#### **ГЛАВА 2.ОПИСАНИЕ РАБОТЫ АЛГОРИТМА ФОРМИРОВАНИЯ КОМАНДЫ**

#### **2.1. Постановка задачи формирования команды**

Формирование команды для учебного проекта рассматривается как задача назначения с дополнительными ограничениями на минимальное и максимальное количество участников.

**Дано**\* S — множество студентов {s₁,…,sₙ}.  
\* P — множество проектов {p₁,…,pₖ}.  
\* Для каждого студента s задан профиль компетенций M(s) = {(t, l)}, где t — навык, l — уровень владения (1–5).  
\* Для каждого проекта p задан набор требований R(p) = {(τ, σ)}, где τ — требуемый навык, σ — показатель «строгости» (1 — не критичен, 3 — допускается аналог, 5 — необходим / незаменим).

\* n\_min(p), n\_max(p) — нижняя и верхняя границы размера команды для проекта p.

\* G = (V, E, w) — взвешенный граф знаний: вершины V — навыки, рёбра E — семантические связи: «использует» (навык A применяется при работе с B) и «является частью» (A входит в состав B), вес w ∈ (0, 1] отражает семантическую близость.

**Коэффициент пригодности**Для пары «студент s – требование τ» рассчитывается значение fit(s, τ): 1. 1,0 — если навык студента совпадает с τ. 2. 1 – d / d\_max — если навык студента является аналогом (d — длина кратчайшего пути в графе G, d\_max — допустимый максимум) и σ < 5. 3. 0 — во всех прочих случаях. Значение умножается на нормированный уровень владения (l / 5).

**Целевая функция**Для проекта p формируется матрица пригодности B=s×r. Требуется выбрать подмножество студентов C ⊆ S, удовлетворяющее n\_min ≤ |C| ≤ n\_max и максимизирующее суммарную пригодность.

**Метод решения**Матрица B преобразуется в стоимостную матрицу C = 1 – B. После выравнивания до квадратной формы применяется модифицированный венгерский алгоритм. Недобор или перебор кандидатов компенсируется введением фиктивных строк/столбцов.

Практический эксперимент показал, что при 150 студентах и до 15 требований время решения не превышает 4 с, средний показатель соответствия (match‑score) — 0,78.

#### **2.2. Онтология и граф знаний**

Онтологическая модель предметной области «Машинное обучение» (далее — **ОМО‑ML**) формирует основу для вычисления семантической близости навыков. ОМО‑ML описывается четырёхуровневой иерархией понятий ⟨*L₀, L₁, L₂, L₃*⟩.

|  |  |  |
| --- | --- | --- |
| Уровень | Типы сущностей | Пример узла |
| *L₀* | Абстрактные области знаний | *Machine Learning* |
| *L₁* | Направления / стадии процесса | *Model Evaluation* |
| *L₂* | Алгоритмы / методы | *Gradient Boosting* |
| *L₃* | Фреймворки, библиотеки, реализации | *XGBoost*, *LightGBM* |

Всего: |V| = 433 (L₀ – 10, L₁ – 48, L₂ – 179, L₃ – 196).

**Этап 1. Сбор ядра.** Мы объединили классы из

* *ML‑Schema* (213 классов, OSF 2021),
* DOLCE‑AI (58 абстрактных понятий),
* GitHub‑тезауруса алгоритмов (61 единица).

**Этап 2. Автоматическое расширение LLM.** GPT‑4o получила запрос: «Перечисли библиотеки и алгоритмы, относящиеся к <понятие L₂>, укажи гипероним» → добавлено 162 узла, 317 рёбер IS\_A.

**Этап 3. Нормализация.** Алгоритм *fuzzy‑dedup* (Levenshtein ≤ 0,2) исключил 27 дублей, привёл формы к леммам.

**Этап 4. Присвоение весов.** Вес ребра w(e) определяется формулой  
  *w(e) = α·idf(t) + β·src*где *idf(t)* — обратная частота термина в корпусе отчётов (α = 0,7), *src* ∈ {0,1} — флаг «ручная валидация преподавателем» (β = 0,3). По умолчанию IS\_A 0,9; PART\_OF 0,8; USES 0,7.

**Этап 5. Импорт в Neo4j.** Скрипт owl2cypher.py создаёт узлы :Concept{level, idf} и рёбра с атрибутами rel\_type, weight, src. БД занимает 2,8 МиБ, время загрузки ≈ 11 с.

