аудиторию в течении нескольких минут

В рамках работы авторы описывают two-stage подход, где в оффлайне обучают эмбединги для пользователей и рекламных кампаний, и в онлайн обучают легковесные модели для расширения аудитории.

Основным недостатком предыдущих работ [1, 4] с two-stage подходом является недостаточно хорошее качество пользовательских эмбедингов, которые не отражают все многообразие интересов пользователя.

В работе представлен ряд оффлайн и онлайн экспериментов, где показано превосходство нового метода над существующими state-of-the-art подходами к решению задачи расширения аудитории.

## Как решают?

Общая архитектура системы Hubble представлена на Рисунке 5.1

Рассмотрим подробнее, что происходит в рамках онлайн и оффлайн фаз.



## Online Audience Expansion with Knowledge Distillation

Система получает аудиторию  $\mathcal{S}_c$ , которую нужно расширить. Если для кампании  $c$  уже есть эмбединг, то есть если она присутствует в User-Campaign графе, то для расширения аудитории достаточно найти пользователей с большим скором  $\sigma(\mathbf{h}_u, \mathbf{h}_c)$ .

На практике, часто создаются новые кампании, для которых еще нет вершины в графе. В данной ситуации обучается классификатор, который предсказывает вероятность того, что пользователь совершит целевое действие.

Для обучения модели в онлайн генерируется множество случайных пользователей  $\mathcal{U}_n \subset \mathcal{U} \setminus \mathcal{S}_c$  (см. Рисунок 5.1), и модель обучается на эмбедингах различать пользователей из  $\mathcal{S}_c$  от  $\mathcal{U}_n$ .

Для того чтобы в онлайн модель передать больше знаний от оффлайн модели AD-GNN, авторы используют *knowledge distillation*, то есть обучают онлайн модель не только на *hard* лэйблах (1 - если пользователь из  $\mathcal{S}_c$  и 0 - в противном случае), но и на *soft* лэйблах, которые получают от AD-GNN.

Soft лэйблы получают следующим образом.

1. Выполняется поиск похожих кампаний на кампанию  $c$ . Делается это с помощью разбиения пользователей из  $\mathcal{S}_c$  на  $k$  кластеров и последующим поиском наиболее похожей кампании на каждый кластер. В итоге получаем множество  $\{c_i\}_{i=1}^k$  из  $k$  кампаний.
2. Soft label для пользователя - средняя похожесть на вектора для кампаний из множества  $\{c_i\}_{i=1}^k$ .

Итоговый лосс выглядит следующим образом:

$$\begin{aligned} \mathcal{L} = & -\frac{1}{|\mathcal{D}|} \sum_{(u, y_h, y_s) \in \mathcal{D}} y_h \log(\hat{y}) + (1 - y_h) \log(1 - \hat{y}) \\ & - \frac{\gamma}{|\mathcal{D}|} \sum_{(u, y_h, y_s) \in \mathcal{D}} y_s \log(\hat{y}) + (1 - y_s) \log(1 - \hat{y}) \\ & + \frac{\lambda}{2} \|\Theta\|_2, \end{aligned}$$

где  $\hat{y} = MLP(\mathbf{h}_u)$ ,  $y_h$  - hard label,  $y_s$  - soft label и  $\gamma$  - гиперпараметр, который контролирует степень переноса знаний из AD-GNN в онлайн модель.

В рамках экспериментов авторы показывают, что при  $\gamma > 0$  улучшается качество онлайн модели, то есть модель действительно получает дополнительное полезное знание от AD-GNN.

## Experiments

Во-первых, авторы показывают, что система работает лучше текущих state-of-the-art решений для задачи расширения аудитории.

Во-вторых, демонстрируют, что предложенная архитектура сети AD-GNN работает лучше других sota решений на задаче *link-prediction*.

И наконец, показывают многократное ускорение по времени в онлайн, которое достигается в основном за счет того, что на вход модели поступают эмбединги пользователей, размерность которых существенно меньше размерности обычного описания с помощью фичей — 64 vs 300k.

## Мое мнение

Тут <https://vk.com/@papersreaders-finding-users-who-act-alike-transfer-learning-for-expanding> можно почитать конспект статьи от Pinterest с KDD'19, в которой описано решение [1].



# AutoFIS: Automatic Feature Interaction Selection in Factorization Models for CTR Prediction

Huawei

Reference: <https://arxiv.org/abs/2003.11235>

Keywords: CTR prediction, factorization machines, neural architecture search

Обзорный коспект про CTR prediction:

<https://vk.com/@papersreaders-a-sparse-deep-factorization-machine-for-efficient-ctr-predic>

## Какую задачу решают авторы?

С появлением Factorization Machines (FM) стало понятно, что добавление в линейную модель взаимодействий второго порядка между признаками существенно улучшает качество модели.

Но также известно, что среди всех возможных пар признаков полезными является лишь небольшая часть (20-50%).

Поэтому добавление в модель всех пар признаков и учет их с одинаковым весом является субоптимальным решением.

Авторы решают задачу поиска хороших пар признаков в FM-like моделях.

Это не первая попытка решить данную задачу для FM-like моделей (см. [8, 5]).

Но в отличие от предыдущих работ, предложенный подход решает эту задачу лучше.

## Как решают

Решение состоит из двух фаз: *search stage* и *re-train stage*.

В рамках *search stage* алгоритм находит хорошие пары признаков, а в рамках *re-train stage* FM-like модель обучается уже с использованием только отобранных пар.

Рассмотрим чуть подробнее каждую из фаз.

**Search stage** Модель для поиска хороших пар признаков выглядит следующим образом:

$$l_{\text{AutoFIS}} = \langle \mathbf{w}, \mathbf{x} \rangle + \sum_{i=1}^m \sum_{j>i}^m \alpha_{(i,j)} \langle \mathbf{e}_i, \mathbf{e}_j \rangle,$$

по сути обычная FM модель, с той лишь разницей, что перед скалярным произведением векторов признаков  $\mathbf{e}_i, \mathbf{e}_j$  стоит обучаемый вес  $\alpha_{(i,j)}$ .

Модель обучается на Binary Cross-Entropy Loss, но с использованием GRDA Optimizer'a, который позволяет получить sparse решение, то есть большая часть коэф  $\alpha$  будет равна нулю.

Соответственно  $\alpha_{(i,j)}$  принимает не нулевое значение для важных пар признаков.

**Re-train stage** После того как были отобраны пары признаков, модель дообучается (тунятся как веса модели, так и коэффициенты  $\alpha$ , но только для отобранных признаков).

## Преимущества подхода

Решение является довольно робастным, то есть запуск search stage с разным seed'ом приводит к похожим результатам.

Найденные пары признаков можно легко использовать при обучении любых других FM-like моделей.

## Мое мнение

Интересно, что на сайте <https://paperswithcode.com/sota/click-through-rate-prediction-on-criteo> AutoFIS не показывает sota результаты.

