

## **Применение алгоритмов К-ближайших соседей в коллаборативных рекомендательных системах**

Отчёт о прохождении учебной (ознакомительной) практики

**Автор:**

Сергиенко Андрей

**Научный руководитель:**

старший преподаватель кафедры системного программирования  
Юрий Александрович Андреев

Санкт-Петербург, 2025

# Постановка задачи

## Цель работы

Исследовать эффективность различных алгоритмов К-ближайших соседей (KNN) для реализации коллаборативной рекомендательной системы.

## Задачи

- Изучить принципы коллаборативной фильтрации и KNN
- Реализовать и сравнить 4 метода поиска:
  - **Exact KNN** (scikit-learn) — эталонный метод
  - **Annoy** — случайные деревья
  - **FAISS** — кластеризация IVF
  - **HNSW** — иерархические графы
- Провести измерения производительности, точности и памяти
- Сформулировать рекомендации по применению

## Датасет MovieLens (32M)

- **Фильтрация данных:**
  - Пользователи:  $> 50$  оценок  $\rightarrow$  126 588 чел.
  - Фильмы:  $> 10$  оценок  $\rightarrow$  30 521 шт.
- **Матрица:**  $126\,588 \times 30\,521$  (99.24% разреженность)

## Снижение размерности через TruncatedSVD

- **Цель:** Переход от разреженного пространства к плотным эмбедингам
- **Размерность:** 128 латентных компонентов
- **Результат:** Объяснено 42.07% дисперсии, сжатие в 96 раз

$$R \approx U_k \Sigma_k V_k^T \text{ — сохраняем только главные компоненты}$$

## Обоснование выбора 128 компонент

Таблица: Зависимость метрик от размерности эмбедингов

Компоненты	Дисперсия	Размер (MB)	Время SVD (с)	Сжатие
32	31.83%	38.36	24.17	384.2×
64	36.53%	76.71	47.38	192.1×
128	42.07%	153.43	82.61	96.1×
256	49.31%	306.85	170.39	48.0×
512	58.75%	613.71	440.26	24.0×

- 32 → 64: прирост +4.70% при удвоении размера и времени
- 64 → 128: прирост +5.54% при удвоении размера, время ×1.74
- 128 → 256: прирост +7.24% при удвоении размера, время ×2.06
- 256 → 512: прирост +9.44% при удвоении размера, время ×2.58

Видно, что эффективность прироста качества на единицу ресурсов падает. Точка 128 компонент находится на "излом кривой где дальнейшее увеличение размерности даёт непропорционально малый выигрыш.

## Изолированные процессы для точности

### Orchestrator:

- Подготовка данных
- Генерация ground truth
- Запуск worker-ов
- Агрегация результатов

### Worker (subprocess):

- Чистая память
- Построение индекса
- Выполнение запросов
- Возврат метрик

## Метрики оценки

- **Производительность:** Время построения, время запроса, память (psutil)
- **Точность:** Recall@20, Precision@20 (сравнение с Exact KNN)
- **Тестирование:** 100 случайных запросов (seed=42)

## Композитная метрика Grid Search

$$\text{Score} = 0.4 \cdot \text{Recall@20} - 0.6 \cdot \ln(1 + \text{QueryTime}_{\text{ms}})$$

Баланс точности (40%) и скорости (60%) для production

Метод	Оптимальные параметры	Score
Annoy	n_trees=50	0.199
FAISS	nlist=1600	0.291
HNSW	ef_construction=200, M=16	<b>0.327</b>

HNSW показал наилучший компромисс при минимальных параметрах

## Результаты: Точность

Метод	Recall@20	Precision@20	Потеря точности
Exact KNN	1.000	1.000	—
HNSW	0.973	0.973	2.7%
FAISS	0.939	0.939	6.1%
Annoy	0.760	0.760	24.0%

### Ключевой вывод

**HNSW** достигает почти эталонной точности (97.3%), что критично для персональных рекомендаций, где каждый процент влияет на вовлечённость пользователей.

## Результаты: Производительность

Метод	Время запроса (user)	Ускорение
Exact KNN	139.59 мс	1×
Annoy	0.19 мс	734×
FAISS	0.18 мс	775×
HNSW	0.12 мс	<b>1163×</b>

### Время построения индекса (user-based)

- Exact KNN: 0.006 с (без структур)
- Annoy: 2.78 с
- HNSW: 7.27 с
- FAISS: 8.54 с

*Замедление построения окупается тысячекратным ускорением запросов*



## Результаты: Потребление памяти

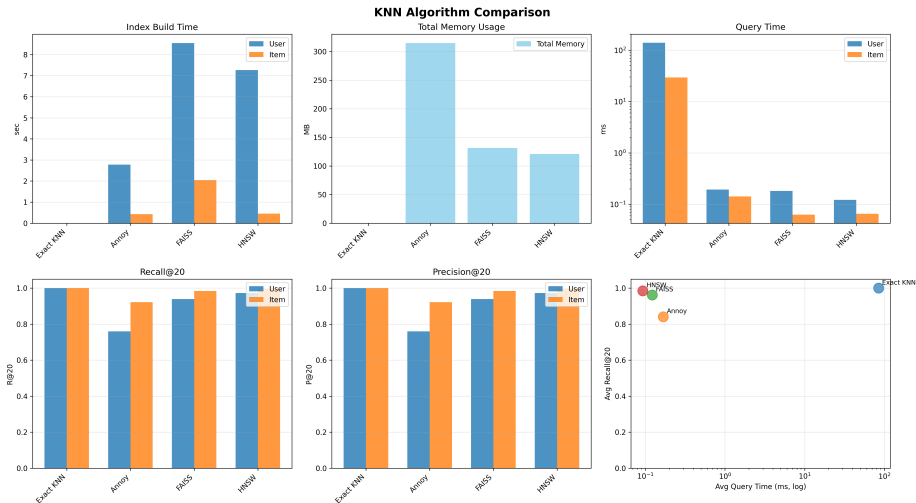
Метод	Peak Memory	Относительно
Exact KNN	0.00 МБ	(базовая линия)
HNSW	121.05 МБ	1.0×
FAISS	131.54 МБ	1.1×
Annoy	314.74 МБ	2.6×

### Баланс скорость—точность—память

**HNSW** лидирует по всем критериям:

- Максимальная скорость запросов (0.12 мс)
- Высочайшая точность среди ANN (97.3%)
- Минимальное потребление памяти (121 МБ)

# Итоговое сравнение



## Выводы и рекомендации

### 1. Exact KNN — Эталон для тестирования

Офлайн-аналитика, малые датасеты ( $< 10K$  объектов)

### 2. Annoy — Максимальная скорость

Когда допустима потеря 24% точности: предварительная фильтрация, low-priority рекомендации

### 3. FAISS — Масштабируемость + GPU

Большие системы ( $> 1M$  объектов), особенно с GPU-ускорением

### 4. HNSW — Production-стандарт

**Оптимальный выбор:** Почти эталонная точность (97.3%), максимальная скорость (1163× быстрее), минимальная память (121 МБ). Рекомендован для высоконагруженных систем реального времени.

# Спасибо за внимание!

Вопросы?

Код и результаты доступны в репозитории проекта