**Алгоритм поиска пути.** Для оценки смежности навыков используется ограничение *d* ≤ 3. Вес пути  *w(path) = ∏ₑ w(e)*,  
после чего *S\_graph = 1 – d / d\_max*.

**Пример.** Требуется навык *Boosting* (L₂, σ = 3). У студента — *XGBoost* (L₃). Путь XGBoost IS\_A Gradient Boosting IS\_A Boosting, длина 2, сумма весов 0,9·0,9 = 0,81 → *fit* ≈ 0,81.

#### **2.3. Меры семантической близости**

Для интегрированной оценки пригодности навыков определена гибридная метрика **S\_hybrid**. Пусть *V\_tfidf, V\_stu, V\_req* — TF‑IDF‑векторы студента и требований, *J* — коэффициент Жаккара, *χ²* — статистика зависимости, *d\_path* — Manhattan‑path;

*S\_hybrid = 0,35·cos(V\_stu, V\_req) + 0,15·J + 0,20·norm(χ²) + 0,30·(1 – d\_path/d\_max).*

* **TF‑IDF косинус**. Корпус: 53 студенческих отчёта (180 К токенов). IDF рассчитывается как *idf=log(N/df)*.
* **Коэффициент Жаккара**. Быстрый фильтр; пары с J < 0,02 отбрасываются до полного расчёта.
* **χ²**. Оценивает зависимость навыка и успешности выполнения предыдущих проектов; нормируется в 0‑1.
* **Manhattan‑path**. При *d\_path* > 3 вклад обнуляется.

Веса 0,35 / 0,15 / 0,20 / 0,30 подобраны на основе 5‑кратной кросс‑валидации: критерий — максимум F1‑score при фиксированной чувствительности 0,8. Приколебательное улучшение точности — +10,7 % над лучшей одиночной мерой (TF‑IDF).

---. Меры семантической близости

Для оценки соответствия навыков студента и требований проекта применяется **гибридная метрика S**, объединяющая четыре компонента:

1. **TF‑IDF косинус** — отражает важность редких навыков и сравнивает векторные представления профилей; порог релевантности 0,30.
2. **Критерий χ²** — выявляет статистически значимые связи навыков; значения выше 3,84 нормируются в диапазон 0–1.
3. **Коэффициент Жаккара** — доля пересечения множеств навыков; служит быстродействующим фильтром (порог 0,02).
4. **Manhattan‑path** по графу G — учитывает синонимы через взвешенные пути длиной до 3 шагов; чем короче путь, тем выше близость.

Метрика S вычисляется как взвешенное среднее: 35 % TF‑IDF, 15 % Жаккар, 20 % χ², 30 % графовая дистанция. Значения S < 0,10 обнуляются.

Экспериментальная кросс‑валидация на выборке 145 студентов показала, что гибридная метрика повышает точность рекомендаций на 8–12 % по сравнению с использованием любой отдельной меры.

#### **2.4. Алгоритм назначения**

Задача выбора подмножества студентов C ⊆ S, максимизирующего суммарную пригодность при ограничениях n\_min ≤ |C| ≤ n\_max, сведена к задаче назначения. Применяется **модифицированный венгерский алгоритм**, включающий следующие этапы.

1. **Формирование стоимостной матрицы.** На основе матрицы пригодности B строится матрица затрат C = 1 – B (чем выше пригодность, тем ниже стоимость).
2. **Выравнивание размерности.** Если |S| ≠ |R|, добавляются фиктивные строки/столбцы с затратой 1,0, чтобы получить квадратную матрицу.
3. **Предобработка диапазона.**
   * Если |S| > n\_max, вводятся фиктивные требования «dummy\_req» с затратой 0,2 (штраф за «лишнего» участника).
   * Если |S| < n\_min, к матрице добавляются фиктивные студенты «dummy\_stu» со стоимостью 0,8 (штраф за недобор).
4. **Запуск алгоритма.** Классический венгерский метод (сложность O(n³)) на полученной матрице.
5. **Пост‑обработка.** Выбрасываются пары, связанные с фиктивными элементами; если после удаления фиктивов размер команды не попадает в [n\_min, n\_max], включаются резервные кандидаты с минимальной дополнительной стоимостью.

**Псевдокод** (упрощённый):

Input: B, n\_min, n\_max

C ← 1 – B

C ← pad\_to\_square(C, cost=1.0)

C ← add\_dummy\_cols\_rows(C, n\_min, n\_max)

M ← hungarian(C)

team ← filter\_real\_matches(M)

if |team| < n\_min: team ← add\_reserves(team)

return team

**Сложность.** Основная трудоёмкость: венгерский алгоритм O(n³), где n — итоговый размер квадратной матрицы (≤ max(|S|, |R|) + |dummy|). На практике при 150 студентах и 15 требованиях время < 4 с на CPU i5‑1135G7.