Более того, незадолго до начала конференции KDD, авторы из Yahoo Research опубликовали работу DeepLight <https://arxiv.org/abs/2002.06987>, которая показывает sota результаты на датасете Criteo.

## Другие работы

Пара работ про совместное ранжирование органического контента и рекламы.

### Jointly Learning to Recommend and Advertise

Reference: <https://arxiv.org/abs/2003.00097>

### Ads Allocation in Feed via Constrained Optimization

LinkedIn

Reference: <https://dl.acm.org/doi/pdf/10.1145/3394486.3403391>

# Литература

- [1] Stephanie deWet and Jiafan Ou. Finding users who act alike: Transfer learning for expanding advertiser audiences. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2251–2259, 2019.
- [2] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.
- [3] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9, 2014.
- [4] Yudan Liu, Kaikai Ge, Xu Zhang, and Leyu Lin. Real-time attention based look-alike model for recommender system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2765–2773, 2019.
- [5] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. Field-weighted factorization machines for click-through rate prediction in display advertising. In *Proceedings of the 2018 World Wide Web Conference*, pages 1349–1357, 2018.
- [6] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. Practice on long sequential user behavior modeling for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2671–2679, 2019.
- [7] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD’17*, pages 1–7. 2017.
- [8] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*, 2017.
- [9] Junqi Zhang, Bing Bai, Ye Lin, Jian Liang, Kun Bai, and Fei Wang. General-purpose user embeddings based on mobile app usage. *arXiv preprint arXiv:2005.13303*, 2020. URL <https://arxiv.org/pdf/2005.13303.pdf>.
- [10] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1059–1068, 2018.

## Часть II

# Recommender Systems

---

8	On Sampled Metrics for Item Recommendation	30
9	Temporal-Contextual Recommendation in Real-Time	33
10	SimClusters: Community-Based Representations for Heterogeneous Recs at Twitter	36
11	Learning to Cluster Documents into Workspaces Using Large Scale Activity Logs	37
12	Другие работы	40

---

# On Sampled Metrics for Item Recommendation

Google Research

Best research paper award

Reference: <http://walid.krichene.net/papers/KDD-sampled-metrics.pdf>

Keywords: ranking evaluation, metrics, sampling

## Какую задачу решают авторы?

Типичный протокол оценки качества рекомендательных систем выглядит следующим образом:

1. Для отобранного множества пользователей  $D$  ранжируем алгоритмом  $A$  все множество кандидатов, состоящее из  $n$  объектов
2. Для каждого пользователя  $\mathbf{x}$  вычисляем  $R(A, \mathbf{x})$  — множество позиций релевантных объектов
3. После чего для пользователя считаем метрику  $M$ , например, **ROC AUC** или **Precision@K, Recall@K**

Итоговое значение метрики получается усреднением метрик посчитанных по всем пользователям

$$\frac{1}{|D|} \sum_{\mathbf{x} \in D} M(R(A, \mathbf{x})).$$

В ситуации когда  $n$  велико часто прибегают к сэмплингованию — вместо того чтобы ранжировать все  $n$  кандидатов, ранжируют случайную подвыборку из  $m$  объектов ( $m \ll n$ ) вместе с релевантными для пользователя объектами.

Ожидается, что метрики посчитанные с сэмплингованием позволяют упорядочить алгоритмы ранжирования по качеству так же как и метрики посчитанные без сэмплингования.

Авторы статьи впервые тестируют это предположение и показывают что для большинства используемых метрик оно не верно, даже при многократном сэмплинговании и усреднении результатов.

В статье предложены скорректированные варианты привычных метрик, которые позволяют при использовании сэмплингования сортировать алгоритмы по качеству также как если бы сэмплингования не было.

## Как решают?

**Remark:** для простоты можно считать, что для каждого пользователя есть один релевантный объект.

Работу можно разбить на три части

1. Экспериментальная часть, где проверяют предположение, что метрики с сэмплением и без упорядочивают алгоритмы ранжирования одинаковым образом
2. Теоретическая часть, посвященная скорректированным вариантам метрик
3. Экспериментальная часть, где авторы показывают, что скорректированные варианты метрик работают

## Inconsistency of Sampled Metrics

Ключевое определение данной работы

**Definition 1** (Consistency): *Let the evaluation data  $D$  be fixed. A metric  $M$  is consistent under sampling if the relative order of any two recommenders  $A$  and  $B$  is preserved in expectation. That is, for all  $A, B$ ,*

$$\begin{aligned} \frac{1}{|D|} \sum_{\mathbf{x} \in D} M(R(A, \mathbf{x})) &> \frac{1}{|D|} \sum_{\mathbf{x} \in D} M(R(B, \mathbf{x})) \\ \iff \mathbb{E} \left[ \frac{1}{|D|} \sum_{\mathbf{x} \in D} M(\tilde{R}(A, \mathbf{x})) \right] &> \mathbb{E} \left[ \frac{1}{|D|} \sum_{\mathbf{x} \in D} M(\tilde{R}(B, \mathbf{x})) \right] \end{aligned}$$

В рамках первой группы экспериментов (см. Рисунок 8.1) показывают как меняются метрики для разных алгоритмов в зависимости от  $m$  (число случайно отобранных объектов).

Из экспериментов видно, что единственная консистентная метрика — **ROC AUC**.

## Corrected Metrics

Очень рекомендую самостоятельно ознакомиться с данной частью работы, так как ее не очень удобно изложить в виде конспекта.

## Выводы

### Key Takeaways:

1. Нужно избегать сэмпления при расчете метрик
2. Если нет возможности избежать сэмпления, то нужно использовать скорректированные версии метрик
3. Вычисление метрик нужно повторять несколько раз с разным seed'ом для того чтобы уменьшить дисперсию

Все это в лишний раз наводит на мысли о том, что в ряде работ лучшими оказались алгоритмы, которые не обязательно являются лучшими на самом деле, и на результаты экспериментов в статьях всегда надо смотреть с определенной долей скепсиса.

Еще одна работа примерно про тоже On Sampling Top-K Recommendation Evaluation <https://dl.acm.org/doi/pdf/10.1145/3394486.3403262> с этой же конференции.

