МГТУ имени Н.Э. Баумана
Факультет "Информатика, искусственный интеллект и системы управления"
Кафедра "Системы обработки информации и управления"
Дисциплина "Методы машинного обучения"

Отчёт по домашнему заданию

Группа: ИУ5-22М
Студент: Кривцов Н.А.
Преподаватель: Гапанюк Ю.Е.

Москва — 2022

1. ЗАДАНИЕ

Домашнее задание по дисциплине направлено на анализ современных методов машинного обучения и их применение для решения практических задач. Домашнее задание включает три основных этапа:

— выбор задачи;

— теоретический этап;

— практический этап.

Этап выбора задачи предполагает анализ ресурса paperswithcode. Данный ресурс включает описание нескольких тысяч современных задач в области машинного обучения. Каждое описание задачи содержит ссылки на наиболее современные и актуальные научные статьи, предназначенные для решения задачи (список статей регулярно обновляется авторами ресурса). Каждое описание статьи содержит ссылку на репозиторий с открытым исходным кодом, реализующим представленные в статье эксперименты. На этапе выбора задачи обучающийся выбирает одну из задач машинного обучения, описание которой содержит ссылки на статьи и репозитории с исходным кодом.

Теоретический этап включает проработку как минимум двух статей, относящихся к выбранной задаче. Результаты проработки обучающийся излагает в теоретической части отчета по домашнему заданию, которая может включать:

— описание общих подходов к решению задачи;

— конкретные топологии нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения, предназначенных для решения задачи;

— математическое описание, алгоритмы функционирования, особенности обучения используемых для решения задачи нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения;

— описание наборов данных, используемых для обучения моделей;

— оценка качества решения задачи, описание метрик качества и их значений;

— предложения обучающегося по улучшению качества решения задачи.

Практический этап включает повторение экспериментов авторов статей на основе представленных авторами репозиториев с исходным кодом и возможное улучшение обучающимися полученных результатов. Результаты проработки обучающийся излагает в практической части отчета по домашнему заданию, которая может включать:

— исходные коды программ, представленные авторами статей, результаты документирования программ обучающимися с использованием диаграмм UML, путем визуализации топологий нейронных сетей и другими способами;

— результаты выполнения программ, вычисление значений для описанных в статьях метрик качества, выводы обучающегося о воспроизводимости экспериментов авторов статей и соответствии практических экспериментов теоретическим материалам статей;

— предложения обучающегося по возможным улучшениям решения задачи, результаты практических экспериментов (исходные коды, документация) по возможному улучшению решения задачи.

Отчет по домашнему заданию должен содержать:

— Титульный лист.

— Постановку выбранной задачи машинного обучения, соответствующую этапу выбора задачи.

— Теоретическую часть отчета.

— Практическую часть отчета.

— Выводы обучающегося по результатам выполнения теоретической и практической частей.

— Список использованных источников.

2. ПОСТАНОВКА ЗАДАЧИ

В результате анализа содержимого ресурса «Papers with code» была выбрана область рекомендательных систем (Recommendation Systems). В данной области было решено изучить подходы к системам рекомендации новостей (News Recommendation).

3. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В рамках настоящей работы рассматриваются материалы статей "Neural News Recommendation with Attentive Multi-View Learning (NAML)" и "Neural News Recommendation with Multi-Head Self-Attention (NRMS)". Обе статьи рассматривают возможные архитектуры нейронных сетей для решения задачи новостных рекомендаций. В основе обоих архитектур лежат автокодировщики новостей и автокодировщики пользователей. Ключевые отличия между подходами заключены в слоях автокодировщиков и в реализациях механизмов внимания.

3.1. Attentive Multi-View Learning

Ключевая особенность NAML — расширение пространства признаков, выделяемых из новостного контента, путем имплементации различных "точек зрения" (view) на одни и те же новости. К таким "точкам зрения" авторы отнесли само содержание новостей, заголовок, а также набор категориальных тегов, закрепленных за новостями информационным ресурсом. Обобщенная архитектура NAML представлена на рисунке 1.

