# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Переводчик с русского языка на космический язык Lincos

### Введение

В практической части нашей работы мы представляем переводчик с русского языка на космический язык Lincos. Переводчик работает исключительно с математическими выражениями по причине состояние языка LINCOS: Разработчик языка, Ганс Фройденталь, детально проработал только математический раздел. Такие модули, как «Время», «Поведение», «Пространство, движение, масса», находятся на стадии концепции и не имеют завершённой формальной спецификации, таким образом без неподготовленных разделов корректное обучение машины представляет собой очень трудную задачу.

### Основные характеристики датасета

Датасет для обучения модели был сформирован автоматически с использованием программной генерации на языке Python. Основой для генерации послужили несколько тематических словарей, содержащих ключевые понятия, логические конструкции, математические операторы и синтаксические шаблоны, характерные для языка Lincos.

Общий объём датасета составляет 3 229 примеров. Из них:

- ✓ 2 583 примера используются для обучения модели,
- ✓ 646 примеров - для тестирования.

Распределение по категориям:

- ✓ Логика: 1 500 примеров,
- ✓ Математика: 1 231 пример,
- ✓ Сравнения: 498 примеров.

### Обоснование выбора метода для обучения машины

Представьте, что нужно перевести предложение с одного языка на другой, например, на искусственный язык Lincos. Для этой задачи хорошо подходит архитектура «от последовательности к последовательности» (seq2seq).

В ней работают две нейросети:

- ✓ Кодировщик: Он читает исходное предложение и упаковывает его смысл в специальный цифровой «слепок» - вектор.
- ✓ Декодировщик: Берет этот «слепок» и, шаг за шагом, собирает из него перевод, каждый раз помня, что уже «написал» на предыдущих шагах.

Почему другие методы не подходят:

- ✓ Простые словари и жесткие правила не подходят - они не понимают контекст и тонкости смысла, а для Lincos нет и полного набора лингвистических правил.
- ✓ Более современные и сложные модели (например, трансформеры) действительно мощнее, но им нужно очень много примеров для обучения. В нашем случае параллельных текстов на Lincos просто недостаточно.

Таким образом seq2seq - это оптимальный выбор. Он достаточно гибкий, чтобы улавливать смысл, и при этом может работать там, где данных мало, а полной теории перевода между языками еще не существует.

**Каким образом работает обучение модели**

1. Подготовка данных

Цель: превратить пары текстов («русский → Lincos») в числовые векторы, которые понимает нейросеть.

Создаются два словаря (Vocabulary): один для русского языка, другой для Lincos. Все уникальные слова и символы (включая логические: ∀, →, ∈ и т.д.) нумеруются. Добавляются специальные токены:

<SOS> — начало предложения,

<EOS> — конец предложения,

<PAD> — заполнение до фиксированной длины,

<UNK> — неизвестное слово.

Каждое предложение превращается в список чисел (numericalize), обрамлённый <SOS> и <EOS>, и дополненный <PAD>, чтобы длина всех предложений была одинаковой (max_length = 50). Данные упаковываются в объект TranslationDataset, который DataLoader разбивает на батчи (по 32 пары в каждом).

2. Архитектура модели

Модель состоит из двух частей:

Encoder (кодировщик):

- ✓ Принимает последовательность чисел (русское предложение).
- ✓ Преобразует каждое число в вектор (embedding).
- ✓ Пропускает последовательность векторов через двухслойную LSTM.
- ✓ На выходе - скрытые состояния (hidden, cell), которые содержат «смысл» всего предложения.

Decoder (декодировщик)

- ✓ Берёт скрытые состояния от encoder'a как начальный контекст.
- ✓ Начинает генерацию с токена <SOS>.
- ✓ На каждом шаге:

Превращает текущее слово в вектор,

Пропускает его через ту же LSTM,

Получает распределение вероятностей по всем словам Lincos-словаря,

Выбирает наиболее вероятное слово

3. Процесс обучения

В encoder подается русское предложение, после его обработки на выходе получаем скрытое состояние (hidden, cell). Далее подаем Lincos-перевод в decoder.

Затем модель пытается воспроизвести перевод, она начинает говорить перевод по одному слову за раз. На старте она говорит <SOS> («начало»), а дальше - угадывает следующие слова.

Чтобы не запутаться на ранних этапах, ей помогают, в 50% случаев вместо того, чтобы использовать своё предыдущее (возможно, ошибочное) слово, ей подсказывают правильное слово из настоящего перевода. Это называется «учитель ведёт за руку» (teacher forcing). Благодаря этому модель учится быстрее и не «сбивается с пути».

После того как модель выдала весь перевод, её ответ сравнивается с правильным: подсчитывается, насколько она ошиблась. Модель

«подкручивает» свои внутренние настройки: она смотрит, где ошиблась, и немного меняет свои веса, чтобы в следующий раз ошибиться меньше.

Так повторяется для всех фраз в батче, а потом - для всех батчей в эпохе. В конце эпохи считается средняя ошибка: насколько хорошо модель переводит в целом.

4. Валидация

То же самое, что и в п. 3, но без teacher forcing (teacher_forcing_ratio=0). Модель генерирует перевод полностью самостоятельно, используя только свои предыдущие предсказания. Потеря на тестовых данных показывает, насколько модель обобщает, а не просто запоминает.