Информационный поиск. NLP для рекомендательных систем.

Хрыльченко Кирилл

Математические методы анализа текстов 2021

23 ноября, 2021

Информационный поиск

Задача: глядя на запрос и множество документов, отобрать для пользователя наиболее релевантные документы.

Популярные приложения информационного поиска:

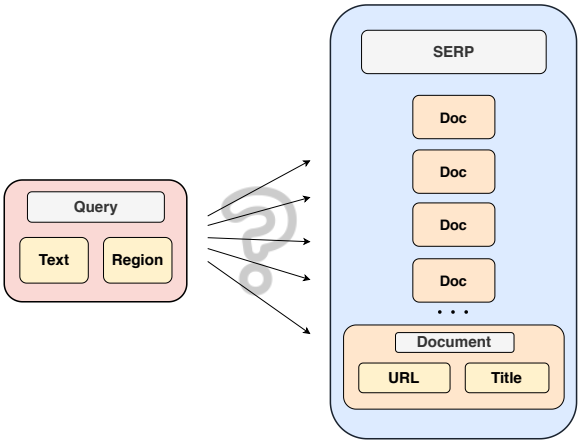
- поиск по базам знаний
- веб-поиск

Методы, основанные на ключевых словах:

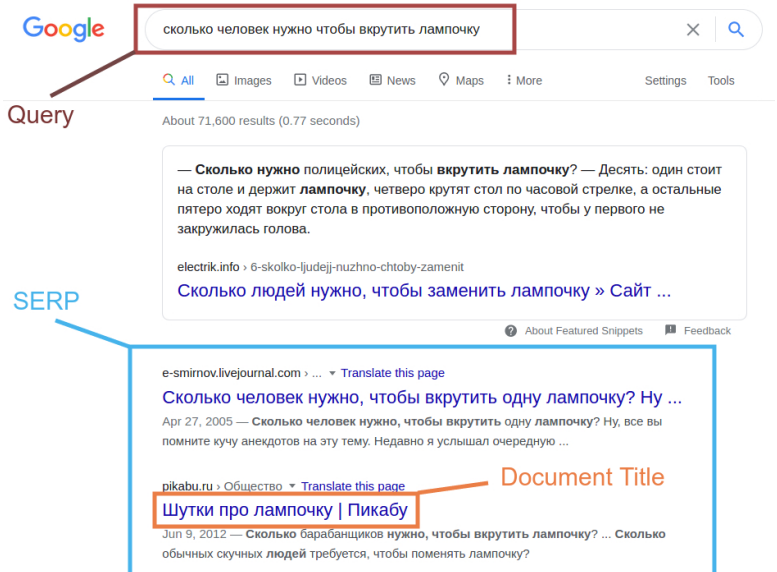
- TF-IDF
- BM-25

Веб-поиск

- Search Engine Results Pages (SERP) — поисковая выдача



Веб-поиск


Google
сколько человек нужно чтобы вкрутить лампочку
Query
All
Images
Videos
News
Maps
More
Settings
Tools
About 71,600 results (0.77 seconds)
— Сколько нужно полицейских, чтобы вкрутить лампочку? — Десять: один стоит на столе и держит лампочку, четверо крутят стол по часовой стрелке, а остальные пятеро ходят вокруг стола в противоположную сторону, чтобы у первого не закружилась голова.
electrik.info › 6-skolko-ljudejj-nuzhno-chtoby-zamenit
Сколько людей нужно, чтобы заменить лампочку » Сайт ...
SERP
About Featured Snippets
Feedback
e-smirnov.livejournal.com › ... Translate this page
Сколько человек нужно, чтобы вкрутить одну лампочку? Ну ...
Apr 27, 2005 — Сколько человек нужно, чтобы вкрутить одну лампочку? Ну, все вы помните кучу анекдотов на эту тему. Недавно я услышал очередную ...
pikabu.ru › Общество Translate this page
Document Title
Шутки про лампочку | Пикабу
Jun 9, 2012 — Сколько барабанщиков нужно, чтобы вкрутить лампочку? ... Сколько обычных скучных людей требуется, чтобы поменять лампочку?

Deep Structured Semantic Models[1]

По аналогии с word2vec, хочется видеть схожесть даже при отсутствии общих слов:

- запрос: univ of penn
- заголовок документа: university of pennsylvania wikipedia the free encyclopedia

Как можно это побороть:

- и запрос, и документ переводятся из мешка слов в вектор из «семантического пространства»
- релевантность документа для запроса определяется как косинус между соответствующими векторами

[1] https://www.microsoft.com/en-us/research/publication/learning-deep-structured-semantic-models-for-web-search-using-clickthrough-data/

Word Hashing

Размерности словарей:

- в словах: 500000. Слишком много
- в буквенных триграммах: 30621

Будем переводить слова (тексты) в векторы побуквенных триграмм:

- good $\rightarrow$ #good# $\rightarrow$ (#go, goo, ood, od#)

и представлять их как мешок, т.е. вектор размерности 30621.