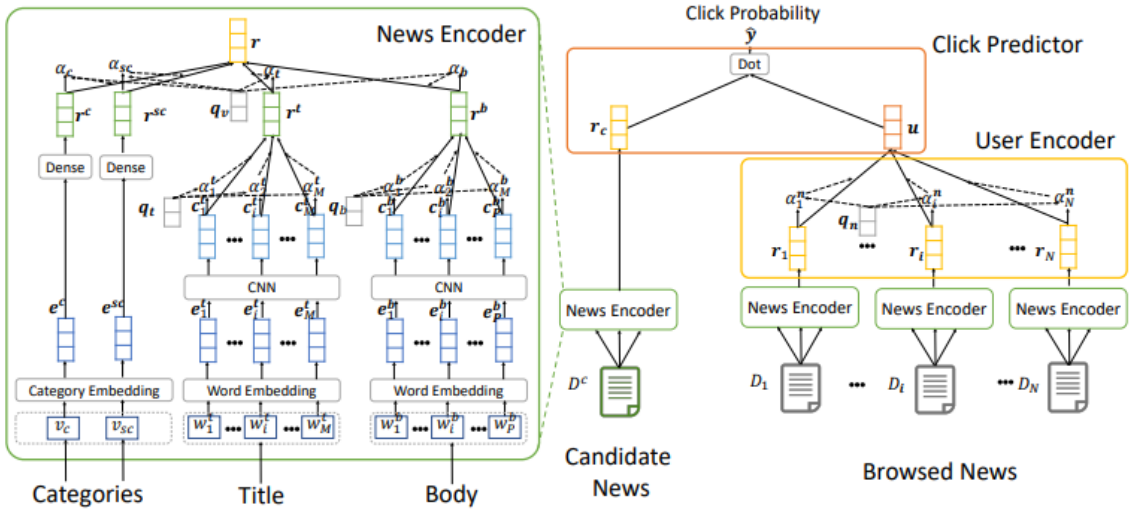


Рисунок 1 — архитектура NAML

3.1.1. News Encoder

Кодировщик новостей разделен на три самостоятельных кодировщика (по одному на каждую новостную "точку зрения"), объединенных общим механизмом аттентивного пуллинга. Кодировщики заголовков и содержания состоят из трех слоев: embedding входных слов, CNN, определяющая контекст использования тех или иных слов, и механизм внимания на уровне слов, определяющий значимость различных слов в различных контекста Кодировщик категорий состоит из двух слоев:

embedding идентификаторов категорий и подкатегорий и MLP. Аттентивный пуллинг определяет, какой вид информации о новостях наиболее важен для конкретной новости.

3.1.2. User Encoder

Кодировщик пользователей выявляет предпочтения и интересы читателей новостей, используя историю прочитанного ими контента. Каждая прочитанная новость кодируется News Encoder-ом, а затем попадает на вход механизма внимания на уровне новостей, который определяет значимость той или иной новости для того или иного пользователя.

3.1.3. Click Predictor

Последний компонент архитектуры NAML предсказывает вероятность прочтения i-ой новости j-м пользователем, используя представления об обоих, полученных из соответствующих кодировщиков.

3.1.4. Модель обучения

Для каждой успешно рекомендованной новости происходит выборка K новостей из той же сессии чтения, которые пользователь *не выбрал*. Таким образом, качестве математической модели обучения авторы статьи используют псевдо-классификацию по K+1 классам. Оценка вероятности выбора той или иной новости рассчитывается по следующей формуле:

$$p_i = \frac{\exp(\hat{y}_i^+)}{\exp(\hat{y}_i^+) + \sum_{j=1}^{K} \exp(\hat{y}_{i,j}^-)},$$

В качестве функции ошибки авторами выбрана функция negative log-likelihood.

3.1.5. Используемый набор данных и параметры эксперимента

Набор обучающих данных представляет собой коллекцию логов MSN News за один месяц (13.12.2018-12.01.2019). В тестовую выборку вошли новости за последнюю неделю собранных экземпляров, остальные новости вошли в обучающую выборку. Также авторами была случайным образом сгенерирована валидационная выборка, представляющая собой 10% от всего датасета. Статистические данные о датасете представлены на рисунке 2.

# users	10,000	**avg. # words per title**	11.29
# news	42,255	**avg. # words per body**	730.72
# impressions	445,230	**# positive samples**	489,644
# samples	7,141,584	**# negative samples**	6,651,940

Рисунок 2 — статистические данные о датасете

Размерности эмбеддингов для слов и новостный категорий составляют 300 и 100 соответственно. Количество фильтров сверточных сетей — 400, с размером окна, равным 3. Размерность выхода полносвязного слоя кодировщика категорий — 400. Размерность query-векторов механизмов внимания выбрана равной 200. Параметр K (количество невыбранных новостей на каждую правильно выбранную новость) равен 4. Для избежания переобучения авторы применяют 20-процентный dropout на каждом слое.