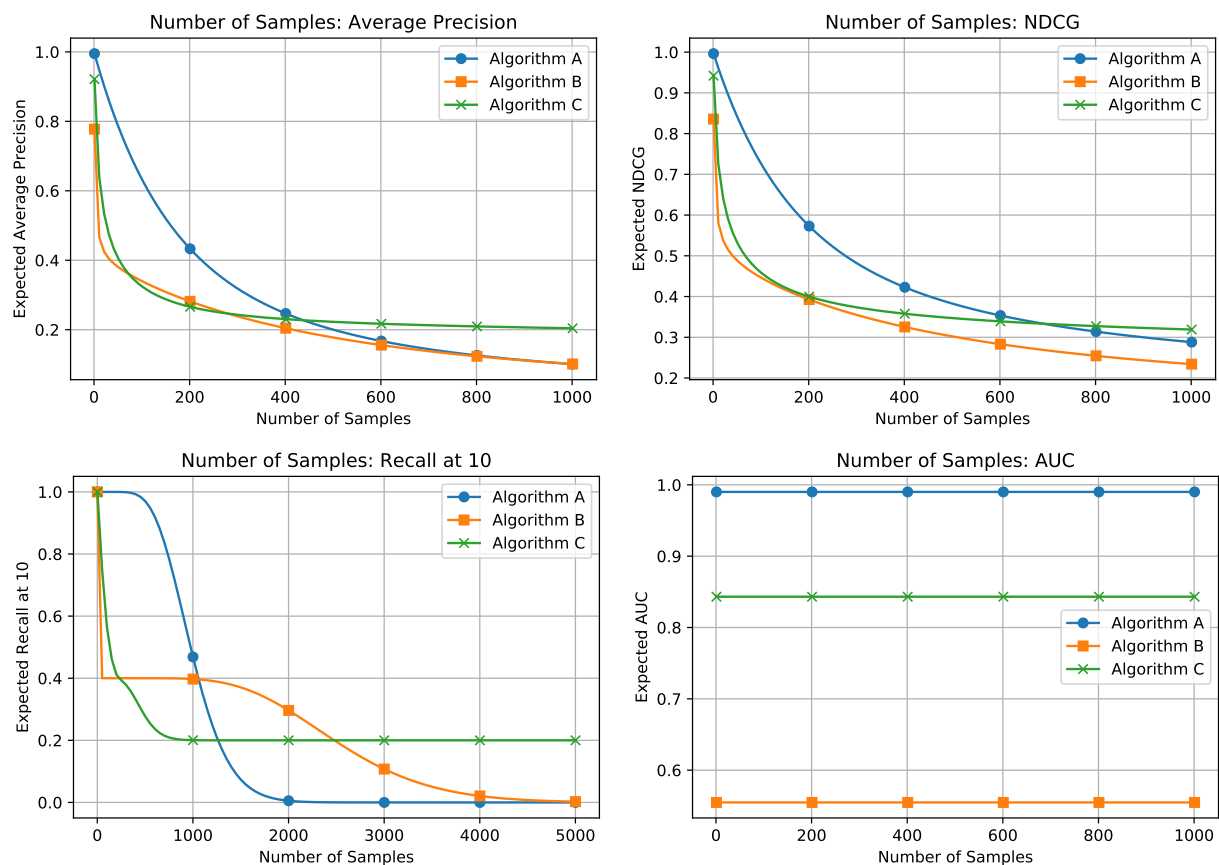


Рис. 8.1: Expected sampling metrics for the running example while increasing the number of samples. For Average Precision, NDCG and Recall, even the relative order of algorithm performance changes with the number of samples. That means, conclusions drawn from a subsample are not consistent with the true performance of the algorithm.



# Temporal-Contextual Recommendation in Real-Time

Amazon

Best ADS paper

Reference: <https://dl.acm.org/doi/pdf/10.1145/3394486.3403278>

Keywords: recommender systems, recurrent neural networks, hybrid model

## Какую задачу решают авторы?

Разработчики современных рекомендательных систем сталкиваются со следующими челенджами: Система должна

- Оперативно реагировать на изменение интересов пользователя
- Обучаться на историях большого числа пользователей, состоящих из сотен событий, за разумное время
- Быть эффективной для новых пользователей и объектов (cold-start problem)
- Хорошо масштабироваться на случай ранжирования большого числа объектов

Многие популярные решения не удовлетворяют всем указанным требованиям.

Например, решения, в основе которых лежит факторизация матрицы **Users-Items**, не позволяют оперативно реагировать на изменение интересов пользователя, и не могут строить рекомендации для новых пользователей/объектов.

В статье авторы предлагают решение, которое удовлетворяет всем перечисленным требованиям.

В основе решения — **RNN-like** архитектура, которая наряду с идентификаторами пользователя и объекта использует доступную мета-информацию (признаки пользователя и объекта).

Для того чтобы модель можно было за разумное время обучить на большом каталоге объектов, авторы предлагают использовать **Negative Sampling** [3] (NS) вместо классической многоклассовой классификации.

## Как решают?

Не смотря на то, что в статье описано много интересного, я хотел бы уделить внимание только одному моменту в работе — использование NS для ускорения обучения.

## Negative Sampling

Сначала ответим на вопрос почему обучение моделей рекомендаций при большом размере каталога товаров может быть медленным.

Рассмотрим следующую постановку задачи рекомендаций: пусть у нас есть история пользователя  $\{i_1, \dots, i_k\}$ , например, какие товары он уже купил, наша задача научиться предсказывать следующий товар  $i_{k+1}$ , который заинтересует пользователя.

Довольно часто задачу в такой постановке решают сводя ее к задаче *многоклассовой классификации*, где классам соответствуют товары из каталога.

Детальнее, пусть каталог состоит из  $m$  товаров,

1. Используя модель, мы получаем скрытое представление  $\mathbf{h}_k := \mathbf{h}(i_1, \dots, i_k) \in \mathbb{R}^d$ , описывающее историю пользователя
2. Полученное представление  $\mathbf{h}_k$  домножаем на матрицу  $W \in \mathbb{R}^{m \times d}$ , чтобы получить скор для каждого товара в каталоге
3. Итоговые вероятности для товаров

$$\hat{y} := \text{softmax}(W \cdot \mathbf{h}_k)$$

Модель  $\mathbf{h}$  и матрица  $W$  обучаются оптимизируя **Cross Entropy Loss**.

Домножение вектора  $\mathbf{h}_k$  на матрицу  $W$  с последующим softmax'ом занимает много времени, кроме того, матрица  $W$  занимает очень много места.

При использовании NS [3] нам не нужно вычислять скоры для всех товаров, а только для релевантного товара и для небольшого случайного множества негативных примеров, так как решаем задачу *бинарной классификации* — учимся отличать релевантные товары от нерелевантных. Это позволяет не хранить матрицу  $W$  и существенно ускоряет процесс обучения.

## Experiments

В рамках экспериментов авторы исследуют то как зависит скорость обучения (#items/sec) при использовании NS от размера каталога.

Авторы замерили скорость обучения и метрики на датасетах MovieLens (1m, 10m, 20m) и на датасете Taobao, при использовании Dense output слоя и при использовании NS.

На датасетах MovieLens модель обучали в течении четырех эпох. На датасете Taobao время на обучение было ограничено шестью часами.

Исходя из результатов (см. таблицу 9.1), можно сделать вывод, что использование NS существенно ускоряет обучение в случае большого каталога объектов (Taobao).

Кроме того, при ограничениях на время обучения, использование NS позволяет добиться лучшего качества за счет того, что модель успевает обработать большее число объектов.