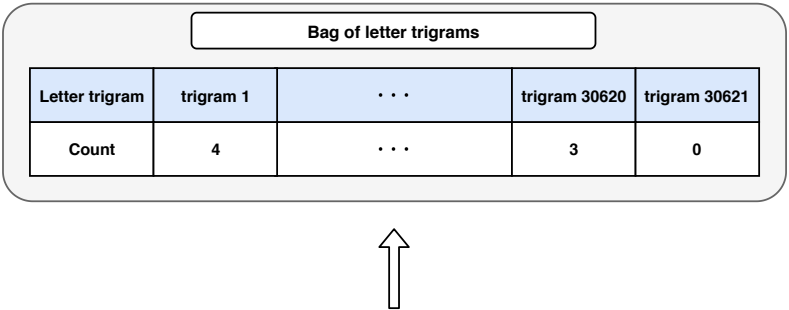
Возникают коллизии — разные слова представляются одинаковыми мешками побуквенных триграмм. Но их не очень много:

	Letter-Bigram		Letter-Trigram	
Word Size	**Token Size**	**Collision**	**Token Size**	**Collision**
40k	1107	18	10306	2
500k	1607	1192	30621	22

Table 1: Word hashing token size and collision numbers as a function of the vocabulary size and the type of letter ngrams.

Из плюсов — более менее решили проблему out-of-vocabulary слов.

Word Hashing



DSSM. Архитектура

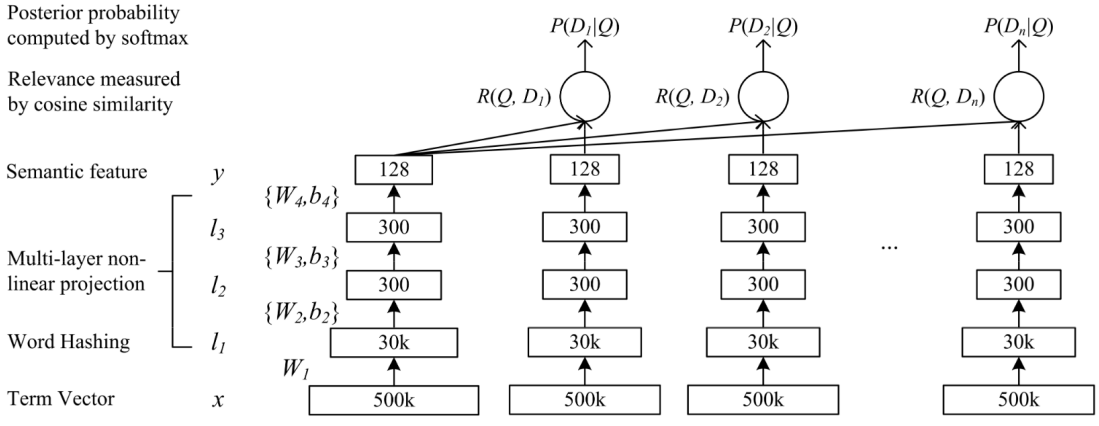


Figure 1: Illustration of the DSSM. It uses a DNN to map high-dimensional sparse text features into low-dimensional dense features in a semantic space. The first hidden layer, with 30k units, accomplishes word hashing. The word-hashed features are then projected through multiple layers of non-linear projections. The final layer's neural activities in this DNN form the feature in the semantic space.

DSSM

Линейные слои:

- перевод из мешка слов в мешок триграмм с помощью умножения на матрицу
- три линейных слоя с tanh в качестве функции активации: $30621 \to 300 \to 300 \to 128$
- проводим через эту сеть текст запроса и документа, считаем косинус
- это двухбашенная архитектура — один раз считаем вектор для запроса

Обучение:

- максимизируем правдоподобие пар (запрос, кликнутый документ)
- негативное сэмплирование по аналогии с word2vec: на каждую пару (запрос, кликнутый документ) берется 4 случайных документа для формирования отрицательных примеров
- SGD, размер батча — 1024

Использование — скорим каждую пару и сортируем по косинусной близости

DSSM. Результаты

#	Models	NDCG@1	NDCG@3	NDCG@10
1	TF-IDF	0.319	0.382	0.462
2	BM25	0.308	0.373	0.455
3	WTM	0.332	0.400	0.478
4	LSA	0.298	0.372	0.455
5	PLSA	0.295	0.371	0.456
6	DAE	0.310	0.377	0.459
7	BLTM-PR	0.337	0.403	0.480
8	DPM	0.329	0.401	0.479
9	DNN	0.342	0.410	0.486
10	L-WH linear	0.357	0.422	0.495
11	L-WH non-linear	0.357	0.421	0.494
12	**L-WH DNN**	**0.362**	**0.425**	**0.498**

Table 2: Comparative results with the previous state of the art approaches and various settings of DSSM.