3.1.6. Оценка качества модели

Для оценки качества модели используются следующие метрики: AOC (Area Under Curve), MRR (Mean Reciprocal Rank) и nDCG (Normalized Discounted Cumulative Gain).

В качестве базовых подходов для сравнения с моделью были использованы следующие архитектуры:

— **LibFM** — современный метод матричной факторизации признаков;

— **CNN** — использует сверточные сети для конкатенации новостных представлений;

— **DSSM** — глубокая структурированная семантическая модель, использующая хеширование слов путем выделения триграмм и классификации несколькими полносвязными слоями;

— **Wide&Deep** — использует композицию широких линейных каналов и глубоких нейронных каналов;

— **DeepFM —** совмещает подходы матричной факторизации и глубоких нейронных сетей;

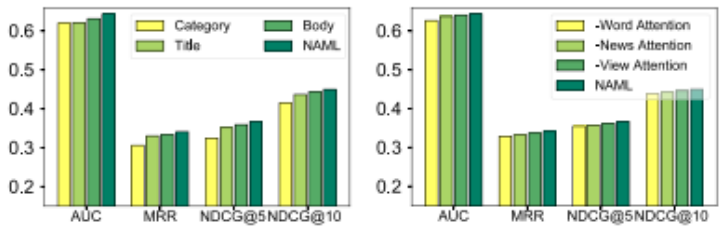
— **DFM** — глубокая синтезирующая модель ,совмещающая в себе слои различных степеней глубины;

— **DKN** — глубокая новостная рекомендательная система, основанная на сверточных сетях с базами знаний.

Каждый эксперимент был повторен 10 раз, и в сводную таблицу (рис. 3) внесены средние значения эффективности. График эффективности механизмов внимания, а также обучения по нескольким точкам зрения, представлена на рисунке 4.

Methods	AUC	MRR	nDCG@5	nDCG@10
LibFM	0.5880	0.3054	0.3202	0.4090
CNN	0.5909	0.3068	0.3221	0.4109
DSSM	0.6114	0.3188	0.3261	0.4185
Wide&Deep	0.5846	0.3009	0.3167	0.4062
DeepFM	0.5896	0.3066	0.3221	0.4117
DFM	0.5996	0.3133	0.3288	0.4165
DKN	0.5966	0.3113	0.3286	0.4165
NAML*	**0.6434**	**0.3411**	**0.3670**	**0.4501**

Рисунок 3 — сводная таблица эффективности NAML



(a) Multi-view learning. (b) Attention networks.

Рисунок 4 — эффективность механизмов внимания и multi-view обучения.

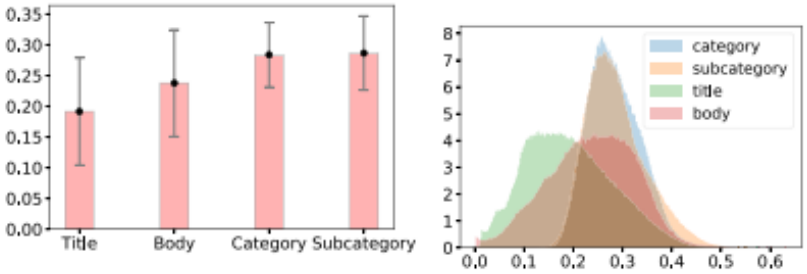


Рисунок 5 — статистика распределений attention весов в multi-view.

3.2. Multi-Headed Self-Attention

Ключевая особенность NRMS — механизмы самовнимания в кодировщиках образованы из нескольких "голов внимания" (attention heads). Такой подход позволяет модели определять согласованность и взаимодействия слов друг с другом в содержимом новостей. В сравнении с NAML, в данной архитектуре multi-headed механизмы самовнимания занимают положение сверточных слоев NAML. Обобщенная архитектура NRMS представлена на рисунке 6.