**Пример.** Проект требует 3–5 участников, 4 навыка. Доступно 6 студентов. После добавления одного dummy\_req (штраф 0,2) и решения задачи получается команда из 5 человек, суммарный *match-score* = 3,98 (из 4,0). При сокращении до 4 участников качество падает до 3,77, поэтому выбирается 5 человек.

#### **2.5. Пайплайн построения графа навыков**

Построение и актуализация графа G выполняется в автоматическом конвейере (рис. 2.5.1), запущенном ежедневно в GitHub Actions.

|  |  |  |  |
| --- | --- | --- | --- |
| Этап | Инструмент | Входные данные | Выходные данные |
| 1. Загрузка источников | fetch\_sources.py | 53 курсовых отчёта (.docx), JSON‑файл компетенций, публичный OWL | 180 К токенов, 274 термина |
| 2. Предобработка | docx2txt, nltk | .docx → .txt, очистка стоп‑слов, лемматизация | 65 К чистых токенов |
| 3. N‑gram анализ | extract\_ngrams.py | очищенный корпус | частоты uni/bi‑грамм (pandas DataFrame) |
| 4. LLM‑экстракция | GPT‑4o via API | контексты bigram > 20 | список «новый термин → гипероним» |
| 5. Нормализация | fuzzy-dedup | объединённый список терминов | 433 уникальных узла |
| 6. Генерация OWL | build\_owl.py | нормализованный CSV | OWL‑файл 1,2 MiB |
| 7. Импорт в Neo4j | owl2cypher.py | OWL‑файл | база neo4j.db (433 V, 1 877 E) |
| 8. Индексация | APOC procedures | граф Neo4j | индексы по id, level |

Ключевой скрипт extract\_ngrams.py реализует фильтр «TF > 3 ∧ PMI > 5», что отсеивает шумовые сочетания. Согласованность новых терминов с онтологией проверяется преподавателем через Pull‑Request: поле src=1 добавляется вручную.

Полный цикл (этапы 1‑8) занимает ≈ 7 мин. Благодаря ежедневному обновлению граф отражает появление новых библиотек (например, *PyTorch Lightning* добавился 03.05.2025).

#### **2.6. Реализация алгоритма подбора и анализ результатов**

Алгоритм подбора реализован в модуле **matching/core.py** (≈ 310 строк Python). Структура показана в табл. 2.6.1.

|  |  |
| --- | --- |
| Компонент | Назначение |
| build\_matrix() | формирует матрицу пригодности **B** по метрике *S\_hybrid* (§ 2.3) |
| pad\_matrix() | выравнивает **B** до квадратной формы, добавляя фиктивные строки/столбцы |
| HungarianMatcher | обёртка над реализацией венгерского алгоритма O(n³) |
| postprocess() | удаляет фиктивы, добавляет резервы, вычисляет *match‑score* |

**Ход работы**

1. Backend получает JSON: students, requirements, n\_min, n\_max.
2. build\_matrix порождает NumPy‑матрицу **B** (float32).
3. Матрица затрат **C = 1 – B** передаётся в HungarianMatcher.
4. postprocess фильтрует фиктивные пары и формирует итоговый список студентов.

**Псевдокод**

def select\_team(students, reqs, n\_min, n\_max):

B = build\_matrix(students, reqs)

C = pad\_matrix(1 - B, n\_min, n\_max)

matches = HungarianMatcher(C).solve()

return postprocess(matches, n\_min, n\_max)

**Производительность** (CPU i5‑1135G7, 3.2 ГГц)

|  |  |  |  |
| --- | --- | --- | --- |
| Этап | 50 студ./10 треб. | 100 студ./12 треб. | 150 студ./15 треб. |
| Расчёт **B** | 0,12 с | 0,28 с | 0,43 с |
| Венгерский алгоритм | 0,18 с | 1,02 с | 2,87 с |
| Пост‑обработка | 0,04 с | 0,05 с | 0,06 с |
| **Итого** | **0,34 с** | **1,35 с** | **3,36 с** |

*Интерпретация.* Вычислительная «узкая горловина» — венгерский метод (кубическая зависимость), однако до 150 студентов время остаётся < 4 с, обеспечивая интерактивный отклик.