Convolutional DSSM[2]

Проблема — всё еще рассматриваем запросы и документы как мешки слов в DSSM, не учитывается порядок слов. Модель не способна уловить разницу в значении слова «office» для запросов:

- office excel
- apartment office

Решение — будем смотреть на n-граммы из слов с помощью сверток

[2] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/cikm2014_cdssm_final.pdf

CDSSM. Архитектура

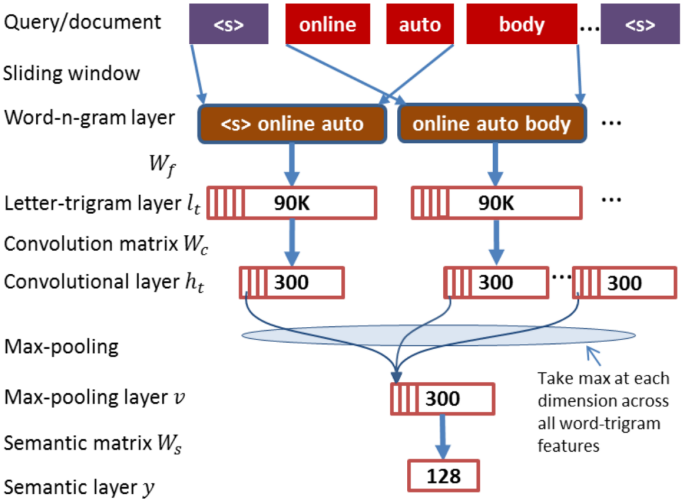


Figure: Архитектура CDSSM с триграммной сверткой.

CDSSM. Архитектура

В начале и конце текста добавляется специальный токен <s>, затем:

- для каждой n-граммы формируется векторное представление
  1. каждое слово в n-грамме переводим в посимвольный триграммный вектор
  2. конкатенируем триграммные векторы по всем словами n-граммы
  3. применяем линейный слой (свертку) сверху этой конкатенации с последующим гиперполическим тангенсом
- по всем n-граммам берется max-pooling
- линейный слой с tanh

CDSSM. Результаты

#	Models	NDCG @1	NDCG @3	NDCG @10
1	DSSM ($J = 50$)	0.327	0.363	0.438
2	CLSM ($J = 50$) win =1	0.340^{α}	0.374^{α}	0.443^{α}
3	CLSM ($J = 50$) win =3	$0.348^{\alpha\beta}$	$0.379^{\alpha\beta}$	$0.449^{\alpha\beta}$
4	CLSM ($J = 50$) win =5	0.344^{α}	0.376^{α}	$0.448^{\alpha\beta}$

Figure: Сравнение CDSSM с прошлой SOTA моделью.

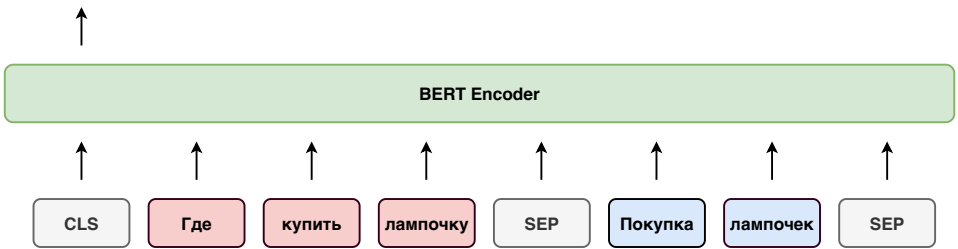
- Обнаружили, что лучше обучать разные сети для запроса и для документа
- Пробовали использовать «тело» документа, вместо заголовка, получили качество значительно хуже

BERT

Основные проблемы DSSM, CDSSM и других «двухбашенных» моделей:

- только частично учитываем порядок слов
- векторы для документа и запроса формируются независимо[3]

В 2019-м году Google внедрил в поиск BERT[4]:



[3] https://arxiv.org/abs/1711.08611
[4] https://arxiv.org/pdf/1901.04085.pdf

Персонализация веб-поиска. Context-Aware Recommender System[5]

Используем не только последний запрос, но также пользовательскую историю — его прошлые запросы и клики.