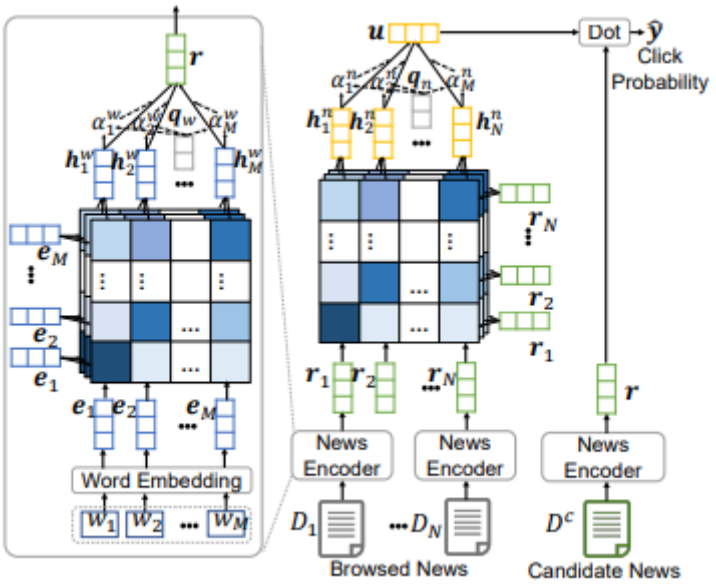


Рисунок 6 — обобщенная архитектура NRMS

3.2.1. News Encoder

Кодировщик новостей состоит из трех слоев: эмбеддинг входного новостного контента; multi-headed механизм самовнимания на уровне слов; выявляющий контекст использования ключевых слов, их согласованность и взаимодействия в содержании новостей, уделяя внимание даже удаленным друг от друга словам (в отличие от CNN); и слоя аддитивного внимания, определяющим значимость тех или иных ключевых слов в тех или иных новостях.

3.2.2. User Encoder

Кодировщик пользователей состоит из двух слоев. Первый — multie-headed механизм самовнимания на уровне *новостей*, устанавливает возможную согласованность и взаимодействия между *новостями*, которые прочитал тот или иной пользователь. Второй — слой аддитивного внимания, определяющий значимость тех или иных *новостей* для представления того или иного пользователя.

3.2.3. Click Predictor

Последний компонент архитектуры NAML предсказывает вероятность прочтения i-ой новости j-м пользователем, используя представления об обоих, полученных из соответствующих кодировщиков.

3.2.4. Модель обучения

Для каждой успешно рекомендованной новости происходит выборка K новостей из той же сессии чтения, которые пользователь *не выбрал*. Таким образом, качестве математической модели обучения авторы статьи используют псевдо-классификацию по K+1 классам. Оценка вероятности выбора той или иной новости рассчитывается по следующей формуле:

$$p_i = \frac{\exp(\hat{y}_i^+)}{\exp(\hat{y}_i^+) + \sum_{j=1}^{K} \exp(\hat{y}_{i,j}^-)},$$

В качестве функции ошибки авторами выбрана функция negative log-likelihood.

3.2.5. Используемый набор данных и параметры эксперимента

Набор обучающих данных представляет собой коллекцию логов MSN News за один месяц (13.12.2018-12.01.2019). В тестовую выборку вошли новости за последнюю неделю собранных экземпляров, остальные новости вошли в обучающую выборку. Также авторами была случайным образом сгенерирована валидационная выборка, представляющая собой 10% от всего датасета. Статистические данные о датасете представлены на рисунке 7.

# users	10,000	avg. # words per title	11.29
# news	42,255	# positive samples	489,644
# impressions	445,230	# negative samples	6,651,940

Рисунок 7 — статистические данные о датасете

Размерность Glove-эмбеддинга для слов составляет 300. Сети само-внимания состоят из 16 heads, размерность выхода каждой из которых равна 16. Размерность query-векторов аддитивных механизмов внимания выбрана равной 200. Параметр K (количество невыбранных новостей на каждую правильно выбранную новость) равен 4. Для избежания переобучения авторы применяют 20-процентный dropout на каждом слое. Используемый оптимизатор — Adam.

3.2.6. Оценка качества модели

Для оценки качества модели используются следующие метрики: AOC (Area Under Curve), MRR (Mean Reciprocal Rank) и nDCG (Normalized Discounted Cumulative Gain).