	Output	ml-1m	ml-10m	ml-20m	Taobao
Output size ( $m$ )	IS	62	255	362	1087
	Dense	1683	65134	131263	1183451
Throughput (#items/sec)	NS	<u>23k</u>	20k	<b>20k</b>	<b>7.8k</b>
	Dense	23k	23k	17k	631
PPL	NS	<u>377</u>	<u>405</u>	<u>455</u>	<b>17.6k</b>
	Dense	409	439	494	119k
NDCG	NS	0.128	0.123	0.12	<b>0.15</b>
	Dense	0.123	0.119	0.115	0.08

Рис. 9.1: Model throughput and accuracy. Bold-fonts/underlines show significant/insignificant differences, respectively. NS significantly improved model training for up to 1M unique items (Taobao dataset). Additionally, the dense model for the Taobao dataset requires >16GB memory for BPTT, which is infeasible on many GPUs.

Наряду с привычными метриками для данной задачи, в статье используют еще и **Perplexity**.

**Remark:** A recommendation system with a PPL of  $p$  is equivalent to one that recommends a uniform random selection of  $p$  items, one of which is the true next item.

## Выводы

### Key Takeaways:

1. Если каталог объектов очень большой ( $\geq 10^5$ ), то для обучения модели ранжировани лучше использовать **Negative Sampling**. Это позволяет не только ускорить обучение, но и существенно уменьшить размер модели.

# SimClusters: Community-Based Representations for Heterogeneous Recs at Twitter

Twitter

Reference: <https://dl.acm.org/doi/pdf/10.1145/3394486.3403370>

Keywords: user behavior, nearest neighbours, clustering

## Какую задачу решают авторы?

## Как решают?

### Stage 1: Community Discovery

Similarity Graph of Right Nodes

Communities of Right Nodes

Communities of Left Nodes

### Stage 2: Item Representations

### Applications

Предложенная система работает в Twitter уже около года и за это время нашла себе применение в разных продуктах компании.

В рамках каждого из продуктов, новая система показала ощутимый рост метрик в сравнении с существующими методами.

## Преимущества подхода

**Key Takeaway:** In our experience of building recommendations at Twitter, we find that neighborhood-based methods are easier to scale, more accurate, more interpretable, and also more flexible in terms of accommodating new users and/or items

# Learning to Cluster Documents into Workspaces Using Large Scale Activity Logs

Google

Reference: <https://dl.acm.org/doi/abs/10.1145/3394486.3403291> [2]

Keywords: user behavior, embeddings, clustering

## Какую задачу решают авторы?

Авторы решают задачу кластеризации документов в Google Drive в workspace'ы — отдельные папки, содержащие документы похожие не только по смыслу, но и связанные с конкретными задачами пользователя.

У классических подходов к кластеризации документов есть недостатки связанные с тем, что они могут группировать вместе документы одной тематики, но при этом они не учитывают большое количество других факторов.

Например, если пользователь работает над конкретной задачей, то он вполне может использовать документы с разной тематикой.

В рамках данной работы, авторы не решают задачу кластеризации напрямую, но обучают document similarity модель, которая для двух документов предсказывает относятся они к одному кластеру или нет.

Используя предсказания модели, авторы используют иерархическую кластеризацию для группировки документов в workspace'ы.

*Главный вопрос:* где взять данные для обучения document similarity модели?

На практике, получение датасета достаточного размера, состоящего из пар документов с разметкой от пользователей, может быть довольно сложно.

Авторы предлагают способ получить разметку для обучения напрямую из логов активности пользователей — сводят задачу к weakly supervised варианту.

Основное предположение, которое позволяет получить weak разметку — если пользователь выполнял действия с документами с небольшой разницей во времени (*co-access*), то скорее всего эти документы относятся к одной задаче, над которой работает пользователь.

Авторы в онлайн сравнивают свое решение с классическими unsupervised подходами для кластеризации документов, и показывают, что предлагаемое решение существенно лучше бэйслайнов.

## Как решают?

Рассмотрим сначала подробнее то как авторы предлагают получать weak разметку, а затем на предлагаемую в статье document similarity модель.

### Activity-Based Labels and Weak Supervision

Для получения разметки, авторы отталкиваются от предположения о похожести документов, с которыми пользователь работал в течении небольшого промежутка времени.

Два документа считаются co-accessed, и получают  $label = 1$ , если были открыты с разницей во времени не больше двух минут.

На рисунке 11.1 показано на каких данных обучается document similarity модель.

По сути, вся активность пользователя делиться на две части: первая используется для построения признаков и обучения document similarity модели, а вторая — для получения разметки.

Документы считаются похожими, если они были co-accessed в рамках второй части активности пользователя.

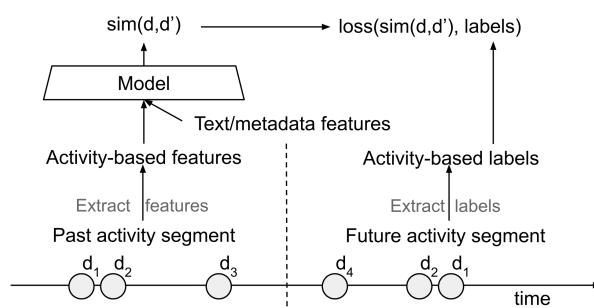


Рис. 11.1: Past and future activity segments used for extracting activity-based features and labels respectively

### Document Similarity Model

Рассмотрим подробнее как выглядит модель (см. Рисунок 11.2).

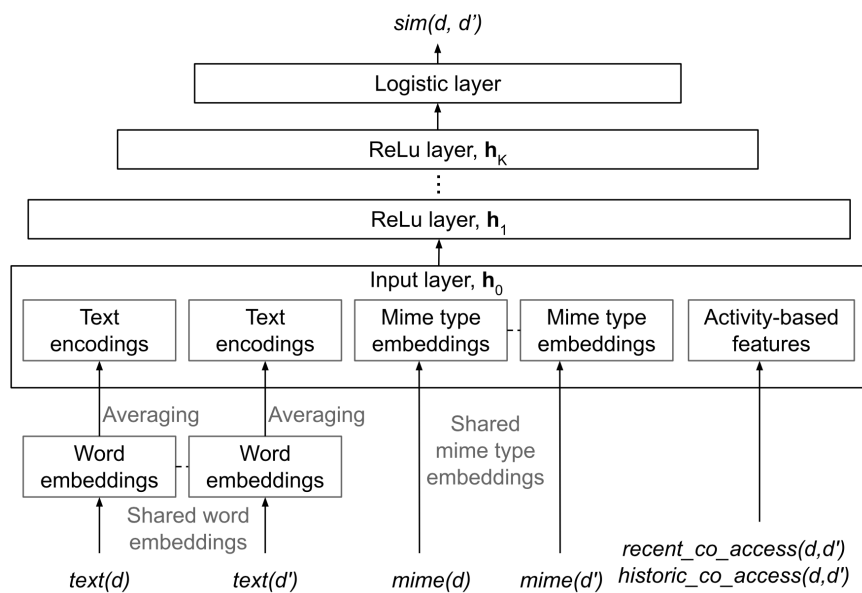


Рис. 11.2: Document Similarity Model

Список признаков, на которых обучается модель:

- **Text features**
  - $text(d)$  - text content for document  $d$
  - $text(d')$  - text content for document  $d'$
- **Metadata features**
  - $mime(d)$  - MIME type of the document  $d$
  - $mime(d')$  - MIME type of the document  $d'$
- **Activity-based features**
  - $recent\_co\_accesses(d, d')$  - number of co-accesses between  $d, d'$  in the past 2 weeks
  - $historic\_co\_accesses(d, d')$  - number of co-accesses between  $d, d'$  in the past 4 weeks
- **Human and activity-based labels**
  - $co\_cluster(d, d')$  - human labels on whether  $d, d'$  should be clustered together in a workspace
  - $future\_co\_accesses(d, d')$  - number of co-accesses between  $d, d'$  in the future week

Авторы показывают, что дополнительное использование Activity-based признаков существенно улучшает качество модели, то есть полезно использовать не только признаки связанные с документами, но и информацию о том как пользователь с ними взаимодействовал.

DSM обучается минимизируя, привычный для задачи бинарной классификации, Binary Cross-Entropy Loss.

## Experiments

Авторы утверждают, что documents similarity модель обученная на weak разметке, по качеству не уступает модели, обученной на разметке от пользователей.

Кроме того, авторы проводят онлайн эксперимент и оценивают качество предложенного алгоритма кластеризации со следующими бэйслайнами:

- Topicality - для документов обучаются эмбединги, которые потом кластеризуются
- Calendar - относит в один кластер документы, прикрепленные к одному событию в календаре
- Favorites - группирует вместе документы, которые пользователь открывает наиболее часто

По результатам экспериментов, предложенный подход работает намного лучше бэйслайнов.

## Мое мнение

Красивый и простой способ извлечения weak лэйблов, для обучения модели похожести документов, из большого количества логов активности пользователя.

## Другие работы

### Embedding-based Retrieval in Facebook Search

Facebook

Reference: <https://arxiv.org/abs/2006.11632>

Конспект: <https://vk.com/@papersreaders-embedding-based-retrieval-in-facebook-search>

Исторически поиск в Facebook работал на основе Boolean matching model.

В статье авторы делятся опытом перехода к использованию embedding-based системы на этапе отбора кандидатов перед ранжированием.

В основе предлагаемого решения модель построения эмбедингов запросов и документов.

Работа наглядно показывает, что даже относительно несложное решение для Embedding-based retrieval показывает существенное улучшение в сравнении с классическим Boolean matching подходом в задаче отбора кандидатов.

### Controllable Multi-Interest Framework for Recommendation

Alibaba

Reference: <https://arxiv.org/abs/2005.09347>

Конспект: <https://vk.com/@papersreaders-controllable-multi-interest-framework-for-recommendation>

Современные рекомендательные системы используют историю пользователя для построения вектора, который описывает пользователя и используется для поиска объектов-кандидатов при построении рекомендаций.

Однако использование единственного вектора для описания пользователя не позволяет уловить его разнообразные интересы.

Авторы статьи предлагают решение, которое позволяет представить пользователя в виде набора из  $K$  векторов, каждый из которых соответствует некоторому интересу пользователя.

Данный подход позволяет делать рекомендации как более точными так и более разнообразными в сравнении с существующими state-of-the-art подходами.



# PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest

Pinterest

Reference: <https://arxiv.org/pdf/2007.03634.pdf>

Статья от Pinterest про прокачку системы рекомендаций пинов для пользователя.

Авторы статьи рассматривают проблемы связанные с представлением пользователя в виде единственного вектора.

Для решения проблем, в статье предлагают представить пользователя в виде набора векторов.

Ключевые отличия от предыдущих работ, предлагающих сделать тоже самое:

1. количество векторов для пользователя не фиксировано
2. вектора для пользователей не обучаются совместно с векторами для пинов

Для того чтобы представить пользователя в виде набора векторов, предлагают делать иерархическую кластеризацию активности пользователя за последнее время (вектора пинов получены black-box моделью).

Каждому кластеру ставят в соответствие его важность.

Для рекомендации релевантных пинов берут 3 наиболее важных кластера и ищут похожие пины с помощью приближенного поиска ближ соседей.

Как и в большинстве последних статей от Pinterest, авторы рассматривают продакшн решение, поэтому достаточно внимания уделяют вопросу о том как все это тащить в прод.

## Improving Recommendation Quality in Google Drive

Google

Reference: <https://research.google/pubs/pub49272/>

Статья про попытки прокачать качество инструмента Quick Access (статья не научная, а просто про опыт и проведенные эксперименты)

В целом, то о чем они пишут очень похоже на наш опыт и на наши эксперименты за последние года полтора.

Эксперименты описанные в статье, которые похожие на наши

- автоматизация пайплайнов переобучения моделей
- попытки затащить DL модели (в статье наоборот пытаются перейти к GBDT)
- изучение того как влияет latency на метрики
- несколько примеров feature engineering'a
- эксперимент с увеличением числа кандидатов для ранжирования
- фича мониторинг

## Managing Diversity in Airbnb Search

Reference: [1]

С продуктовой точки зрения, показ пользователю разнообразных (diverse) рекомендаций ведет к улучшению пользовательского опыта.

Однако, современные модели ранжирования обучаются, оптимизируя relevance, что приводит к однообразным результатам в рекомендациях.

В статье авторы численно проверяют предположение о том, что рекомендации довольно часто бывают однообразными, и предлагают метрику Mean Listing Relevance (MLR), которая позволяет оценить как релевантность объектов в выдаче, так и их разнообразие.

В основе предложенной метрики — вычисление расстояний между объектами.

В работе предложено большое количество способов как можно вычислить расстояние между двумя объектами и то как оптимизировать MLR.

В заключении, авторы приводят результаты оффлайн и онлайн экспериментов где показывают рост как привычных для задачи ранжирования метрик, так и метрик связанных с разнообразием.

## Disentangled Self-Supervision in Sequential Recommenders

Alibaba

Reference: <http://pengcui.thumediablab.com/papers/DisentangledSequentialRecommendation.pdf>

Keywords: sequence model, contrastive learning

Новое sota решение в задаче sequential recommendations. Бьет BERT4Rec (на некоторых датасетах очень сильно).

Судя по оптимизируемому лоссу, обучаться модель должна быстрее чем BERT4Rec.

Жаль, что авторы не выложили исходный код.