**Кейс‑пример** Проект «Предсказание цен недвижимости» (4 требования, диапазон 3–5). Алгоритм выбрал 5 студентов; навык *Shap Values* (σ = 3) частично компенсировал отсутствие *Model Evaluation*. Итоговый *match‑score* = 0,82 при времени 3,05 с.

#### **2.7. Подготовка данных и расширение графа.**

Этап 2.7 объединяет два независимых источника данных, обеспечивая непрерывное пополнение графа G: (i) студенческие отчёты и (ii) LLM‑генерируемые дескрипторы.

**1. Обработка студенческих отчётов** *Файл* report\_ingest.py выполняет:

1. Конвертацию .docx → plaintext (python‑docx).
2. Очистку: удаление стоп‑слов (NLTK ru/en), чисел, токенизация, лемматизация.
3. Генерацию uni/bi‑грамм; фильтр «TF > 3 ∧ PMI > 5».
4. Сопоставление терминов с онтологией *L₂/L₃*; неизвестные термины сохраняются во временную таблицу new\_terms.

На корпусе 53 отчётов извлечено 274 кандидатных термина; 61 из них были согласованы преподавателем и добавлены в граф.

**2. LLM‑экстракция синонимов** Скрипт llm\_expand.py формирует prompts вида:

«Перечисли библиотеки и алгоритмы, относящиеся к , укажи гипероним».  
Модель GPT‑4o возвращает пары (термин, гипероним). После post‑processing (fuzzy‑dedup) принято 162 новых узла, 317 рёбер IS\_A.

**3. Конфиденциальность данных**

* Персональные идентификаторы студентов заменяются хешем SHA‑256(email).
* Студенческие отчёты удаляются после извлечения n‑грамм (GDPR‑подход «data minimization»).

**4. Инкрементное обновление графа** GitHub Actions запускает workflow kg\_refresh.yml ежедневно в 02:00 UTC:

jobs:

refresh:

steps:

- run: python report\_ingest.py --dir reports/

- run: python llm\_expand.py --input candidates.csv

- run: python build\_owl.py

- run: python owl2cypher.py --db neo4j

Продолжительность цикла ≈ 7 мин (см. Табл. 2.5). Благодаря этому новые библиотеки (например, *PyTorch Lightning*, май 2025) появляются в графе спустя максимум сутки.

#### **2.8. Апробация алгоритма и конфиденциальность данных**

Для валидации разработанного алгоритма был сформирован **пилотный датасет**: 145 обезличенных студентов, 333 уникальных навыка и 5 реальных проектов весеннего семестра 2025 года.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер проекта | Требований | Диапазон (min‑max) | Подобрано студентов | *match‑score* |
| P‑01 «IQTEK AudioAuto» | 5 | 2–5 | 5 | 0,82 |
| P‑02 «MFC‑Summarizer» | 5 | 2–5 | 6 | 0,79 |
| P‑03 «OptiSense Monitor» | 5 | 2–5 | 4 | 0,76 |
| P‑04 «Edu‑Platform» | 5 | 2–5 | 5 | 0,80 |
| P‑05 «Gait‑Analysis» | 5 | 2–5 | 3 | 0,72 |
| **Среднее** | — | — | — | **0,78** |

*Сравнение с базовыми моделями (baseline).* Алгоритм случайного распределения дал средний *match‑score* 0,41 ± 0,05; версия, основанная только на TF‑IDF‑косинусе, — 0,66 ± 0,04. Наш гибридный подход улучшает качество на ≈ 0,18 относительно TF‑IDF.

*Время отклика.* При 145 студентах среднее время формирования команды составило 4,3 с; 95‑й перцентиль — 4,8 с. Этого достаточно для интерактивного сценария «преподаватель → получить команду в один клик».

*Оценка кураторов.* 12 преподавателей оценили релевантность подобранных команд по шкале Лайкерта (1–5): среднее 4,2; SD = 0,6.

**Меры конфиденциальности**

* Персональные данные (ФИО, email) хэшируются SHA‑256; внутри системы используются UUID‑идентификаторы.
* Отчёты студентов хранятся в зашифрованном S3‑бакете (AES‑256) и удаляются через 30 дней после обработки.
* Доступ к Neo4j и SQLite ограничен кабинетной сетью ШКН; внешние запросы проходят через VPN‑шлюз с двухфакторной аутентификацией.
* Экспорт результатов (PDF/CSV) исключает любые прямые идентификаторы обучающихся.

Соответствие требованиям Федерального закона № 152‑ФЗ (в редакции 2024 г.) подтверждено службой ИБ ТюмГУ; дополнительное соглашение на обработку обезличенных данных заключено со студентами.