В качестве базовых подходов для сравнения с моделью были использованы следующие архитектуры:

— **LibFM** — современный метод матричной факторизации признаков;

— **DSSM** — глубокая структурированная семантическая модель, использующая хеширование слов путем выделения триграмм и классификации несколькими полносвязными слоями;

— **Wide&Deep** — использует композицию широких линейных каналов и глубоких нейронных каналов;

— **DeepFM —** совмещает подходы матричной факторизации и глубоких нейронных сетей;

— **DFM** — глубокая синтезирующая модель ,совмещающая в себе слои различных степеней глубины;

— **DKN** — глубокая новостная рекомендательная система, основанная на сверточных сетях с базами знаний;

— **Conv3D** — рекомендательная нейронная сеть, использующая 3D-свертку для выявления представлений о пользователях;

— **GRU** — рекомендательная нейронная сеть, использующая GRU (Gated Recurrent Units) для выявления представлений о пользователях.

Каждый эксперимент был повторен 10 раз, и в сводную таблицу (рис. 8) внесены средние значения эффективности. Авторы также произвели оценку времени, затраченного на обучение модели (рис. 9), и эффективности различных механизмов внимания (рис. 11).

Methods	AUC	MRR	nDCG@5	nDCG@10
LibFM	0.5661	0.2414	0.2689	0.3552
DSSM	0.5949	0.2675	0.2881	0.3800
Wide&Deep	0.5812	0.2546	0.2765	0.3674
DeepFM	0.5830	0.2570	0.2802	0.3707
DFM	0.5861	0.2609	0.2844	0.3742
DKN	0.6032	0.2744	0.2967	0.3873
Conv3D	0.6051	0.2765	0.2987	0.3904
GRU	0.6102	0.2811	0.3035	0.3952
NRMS*	**0.6275**	**0.2985**	**0.3217**	**0.4139**

Рисунок 8 — сводная таблица эффективности NRMS

Method	# Parameters	News Encoding Time	User Encoding Time
DKN	681 K	46.4 s	10.1 min
Conv3D	575 K	29.8 min	16.7 min
GRU	541 K	59.8 s	148.0 min
NRMS	530 K	39.7 s	6.7 min

Рисунок 9 — сравнение времени обучения моделей

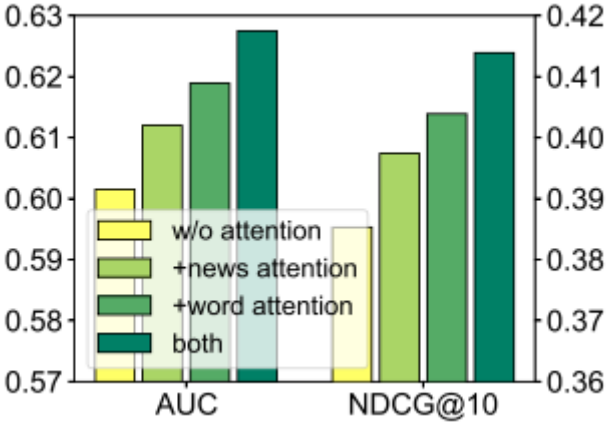


Рисунок 10 — эффективность различных уровней внимания

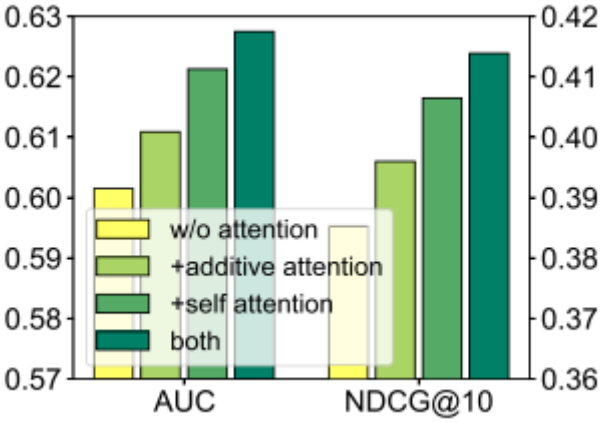


Рисунок 11 — эффективность различных слоев внимания

4. ПРАКТИЧЕСКАЯ ЧАСТЬ

4.1. Attentive Multi-View Learning

Глобальные настройки и зависимости

```
import sys
import os
import numpy as np
import zipfile
from tqdm import tqdm
import scrapbook as sb
from tempfile import TemporaryDirectory
import tensorflow as tf
tf.get_logger().setLevel('ERROR') # only show error messages

from recommenders.models.deeprec.deeprec_utils import download_deeprec_resources
from recommenders.models.newsrec.newsrec_utils import prepare_hparams
from recommenders.models.newsrec.models.naml import NAMLModel
from recommenders.models.newsrec.io.mind_all_iterator import MINDAllIterator
from recommenders.models.newsrec.newsrec_utils import get_mind_data_set

print("System version: {}".format(sys.version))
print("Tensorflow version: {}".format(tf.__version__))
```