# Литература

- [1] Mustafa Abdool, Malay Haldar, Prashant Ramanathan, Tyler Sax, Lanbo Zhang, Aamir Mansawala, Shulin Yang, and Thomas Legrand. Managing diversity in airbnb search. *arXiv preprint arXiv:2004.02621*, 2020. URL <https://arxiv.org/pdf/2004.02621.pdf>.
- [2] Weize Kong, Michael Bendersky, Marc Najork, Brandon Vargo, and Mike Colagrosso. Learning to cluster documents into workspaces using large scale activity logs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2416–2424, 2020. URL <https://dl.acm.org/doi/abs/10.1145/3394486.3403291>.
- [3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

## Часть III

# Deep Learning

---

13 Retrospective Loss: Looking Back to Improve Training of Deep Neural Networks	45
14 Correlation Networks for Extreme Multi-label Text Classification	47
15 Другие работы	49

---

# Retrospective Loss: Looking Back to Improve Training of Deep Neural Networks

Adobe

Reference: [2]

Keywords: neural networks, loss functions

## Какую задачу решают авторы?

В статье демонстрируется простой способ улучшить качество практически любой нейросетевой архитектуры.

Улучшение достигается путем добавления к функции потерь дополнительного слагаемого (retrospective loss).

При оптимизации Retrospective loss вместе с основной функцией потерь, учитываются предыдущие состояния параметров модели таким образом, чтобы как можно быстрее двигаться в направлении к оптимальным параметрам.

В статье изложена мотивация стоящая за retrospective loss, теоретическое обоснование того почему ее использование дает преимущества, и большое количество экспериментов на разных задачах, демонстрирующих улучшение качества моделей.

## Как решают?

Рассмотрим сначала как выглядит retrospective loss

**Definition 2** (Retrospective Loss): *Given network  $g: \mathbb{R}^n \rightarrow \mathbb{R}^d$  parametrized by its weights  $\theta$  an input data-label pair  $(\mathbf{x}_i, y_i)$ , the retrospective loss at time step  $T$  is given by:*

$$\mathcal{L}_{retrospective} = (\kappa + 1) \|g_{\theta^T}(\mathbf{x}_i) - y_i\| - \kappa \|g_{\theta^T}(\mathbf{x}_i) - g_{\theta^{T_p}}(\mathbf{x}_i)\|$$

*where  $\kappa$  - scaling co-efficients,  $\theta^T$  - parameters at time step  $T$  and  $T_p$  - previous time step.*

**Интуиция** Минимизируя  $\mathcal{L}_{retrospective}$  мы пытаемся сделать так, чтобы на очередном шаге предсказание модели для  $\mathbf{x}_i$  было ближе к  $y_i$ , чем к предыдущему предсказанию.

Рисунок 13.1 визуализирует данную интуицию.

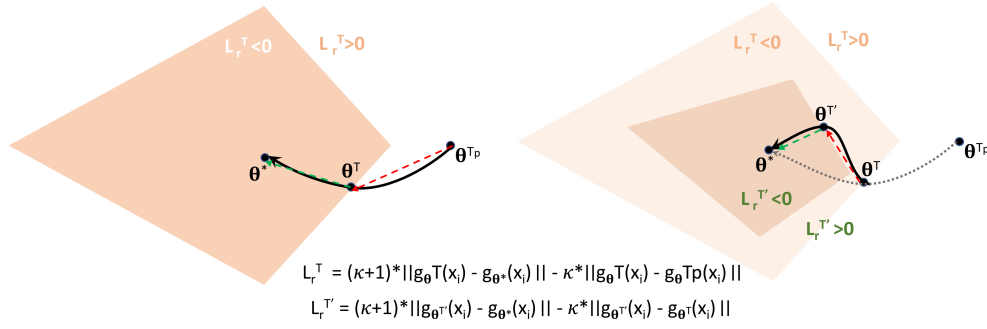


Рис. 13.1: Geometric intuition of the working of the proposed retrospective loss.

По определению,  $\mathcal{L}_{retrospective} < 0$  в ситуации, когда предсказание в текущем состоянии,  $g_{\theta^T}$ , ближе к  $y_i := g_{\theta^*}$ , чем к предыдущему предсказанию  $g_{\theta^{Tp}}$ .

В результате, пространство параметров делиться на две части - первая где  $\mathcal{L}_{retrospective} < 0$ , и вторая где  $\mathcal{L}_{retrospective} \geq 0$ .

Minimizing retrospective loss pushes the network towards parameters further inside the polytope, thus helping speed up the training process.

Новый loss - просто сумма  $\mathcal{L}_{task}$  и  $\mathcal{L}_{retrospective}$ .

**Анализ** Главный вопрос: почему добавление  $\mathcal{L}_{retrospective}$  позволяет ускорить обучение?

Для ответа на этот вопрос авторы вводят следующее определение

**Definition 3** (Consistency of Loss Terms for Neural Network Models): Пусть  $g_{\theta}$  нейросеть с параметрами  $\theta$  и  $\theta^* := \arg \min_{\theta} \mathcal{L}_{task}$  - решение задачи.

Тогда если при добавлении к  $\mathcal{L}_{task}$  функции  $\mathcal{L}_{add}$  с целью ускорить обучение, оптимум достигается в той же точке, то есть

$$\theta^* = \arg \min_{\theta} (\mathcal{L}_{task} + \mathcal{L}_{add}),$$

то функции  $\mathcal{L}_{add}$  и  $\mathcal{L}_{task}$  называются консистентными.

Авторы показывают, что при добавление  $\mathcal{L}_{retrospective}$  к основному лоссу, минимум функции потерь достигается в той же точке что и раньше, то есть  $\mathcal{L}_{retrospective}$  консистентна с любой  $\mathcal{L}_{task}$ .

На практике это означает, что при фиксированном числе итераций, использование retrospective loss позволяет ближе подойти к оптимальному решению, в результате мы наблюдаем улучшение метрик.

## Преимущества подхода

- Метод очень прост в реализации и практически всегда дает улучшение

# Correlation Networks for Extreme Multi-label Text Classification

Reference: [3]

Keywords: multi-label text classification, deep learning, label correlation

Code: <https://github.com/XunGuangxu/CorNet>

## Какую задачу решают авторы?

Задача extreme multi-label text classification (XMTC) состоит в том, чтобы сопоставить тексту набор наиболее релевантных лэйблов.

Пример появления данной задачи на практике: по описанию товара в интернет магазине, найти наиболее подходящие категории, к которым относится товар.

Как и во многих других NLP задачах, Deep Learning подходы дают state-of-the-art результаты.

Как правило, XMTC модели состоят из двух компонент: первая - извлекает информацию из текста и строит некоторое представление, вторая - использует полученное представление для предсказания лэйблов.

Существует большое количество разных подходов к построению первой компоненты, но вторая компонента, чаще всего, представляет из себя полносвязный слой (FC).

Использование простого FC слоя не позволяет учитывать полезные корреляции между лэйблами. Например, если мы уверены в том, что товару соответствуют лэйблы 'Blues' и 'R&B', то скорее всего также товару должен соответствовать лэйбл 'Music'.

Знание о корреляции между лэйблами позволяет лучше решать XMTC задачу.

В рамках статьи авторы предлагают Correlation Networks (CorNet) архитектуру, которая выучивает корреляции между лэйблами.