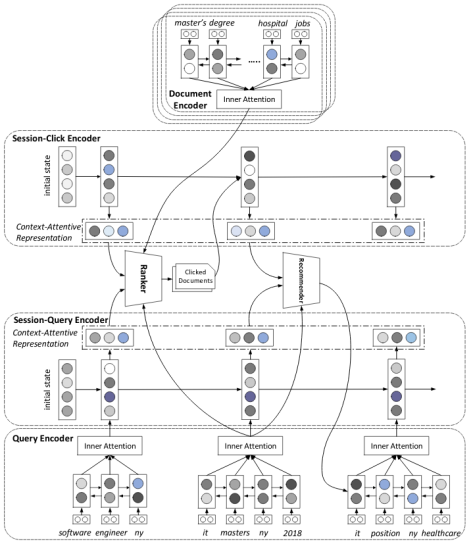
Модель **CARS** решает сразу две задачи:

- ранжирует документы
- генерирует саджест для следующего запроса

[5] https://arxiv.org/pdf/1906.02329.pdf

CARS. Архитектура

- BiLSTM по словам запроса с inner-attention, формирует векторы запросов
- BiLSTM по векторам запросов
- аналогично две BiLSTM для кликнутых документов
- LSTM — декодер для next query suggestion



inner-attention — https://arxiv.org/pdf/1605.09090.pdf

CARS. Результаты

Model	MAP	MRR	NDCG		
			@1	@3	@10
Traditional IR-models					
BM25 [40]	0.230	0.206	0.206	0.269	0.319
QL [39]	0.195	0.166	0.166	0.213	0.276
FixInt [43]	0.242	0.224	0.212	0.275	0.332
Single-task Learning					
DRMM [14]	0.201	0.228	0.129	0.223	0.264
DSSM [18]	0.283	0.307	0.188	0.231	0.341
CLSM [44]	0.313	0.341	0.205	0.252	0.373
ARC-I [16]	0.401	0.411	0.259	0.374	0.463
ARC-II [16]	0.455	0.465	0.309	0.434	0.521
DUET [33]	0.479	0.490	0.332	0.462	0.546
Match Tensor [19]	0.481	0.501	0.345	0.472	0.555
Multi-task Learning					
M-NSRF [2]	0.491	0.502	0.348	0.474	0.557
M-Match Tensor [2]	0.505	0.518	0.368	0.491	0.567
CARS	**0.531**	**0.542**	**0.391**	**0.517**	**0.596**

Figure: Метрики на датасете AOL.

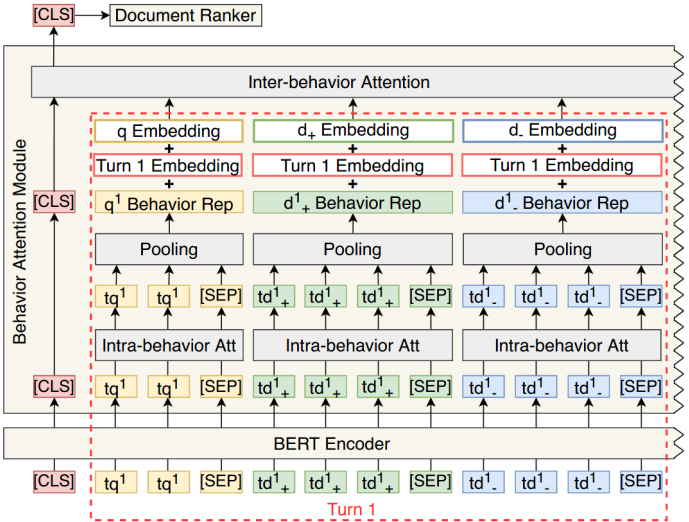
Behavior Aware Transformer (HBA) [6]

Формируем последовательность:

1. запрос
2. кликнутый док
3. скипнутый док

BERT применяется иерархически:

1. ко всему тексту
2. к последовательности закодированных событий



[6] http://ciir-publications.cs.umass.edu/getpdf.php?id=1383

HBA. Результаты

Models	MRR	nDCG		
		@1	@3	@10
CARS[1] [3]	0.4538	0.2940	0.4249	0.5109
BERT [12]	0.5198	0.3592	0.4984	0.5813
BERT-Concat-Q	0.5196	0.3596	0.4977	0.5806
BERT-Concat-QC	0.5340	0.3759	0.5149	0.5934
BERT-Concat-QCS	0.5366	0.3787	0.5174	0.5954
HBA-Transformers-QC	**0.5450**‡	**0.3866**‡	**0.5291**‡	**0.6021**‡
HBA-Transformers-QCS	0.5446‡	0.3850‡	0.5268‡	0.6012‡

Figure: Метрики на датасете AOL. BERT-Concat — обычный BERT над текстом.

Session-based recommendations

Нас интересуют два типа ранжирования:

- pointwise — рекомендуемые продукты оцениваются независимо друг от друга
- pairwise — при обучении пытаемся увеличивать разницу между скорам положительного и отрицательного сэмпла, например:

$$\log \sigma(r_+ - r_-) \to \max$$

Пусть:

- U — множество продуктов, с которыми взаимодействует пользователь
- $u_1, \ldots, u_T$ — пользовательская сессия взаимодействий с продуктами

Тогда можно учить «языковую» модель $P(u_t \mid u_1, \ldots, u_{t-1})$ для «языка», в котором словами являются продукты, а документы — это истории пользователей.
Используем негативное семплирование — учим pointwise ранжирование.