Задание параметров

```
[ ]  epochs = 5
     seed = 42
     batch_size = 32

     # Options: demo, small, large
     MIND_type = 'demo'
```

Загрузка данных

```
[ ]  tmpdir = TemporaryDirectory()
     data_path = tmpdir.name

     train_news_file = os.path.join(data_path, 'train', r'news.tsv')
     train_behaviors_file = os.path.join(data_path, 'train', r'behaviors.tsv')
     valid_news_file = os.path.join(data_path, 'valid', r'news.tsv')
     valid_behaviors_file = os.path.join(data_path, 'valid', r'behaviors.tsv')
     wordEmb_file = os.path.join(data_path, "utils", "embedding_all.npy")
     userDict_file = os.path.join(data_path, "utils", "uid2index.pkl")
     wordDict_file = os.path.join(data_path, "utils", "word_dict_all.pkl")
     vertDict_file = os.path.join(data_path, "utils", "vert_dict.pkl")
     subvertDict_file = os.path.join(data_path, "utils", "subvert_dict.pkl")
     yaml_file = os.path.join(data_path, "utils", r'naml.yaml')

     mind_url, mind_train_dataset, mind_dev_dataset, mind_utils = get_mind_data_set(MIND_type)

     if not os.path.exists(train_news_file):
         download_deeprec_resources(mind_url, os.path.join(data_path, 'train'), mind_train_dataset)

     if not os.path.exists(valid_news_file):
         download_deeprec_resources(mind_url, \
                                    os.path.join(data_path, 'valid'), mind_dev_dataset)
     if not os.path.exists(yaml_file):
         download_deeprec_resources(r'https://recodatasets.z20.web.core.windows.net/newsrec/', \
                                    os.path.join(data_path, 'utils'), mind_utils)
```

```
100%|██████████| 17.0k/17.0k [00:01<00:00, 10.0kKB/s]
100%|██████████| 9.84k/9.84k [00:01<00:00, 8.42kKB/s]
100%|██████████| 95.0k/95.0k [00:06<00:00, 14.5kKB/s]
```

Генерация гиперпараметров

```
[ ]  hparams = prepare_hparams(yaml_file,
                               wordEmb_file=wordEmb_file,
                               wordDict_file=wordDict_file,
                               userDict_file=userDict_file,
                               vertDict_file=vertDict_file,
                               subvertDict_file=subvertDict_file,
                               batch_size=batch_size,
                               epochs=epochs)
```

```
[ ]  iterator = MINDAllIterator
```

Обучение модели NAML

```
[ ]  model = NAMLModel(hparams, iterator, seed=seed)
```

```
[ ]  print(model.run_eval(valid_news_file, valid_behaviors_file))
```

```
18693it [01:18, 239.62it/s]
7507it [00:30, 249.74it/s]
7538it [00:01, 6423.03it/s]
{'group_auc': 0.4807, 'mean_mrr': 0.2104, 'ndcg@5': 0.2141, 'ndcg@10': 0.2766}
```

```
%%time
model.fit(train_news_file, train_behaviors_file,valid_news_file, valid_behaviors_file)
```

```
1085it [01:49,  9.91it/s]
18693it [01:16, 242.85it/s]
7507it [00:30, 244.22it/s]
7538it [00:01, 6540.79it/s]
2it [00:00, 10.26it/s]at epoch 1
train info: logloss loss:1.4915421532046411
eval info: group_auc:0.5789, mean_mrr:0.2473, ndcg@10:0.3371, ndcg@5:0.2699
at epoch 1 , train time: 109.5 eval time: 116.2
1085it [01:43, 10.48it/s]
18693it [01:16, 243.31it/s]
7507it [00:30, 247.71it/s]
7538it [00:01, 6653.75it/s]
1it [00:00,  9.91it/s]at epoch 2
train info: logloss loss:1.4155106401663222
eval info: group_auc:0.607, mean_mrr:0.2659, ndcg@10:0.3586, ndcg@5:0.2934
at epoch 2 , train time: 103.5 eval time: 115.7
1085it [01:43, 10.45it/s]
18693it [01:16, 244.17it/s]
7507it [00:30, 246.78it/s]
7538it [00:01, 5570.62it/s]
1it [00:00,  9.92it/s]at epoch 3
train info: logloss loss:1.3669039504319291
eval info: group_auc:0.6199, mean_mrr:0.2719, ndcg@10:0.3675, ndcg@5:0.3038
at epoch 3 , train time: 103.8 eval time: 115.7
```

```
]  %%time
   res_syn = model.run_eval(valid_news_file, valid_behaviors_file)
   print(res_syn)
```

```
18693it [01:16, 245.66it/s]
7507it [00:30, 249.77it/s]
7538it [00:01, 6922.17it/s]
{'group_auc': 0.6272, 'mean_mrr': 0.2846, 'ndcg@5': 0.31, 'ndcg@10': 0.376}
CPU times: user 4min 2s, sys: 25.8 s, total: 4min 28s
Wall time: 1min 54s
```