CorNet блок можно добавить в любую XMTC модель поверх label-prediction слоя, для улучшения качества основной модели, без существенного роста размера модели и времени обучения.

## Как решают?

**Интуиция** Пусть у нас есть предсказания лэйблов основной модели -  $\mathbf{x}$ . Самый простой способ выучить корреляции между лэйблами - добавить еще один FC слой, то есть  $\mathbf{y} = W\mathbf{x}$ .

Тогда каждый из выходов  $y_i$  нового FC слоя представляет из себя линейную комбинацию лэйблов  $\{x_1, x_2, \dots, x_i, \dots\}$ .

Но использование FC слоя очень непрактично с точки зрения потребления памяти, так как матрица  $W$  должна иметь размеры  $V \times V$ , где  $V$  - число лэйблов. Хранение такой матрицы чаще всего невозможно.

Кроме того, матрица  $W$  должна будет состоять по большей части из нулей, так как большинство лэйблов не скоррелированы.

Исходя из этих соображений, авторы предлагают следующий способ выучить корреляции лэйблов:

$$F(\mathbf{x}) = W_2 \delta(W_1 \sigma(\mathbf{x}) + \mathbf{b}_1) + \mathbf{b}_2,$$

где  $\sigma$  - sigmoid,  $\delta$  - ELU function и  $W_2 \in \mathbb{R}^{V \times R}$ ,  $W_1 \in \mathbb{R}^{R \times V}$ ,  $R \ll V$ .

**CorNet** Итоговая модель CorNet имеет вид  $\mathbf{y} = F(\mathbf{x}) + \mathbf{x}$ .

Так как матрицы  $W_2$ ,  $W_1$  имеют небольшой размер, то использование CorNet блока не сильно влияет на размер модели.

Использование CorNet в общем случае показано на рисунке 14.1.

То есть можно добавлять несколько CorNet слоев подряд для того чтобы выучивать более сложные корреляции между признаками.

**Experiments** Авторы проводят большое количество экспериментов с разными ХМТС моделями на разных датасетах, где сравнивают качество ХМТС модели с и без использования CorNet блока.

Из результатов экспериментов авторы делают выводы, что использование CorNet блока: практически всегда позволяет значительно улучшить качество ХМТС модели, почти не влияет на размер и время обучения.

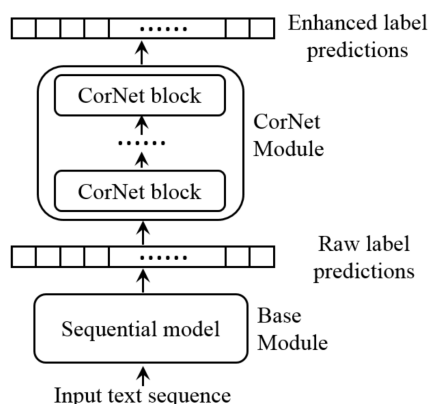


Рис. 14.1: Framework of a CorNet model

## Выводы

### Key Takeaways:

1. Добавление CorNet блока поверх предсказаний основной модели практически бесплатно позволяет улучшить качество модели.
2. Добавление дополнительных CoreNet блоков позволяет дополнительно улучшить качество.



## Другие работы

### Facebook: Compositional Embeddings Using Complementary Partitions for Memory-Efficient Recommendation Systems

Reference: <https://arxiv.org/pdf/1909.02107.pdf>

Keywords: recommendation systems, embeddings, model compression

Современные deep learning модели, которые используются в рекомендательных системах, при обучении используют большое количество категориальных признаков большой размерности. Как правило, для каждого категориального признака обучается матрица эмбедингов, которая хранит вектора для каждого возможного значения признака.

Если размерность признака большая, то размер матрицы эмбедингов может легко достигать нескольких Гб. Это создает определенные проблемы как при обучении так и при использовании сети.

Один из самых простых способов уменьшить размер модели — hashing trick, значения категориального признака хэшируются в меньшее пространство. Hashing trick очень просто в использовании, но коллизии при хэшировании приводят к потере качества.

В статье представлен способ уменьшения размера модели за счет хранения не одной большой матрицы эмбедингов для каждого признака, а нескольких небольших матриц. В рамках экспериментов, авторы показывают, что описанный способ позволяет уменьшить размер модели до 4x раз при этом почти не потеряв в качестве.

### Improving Deep Learning For Airbnb Search

Reference: [1]

Продолжение статьи Applying Deep Learning To Airbnb Search <https://arxiv.org/pdf/1810.09591.pdf>

Довольно интересная статья про переход от GBDT к DL моделям в поиске Airbnb.

В отличие от большинства статей, в данной работе авторы большое внимание уделяют тому какие шаги они набили в процессе перехода к DL и про опыт в целом, а не просто тому какую модель они задеплоили. Не мало места они уделяют обзору того, что хорошо работает в статьях, но у них не заработало.

Особенно мотивирующим получился раздел RETROSPECTIVE.

# Литература

- [1] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tyler Sax, Lanbo Zhang, Aamir Mansawala, Shulin Yang, Bradley Turnbull, and Junshuo Liao. Improving deep learning for airbnb search. *arXiv preprint arXiv:2002.05515*, 2020. URL <https://arxiv.org/pdf/2002.05515.pdf>.
- [2] Suragan Jandial, Ayush Chopra, Mausoom Sarkar, Piyush Gupta, Balaji Krishnamurthy, and Vineeth Balasubramanian. Retrospective loss: Looking back to improve training of deep neural networks. *arXiv preprint arXiv:2006.13593*, 2020. URL <https://arxiv.org/pdf/2006.13593.pdf>.
- [3] Guangxu Xun, Kishlay Jha, Jianhui Sun, and Aidong Zhang. Correlation networks for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1074–1082, 2020. URL <https://dl.acm.org/doi/pdf/10.1145/3394486.3403151>.

## Часть IV

# Big Data Infrastructure

---

16 To Tune or Not to Tune? In Search of Optimal Configurations for Data Analytics	52
17 Lumos: A Library for Diagnosing Metric Regressions in Web-Scale Applications	55

---

# To Tune or Not to Tune?

## In Search of Optimal Configurations for Data Analytics

Reference: <https://dl.acm.org/doi/pdf/10.1145/3394486.3403299>

Code: <https://github.com/ayat-khairi/tuneful-code>

Keywords: spark, configuration tuning, bayesian optimization

### Какую задачу решают авторы?

При разработке Spark приложений, разработчики не так часто занимаются вопросом подбора оптимальных параметров (конфигарации) для запуска приложения.

Это может приводить к различным последствиям, например,

- аллокация под приложение слишком большого числа ресурсов, в результате другим приложениям может нехватать ресурсов
- аллокация слишком маленького для задачи числа ресурсов, как следствие приложение работает слишком медленно или вообще не работает из-за нехватки ресурсов

Существуют инструменты для тюнинга конфигураций приложений, которые позволяют избежать подобных ситуаций.