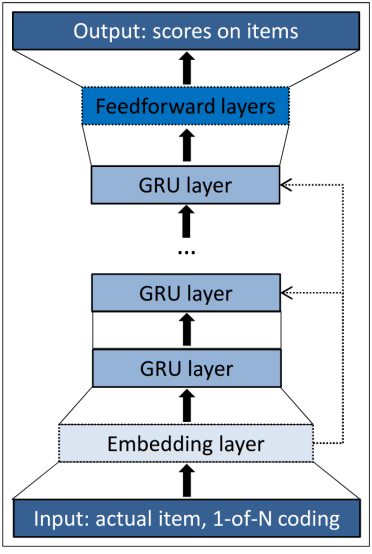
GRU4Rec[7]

Обучение:

- используют попарное ранжирование
- утверждают, что при pointwise ранжировании модель хуже сходится

Table 3: Recall@20 and MRR@20 for different types of a single layer of GRU, compared to the best baseline (item-KNN). Best results per dataset are highlighted.

Loss / #Units	RSC15		VIDEO	
	Recall@20	MRR@20	Recall@20	MRR@20
TOP1 100	0.5853 (+15.55%)	0.2305 (+12.58%)	0.6141 (+11.50%)	0.3511 (+3.84%)
BPR 100	0.6069 (+19.82%)	0.2407 (+17.54%)	0.5999 (+8.92%)	0.3260 (-3.56%)
Cross-entropy 100	0.6074 (+19.91%)	0.2430 (+18.65%)	0.6372 (+15.69%)	0.3720 (+10.04%)
TOP1 1000	0.6206 (+22.53%)	**0.2693 (+31.49%)**	**0.6624 (+20.27%)**	**0.3891 (+15.08%)**
BPR 1000	**0.6322 (+24.82%)**	0.2467 (+20.47%)	0.6311 (+14.58%)	0.3136 (-7.23%)
Cross-entropy 1000	0.5777 (+14.06%)	0.2153 (+5.16%)	–	–



[7] https://arxiv.org/pdf/1511.06939.pdf

BERT4Rec[9]

SASRec[8] — однонаправленный декодер трансформера (аналогично GPT), существенно обогнал прошлые SOTA модели.

BERT4Rec:

- модель: энкодер трансформера
- задача: маскируют и учатся восстанавливать взаимодействия в пользовательской истории — Cloze task, masked language modeling
- применяют без дообучения, добавляют <MASK> в конце истории

[8] https://arxiv.org/pdf/1808.09781.pdf
[9] https://arxiv.org/pdf/1904.06690.pdf