4.2. Multi-Headed Self-Attention

Глобальные настройки и зависимости

```
import sys
import os
import numpy as np
import zipfile
from tqdm import tqdm
import scrapbook as sb
from tempfile import TemporaryDirectory
import tensorflow as tf
tf.get_logger().setLevel('ERROR') # only show error messages

from recommenders.models.deeprec.deeprec_utils import download_deeprec_resources
from recommenders.models.newsrec.newsrec_utils import prepare_hparams
from recommenders.models.newsrec.models.nrms import NRMSModel
from recommenders.models.newsrec.io.mind_iterator import MINDIterator
from recommenders.models.newsrec.newsrec_utils import get_mind_data_set

print("System version: {}".format(sys.version))
print("Tensorflow version: {}".format(tf.__version__))
```

```
/anaconda/envs/tf2/lib/python3.7/site-packages/papermill/iorw.py:50: FutureWarning
  from pyarrow import HadoopFileSystem
System version: 3.7.11 (default, Jul 27 2021, 14:32:16)
[GCC 7.5.0]
Tensorflow version: 2.6.1
```

Задание параметров

```
[ ]  epochs = 5
     seed = 42
     batch_size = 32

     # Options: demo, small, large
     MIND_type = 'demo'
```

Загрузка данных

```
tmpdir = TemporaryDirectory()
data_path = tmpdir.name

train_news_file = os.path.join(data_path, 'train', r'news.tsv')
train_behaviors_file = os.path.join(data_path, 'train', r'behaviors.tsv')
valid_news_file = os.path.join(data_path, 'valid', r'news.tsv')
valid_behaviors_file = os.path.join(data_path, 'valid', r'behaviors.tsv')
wordEmb_file = os.path.join(data_path, "utils", "embedding.npy")
userDict_file = os.path.join(data_path, "utils", "uid2index.pkl")
wordDict_file = os.path.join(data_path, "utils", "word_dict.pkl")
yaml_file = os.path.join(data_path, "utils", r'nrms.yaml')

mind_url, mind_train_dataset, mind_dev_dataset, mind_utils = get_mind_data_set(MIND_type)

if not os.path.exists(train_news_file):
    download_deeprec_resources(mind_url, os.path.join(data_path, 'train'), mind_train_dataset)

if not os.path.exists(valid_news_file):
    download_deeprec_resources(mind_url, \
                               os.path.join(data_path, 'valid'), mind_dev_dataset)
if not os.path.exists(yaml_file):
    download_deeprec_resources(r'https://recodatasets.z20.web.core.windows.net/newsrec/', \
                               os.path.join(data_path, 'utils'), mind_utils)
```

```
100%|██████████| 17.0k/17.0k [00:01<00:00, 9.65kKB/s]
100%|██████████| 9.84k/9.84k [00:01<00:00, 8.93kKB/s]
100%|██████████| 95.0k/95.0k [00:08<00:00, 11.0kKB/s]
```

Генерация гиперпараметров

```
[ ]  hparams = prepare_hparams(yaml_file,
                                wordEmb_file=wordEmb_file,
                                wordDict_file=wordDict_file,
                                userDict_file=userDict_file,
                                batch_size=batch_size,
                                epochs=epochs,
                                show_step=10)
```