(Тут должен быть блок про предыдущие решения и их недостатки, в работе упоминаются два инструмента: Opentuner<sup>1</sup> и Gunther<sup>2</sup>)

Но им часто требуется большое количество (порядка 100) exploration запусков и ими не очень удобно пользоваться.

В статье авторы предлагают фреймворк Tuneful, который интегрирован в экосистему Spark'a и позволяет относительно быстро тюнить конфигурации приложений.

### Как решают

На рисунке 16.1 показано как Tuneful интегрирован в Spark.

Tuneful состоит из трех основных компонент: Significance Analyzer, Cost Modeler и Similarity Analyzer.

---

<sup>1</sup><http://opentuner.org>

<sup>2</sup>[https://link.springer.com/chapter/10.1007/978-3-642-40047-6\\_42](https://link.springer.com/chapter/10.1007/978-3-642-40047-6_42)

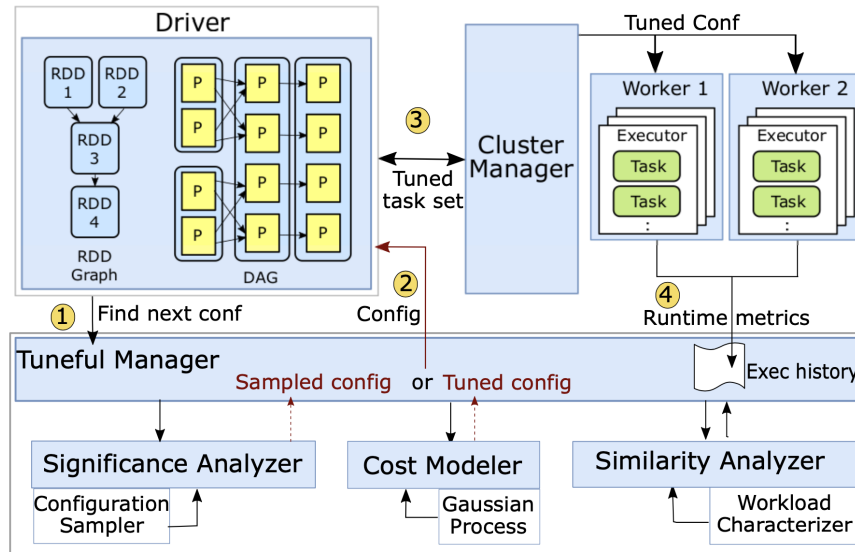


Рис. 16.1: Tuneful-Spark integration: (1) При запуске приложения, Driver запрашивает конфигурацию в Tuneful. (4) После завершения приложения, метрики (время работы, потребление ресурсов итд) отправляются в Tuneful для обновления модели оптимизации.

Кратко работа Tuneful выглядит следующим образом:

**Significance Analyzer** Также как и в большинстве фреймворков для оптимизации гиперпараметров ML алгоритмов, процесс оптимизации в Tuneful можно разбить на фазу поиска (exploration) и фазу использования лучшей из найденных конфигураций (exploitation). В начале фазы поиска Significance Analyzer пытается выделить application-specific важные параметры — запускает приложение с разными конфигурациями, чтобы понять, какие параметры сильнее всего влияют на время работы приложения.

**Cost Modeler** После того как было выполнено достаточное количество exploration запусков (на практике требуется около 35), Cost Modeler использует информацию об exploration запусках и строит модель, которая позволяет по **конфигурации важных параметров** предсказывать время работы приложения (неважные параметры при этом остаются фиксированными).

Для построения модели Cost Modeler использует Гауссовские процессы.

Построенная модель используется для получения конфигурации для очередного запуска приложения.

После каждого очередного запуска приложения, Cost Modeler получает информацию о времени работы приложения и обновляет модель.

Когда Tuneful находит достаточно хорошую конфигурацию для приложения, процесс тюнинга останавливается и для последующих запусков используется найденная конфигурация.

**Similarity Analyzer** Когда Tuneful используется уже продолжительное время (тюнил конфигурации для большого числа приложений), то при тюнинге параметров для нового приложения, Tuneful может использовать информацию о *похожих* приложениях для более удачного старта процесса тюнинга.

Поиском похожих приложений занимается Similarity Analyzer.

Для того чтобы посчитать похожесть между приложениями, авторы описывают их с помощью большого количества метрик, которые собирают во время работы приложения. На собранных метриках обучают autoencoder, для того чтобы для приложения получить описание в виде вектора меньшей размерности.

## Experiments

Авторы проводят большое количество экспериментов для того чтобы оценить качество предложенного фреймворка.

Для экспериментов используют два кластера: первый из 20 машин и второй из 4х.

В качестве приложений для оптимизации авторы выбрали 5 довольно сильно отличающихся друг от друга приложений.

В качестве бэйслайнов авторы используют Random Search, Opentuner и Gunther.

Авторы показывают, что Tuneful позволяет примерно в 3 раза быстрее найти хорошие конфигурации, и что использование Similarity Analyzer'a позволяет в 3 раза ускорить тюнинг для новых приложений.

## Выводы

### Key Takeaways:

1. Если приложение запускается суммарно меньше 100 раз, то нет смысла пытаться его тюнить используя специальные фреймворки из-за большого числа exploration запусков.
2. Если речь идет о приложении, которое, например, запускается по крону каждый день, то оно идеально подходит для тюнинга конфигурации, но нужно быть готовым к тому, что в течении первых 30 запусков почти не будет выигрыша в суммарном времени работы.

## Мое мнение

- В целом, не очень понятно насколько это значимая работа, так как это первая статья, которую я смотрел в области тюнинга конфигураций приложений в рамках кластера.  
Думаю, что полезно начать смотреть в этом направлении.
- Немного странным выглядит выбор бэйслайнов: почему из числа методов для тюнинга гиперпараметров используется только Random Search, а не какой нибудь более современный фреймворк?

# Lumos: A Library for Diagnosing Metric Regressions in Web-Scale Applications

Microsoft

Reference: <https://arxiv.org/abs/2006.12793>

Keywords: online metrics, A/B experiments, anomaly detection

Code: <https://github.com/microsoft/MS-Lumos>

## Какую задачу решают авторы?

Неотъемлемой частью любого web-scale приложения является система мониторинга, которая отслеживает большое количество метрик в течении времени, и сообщает об аномалиях.

Быстрое обнаружение аномалий и возможных причин их возникновения очень важно для стабильной работы сервиса и хорошего пользовательского опыта.

Существующие state-of-the-art алгоритмы детекции аномалий имеют довольно высокий false-positive rate и сами по себе не могут дать ответ на вопрос, что могло послужить причиной возникновения аномалии.

Уменьшение доли ложных срабатываний и помощь в поиске истинной причины появления аномалии могут существенно сэкономить время разработчиков.

В данной статье авторы представляют Python фреймворк Lumos, который решает обозначенные проблемы.