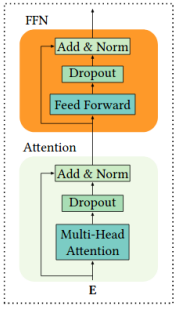
BERT4Rec. Результаты

Datasets	Metric	POP	BPR-MF	NCF	FPMC	GRU4Rec	GRU4Rec$^+$	Caser	SASRec	BERT4Rec	Improv.
Beauty	HR@1	0.0077	0.0415	0.0407	0.0435	0.0402	0.0551	0.0475	0.0906	**0.0953**	5.19%
	HR@5	0.0392	0.1209	0.1305	0.1387	0.1315	0.1781	0.1625	0.1934	**0.2207**	14.12%
	HR@10	0.0762	0.1992	0.2142	0.2401	0.2343	0.2654	0.2590	0.2653	**0.3025**	14.02%
	NDCG@5	0.0230	0.0814	0.0855	0.0902	0.0812	0.1172	0.1050	0.1436	**0.1599**	11.35%
	NDCG@10	0.0349	0.1064	0.1124	0.1211	0.1074	0.1453	0.1360	0.1633	**0.1862**	14.02%
	MRR	0.0437	0.1006	0.1043	0.1056	0.1023	0.1299	0.1205	0.1536	**0.1701**	10.74%
Steam	HR@1	0.0159	0.0314	0.0246	0.0358	0.0574	0.0812	0.0495	0.0885	**0.0957**	8.14%
	HR@5	0.0805	0.1177	0.1203	0.1517	0.2171	0.2391	0.1766	0.2559	**0.2710**	5.90%
	HR@10	0.1389	0.1993	0.2169	0.2551	0.3313	0.3594	0.2870	0.3783	**0.4013**	6.08%
	NDCG@5	0.0477	0.0744	0.0717	0.0945	0.1370	0.1613	0.1131	0.1727	**0.1842**	6.66%
	NDCG@10	0.0665	0.1005	0.1026	0.1283	0.1802	0.2053	0.1484	0.2147	**0.2261**	5.31%
	MRR	0.0669	0.0942	0.0932	0.1139	0.1420	0.1757	0.1305	0.1874	**0.1949**	4.00%
ML-1m	HR@1	0.0141	0.0914	0.0397	0.1386	0.1583	0.2092	0.2194	0.2351	**0.2863**	21.78%
	HR@5	0.0715	0.2866	0.1932	0.4297	0.4673	0.5103	0.5353	0.5434	**0.5876**	8.13%
	HR@10	0.1358	0.4301	0.3477	0.5946	0.6207	0.6351	0.6692	0.6629	**0.6970**	4.15%
	NDCG@5	0.0416	0.1903	0.1146	0.2885	0.3196	0.3705	0.3832	0.3980	**0.4454**	11.91%
	NDCG@10	0.0621	0.2365	0.1640	0.3439	0.3627	0.4064	0.4268	0.4368	**0.4818**	10.32%
	MRR	0.0627	0.2009	0.1358	0.2891	0.3041	0.3462	0.3648	0.3790	**0.4254**	12.24%
ML-20m	HR@1	0.0221	0.0553	0.0231	0.1079	0.1459	0.2021	0.1232	0.2544	**0.3440**	35.22%
	HR@5	0.0805	0.2128	0.1358	0.3601	0.4657	0.5118	0.3804	0.5727	**0.6323**	10.41%
	HR@10	0.1378	0.3538	0.2922	0.5201	0.5844	0.6524	0.5427	0.7136	**0.7473**	4.72%
	NDCG@5	0.0511	0.1332	0.0771	0.2239	0.3090	0.3630	0.2538	0.4208	**0.4967**	18.04%
	NDCG@10	0.0695	0.1786	0.1271	0.2895	0.3637	0.4087	0.3062	0.4665	**0.5340**	14.47%
	MRR	0.0709	0.1503	0.1072	0.2273	0.2967	0.3476	0.2529	0.4026	**0.4785**	18.85%

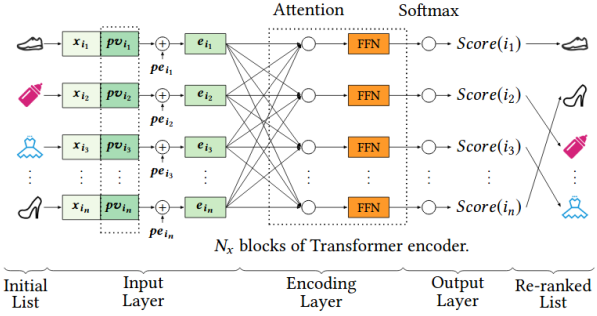
BERT4Rec. Ablation Study

Architecture	Dataset			
	Beauty	Steam	ML-1m	ML-20m
$L = 2, h = 2$	0.1832	0.2241	0.4759	0.4513
w/o PE	0.1741	0.2060	0.2155↓	0.2867↓
w/o PFFN	0.1803	0.2137	0.4544	0.4296
w/o LN	0.1642↓	0.2058	0.4334	0.4186
w/o RC	0.1619↓	0.2193	0.4643	0.4483
w/o Dropout	0.1658	0.2185	0.4553	0.4471
1 layer ($L = 1$)	0.1782	0.2122	0.4412	0.4238
3 layers ($L = 3$)	**0.1859**	**0.2262**	**0.4864**	**0.4661**
4 layers ($L = 4$)	**0.1834**	**0.2279**	**0.4898**	**0.4732**
1 head ($h = 1$)	**0.1853**	0.2187	0.4568	0.4402
4 heads ($h = 4$)	0.1830	**0.2245**	**0.4770**	**0.4520**
8 heads ($h = 8$)	0.1823	**0.2248**	0.4743	**0.4550**

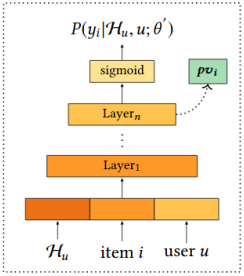
Personalized Re-ranking Model[10]



(a) One block of Transformer encoder.



(b) Architecture of PRM.



(c) The pre-trained model to generate $\boldsymbol{pv}_i, i = i_1, ..., i_n$.

[10] https://arxiv.org/pdf/1904.06813.pdf

PRM. Результаты

Init. List	Reranking	Yahoo Letor dataset.				
		Precision@5(%)	Precision@10(%)	MAP@5(%)	MAP@10(%)	MAP(%)
SVMRank	SVMRank	50.42	42.25	73.71	68.28	62.14
	LambdaMART	51.35	43.08	74.94	69.54	63.38
	DLCM	52.54	43.26	76.52	70.86	64.50
	PRM-BASE	**53.29**	**43.66**	**77.62**	**72.02**	**65.60**
LambdaMART	SVMRank	50.41	42.34	73.82	68.27	62.13
	LambdaMART	52.04	43.00	75.77	70.49	64.04
	DLCM	52.54	43.16	77.81	71.88	65.24
	PRM-BASE	**53.63**	**43.41**	**78.62**	**72.67**	**65.72**

Figure: Сравнение моделей на датасете Yahoo Letor.