Обучение модели NRMS

```
[ ]  iterator = MINDIterator
```

```
[ ]  model = NRMSModel(hparams, iterator, seed=seed)
```

```
[ ]  print(model.run_eval(valid_news_file, valid_behaviors_file))
```

```
586it [00:02, 238.86it/s]
236it [00:04, 57.82it/s]
7538it [00:01, 6381.66it/s]
{'group_auc': 0.4792, 'mean_mrr': 0.2059, 'ndcg@5': 0.2045, 'ndcg@10': 0.2701}
```

```
%%time
model.fit(train_news_file, train_behaviors_file, valid_news_file, valid_behaviors_file)
```

```
step 1080 , total_loss: 1.5155, data_loss: 1.4078: : 1086it [01:07, 16.10it/s]
586it [00:01, 388.70it/s]
236it [00:03, 68.28it/s]
7538it [00:00, 7543.81it/s]
2it [00:00, 16.78it/s]at epoch 1
train info: logloss loss:1.5149870059327746
eval info: group_auc:0.5755, mean_mrr:0.2453, ndcg@10:0.3313, ndcg@5:0.2587
at epoch 1 , train time: 67.4 eval time: 13.3
step 1080 , total_loss: 1.4203, data_loss: 1.3752: : 1086it [01:04, 16.93it/s]
586it [00:01, 412.04it/s]
236it [00:03, 67.25it/s]
7538it [00:00, 9040.56it/s]
2it [00:00, 16.90it/s]at epoch 2
train info: logloss loss:1.4203101933331779
eval info: group_auc:0.5995, mean_mrr:0.2572, ndcg@10:0.3482, ndcg@5:0.273
at epoch 2 , train time: 64.2 eval time: 13.0
step 1080 , total_loss: 1.3770, data_loss: 1.2186: : 1086it [01:05, 16.49it/s]
586it [00:01, 401.41it/s]
236it [00:03, 65.66it/s]
7538it [00:00, 7954.16it/s]
2it [00:00, 16.66it/s]at epoch 3
train info: logloss loss:1.3768525854658686
eval info: group_auc:0.6032, mean_mrr:0.2632, ndcg@10:0.3535, ndcg@5:0.2817
at epoch 3 , train time: 65.9 eval time: 13.3
step 1080 , total_loss: 1.3516, data_loss: 1.2423: : 1086it [01:06, 16.39it/s]
586it [00:01, 390.04it/s]
236it [00:03, 64.53it/s]
7538it [00:01, 5913.76it/s]
```

```
[ ] %%time
    res_syn = model.run_eval(valid_news_file, valid_behaviors_file)
    print(res_syn)
```

```
586it [00:01, 396.35it/s]
236it [00:03, 67.56it/s]
7538it [00:01, 6017.89it/s]
{'group_auc': 0.6127, 'mean_mrr': 0.2697, 'ndcg@5': 0.2912, 'ndcg@10': 0.3625}
CPU times: user 29.8 s, sys: 1.27 s, total: 31.1 s
Wall time: 14.3 s
```

5. ВЫВОДЫ

В рамках домашнего задания был произведен обзор теоретических и практических материалов статей "Neural News Recommendation with Attentive Multi-View Learning (NAML)" и "Neural News Recommendation with Multi-Head Self-Attention (NRMS)". Наилучшее качество решения задачи новостных рекомендаций показала сеть архитектуры NAML. Тем не менее, подход, примененный в NRMS-сети, оказывается более гибким и может быть применен в рекомендательных системах с отличной предметной областью, например, для рекомендации музыки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. News Recommendation | Papers With Code — URL: https://paperswithcode.com/task/news-recommendation (дата обращения: 09.06.2022 г.)
2. Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, Xing Xie — "Neural News Recommendation with Attentive Multi-View Learning" — URL: https://arxiv.org/pdf/1907.05576v1.pdf (дата обращения: 09.06.2022 г.)
3. Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang и Xing Xie — "Neural News Recommendation with Multi-Head Self-Attention" — URL: https://aclanthology.org/D19-1671.pdf (дата обращения: 09.06.2022 г.)
4. NAML: Neural News Recommendation with Attentive Multi-View Learning — URL: https://github.com/microsoft/recommenders/blob/main/examples/00_quick_start/naml_MIND.ipynb (дата обращения: 09.06.2022 г.)
5. NRMS: Neural News Recommendation with Multi-Head Self-Attention — URL: https://github.com/microsoft/recommenders/blob/main/examples/00_quick_start/nrms_MIND.ipynb (дата обращения: 09.06.2022 г.)