Time-Aware User Embeddings as a Service[11]

Действия пользователя происходят неравномерно:

- $a_1, \ldots, a_T$ — последовательность действий пользователя, где $a \in A$ — конечное множество возможных действий пользователя
- каждому действию пользователя $a \in A$ сопоставляется обучаемый эмбеддинг $v \in \mathbb{R}^d$
- $t_1, \ldots, t_T$ — таймстемпы действий

Сделаем следующие преобразования:

$$\tau_j = \frac{t_j}{t_T}$$
$$\sigma_j = \sigma(\theta_j + \mu_j \tau_j)$$
$$\hat{v}_j = \sigma_j v_j,$$

где v_j — исходный вектор действия a_j.

[11] https://research.yahoo.com/publications/9272/time-aware-user-embeddings-service

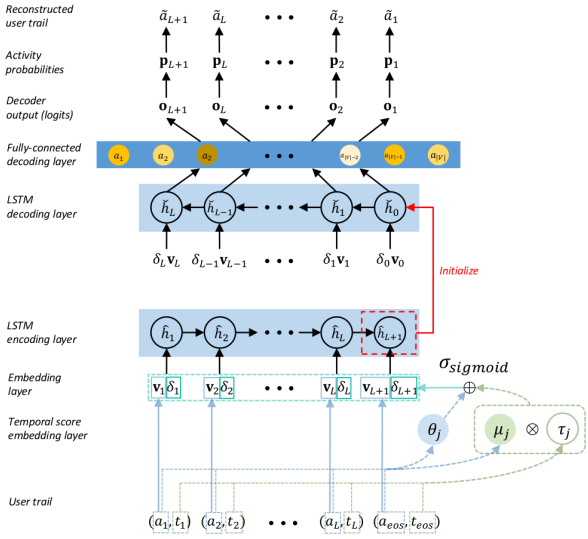
TASA. Архитектура

Обучение:

- кодируем исходную последовательность событий в вектор
- реконструируем исходную последовательность

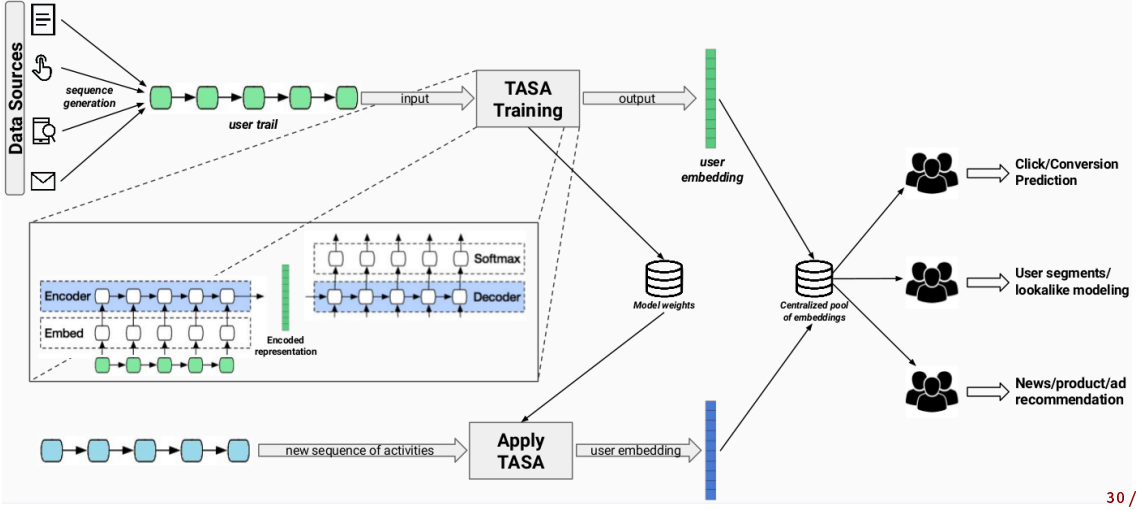
Model	Rec. Acc.	BLEU_n $n=1$	BLEU_n $n=2$	ROUGE_n $n=1$	ROUGE_n $n=2$	ROUGE_n $n=w$
AE	0.0136	0.0136	0.0085	0.0122	0.0082	0.0136
seq2seq	0.1235	0.2396	0.0709	0.2551	0.0742	0.2254
TA-seq2seq	0.1725	0.2664	0.0936	0.2807	0.0965	0.2527
ISA	0.1979	0.2535	0.0927	0.2851	0.0991	0.2464
TASA	0.5244	0.5500	0.3952	0.5691	0.4012	0.5441

Figure: Качество реконструкции на датасете RecSys 2015.



TASA. Результаты

Supervised Task	LR(AE)	LR(seq2seq)	LR(TA-seq2seq)	LR(ISA)	LR(TASA)	LR 1-hot	attRNN
RecSys 2015 Challenge	0.5555	0.6022	0.7523	0.7420	0.7563	0.7277	0.7591



BERT. Дистилляция[12]

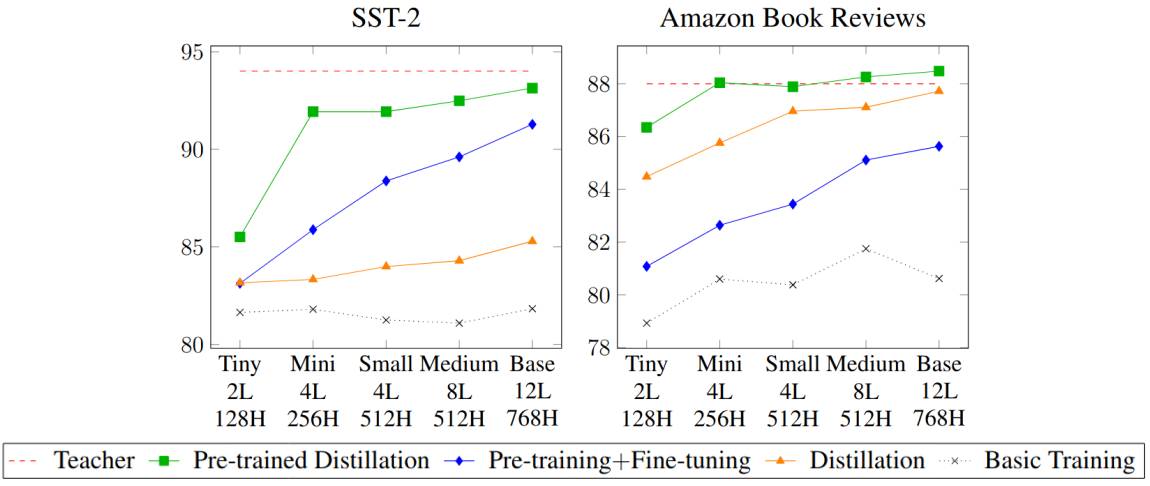
Проблема: хотим трансформеры в продакшне.

- Обучаем «учителя» — большую, тяжелую SOTA модель
- Размечаем большое количество неразмеченных данных
- Обучаем маленькую модель, «ученика», повторять за «учителем»
- функция ошибки — кросс-энтропия или MSE со сглаженными предсказаниями учителя в качестве истинных меток

Amazon Book Reviews — 50k размеченных примеров, 8 млн. неразмеченных.

[12] Не успели в прошлый раз.

Well-Read Students Learn Better, Turc et al[13]



[13] https://arxiv.org/pdf/1908.08962.pdf

Дистилляция

Другие варианты:

- **PKD**[14] — MSE между векторными представлениями
- **DistillBert** — на 60% быстрее, на 40% меньше, сохраняет 97% качества

Model	SST-2 (67k)	MRPC (3.7k)	QQP (364k)	MNLI-m (393k)	MNLI-mm (393k)	QNLI (105k)	RTE (2.5k)
$BERT_{12}$ (Google)	93.5	88.9/84.8	71.2/89.2	84.6	83.4	90.5	66.4
$BERT_{12}$ (Teacher)	94.3	89.2/85.2	70.9/89.0	83.7	82.8	90.4	69.1
$BERT_6$-FT	90.7	85.9/80.2	69.2/88.2	80.4	79.7	86.7	63.6
$BERT_6$-KD	91.5	**86.2/80.6**	70.1/88.8	80.2	79.8	88.3	64.7
$BERT_6$-PKD	**92.0**	85.0/79.9	**70.7/88.9**	**81.5**	**81.0**	**89.0**	**65.5**
$BERT_3$-FT	86.4	80.5/**72.6**	65.8/86.9	74.8	74.3	84.3	55.2
$BERT_3$-KD	86.9	79.5/71.1	67.3/87.6	75.4	74.8	84.0	56.2
$BERT_3$-PKD	**87.5**	**80.7**/72.5	**68.1/87.8**	**76.7**	**76.3**	**84.7**	**58.2**

Figure: Результаты на GLUE. FT — дообучение без дистилляции, KD — обычная дистилляция, PKD — дистилляция промежуточных выходов трансформера.

[14] Patient Knowledge Distillation for BERT Model Compression, Sun et. al