Очистка данных для тренировки моделей машинного перевода

Обработка параллельных корпусов

Задача

Параллельные корпуса

- Нет проблемы сбора данных
- Большие объемы (годы работы), 60-150k пар
- Накапливается мусор
- Невозможно чистить вручную

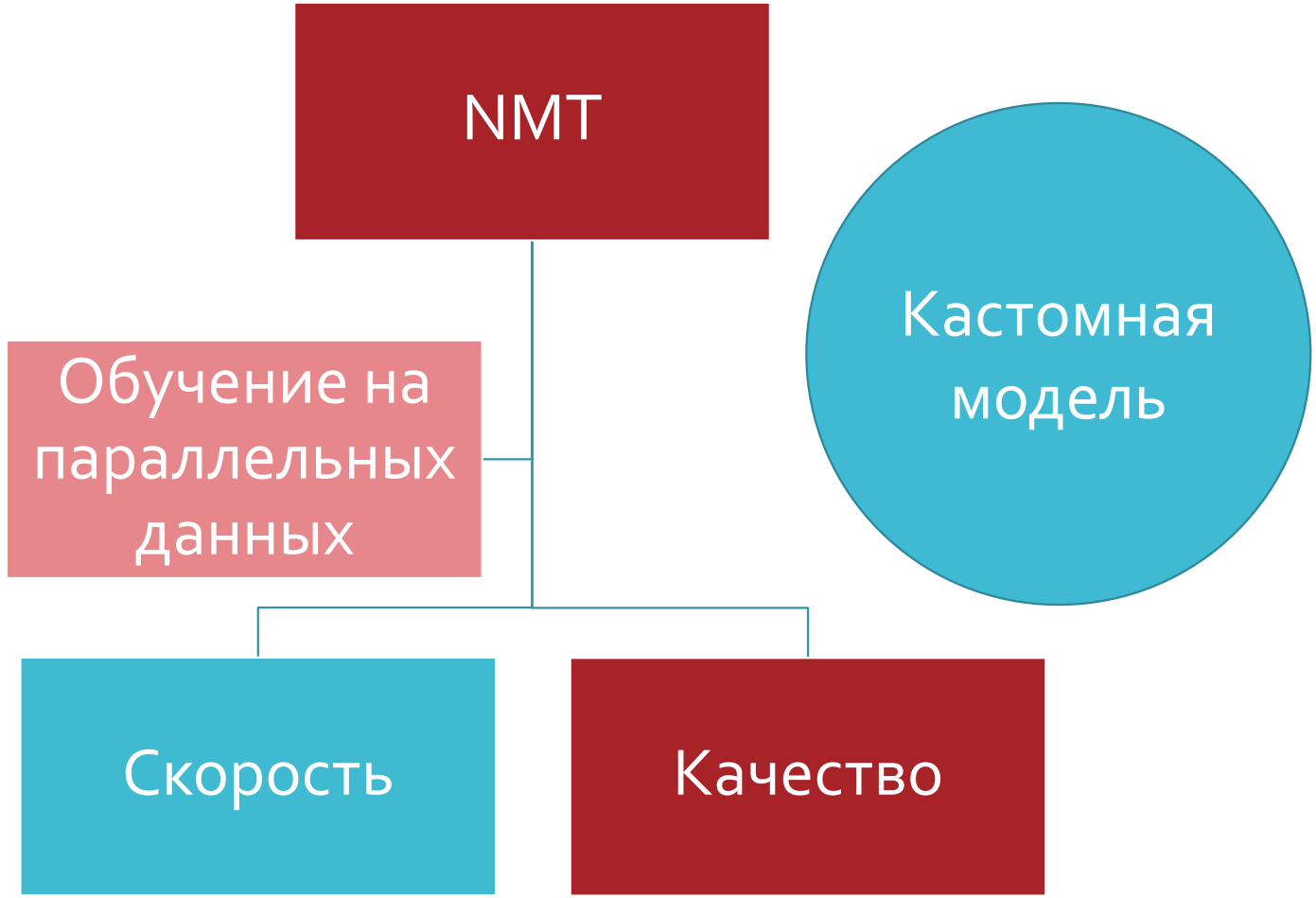
Задача

Идея

- Существующие инструменты не дают гибкости либо не рассчитаны на большие объемы данных
  - Только полные дубликаты
  - Недостаточно опций
- Возможность дальнейшего масштабирования для решения более сложных задач:
  - Более глубокая очистка данных для NMT
  - Извлечение терминологии
  - Составление двуязычных глоссариев

Задача

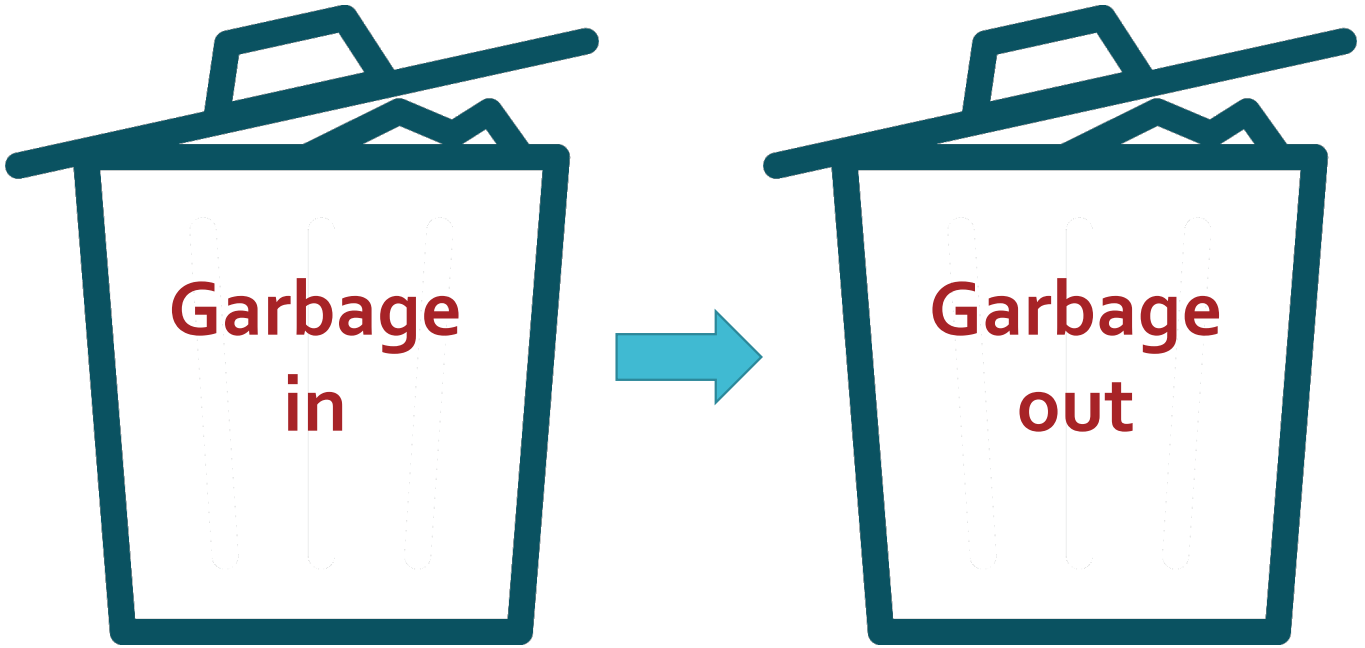
Нейронный машинный перевод



Задача

Обучение на параллельном корпусе

Quality over quantity



Задача

Очистка данных

1. **Удаление дубликатов**, в том числе неполных:
   - Различия в датах
   - Номера телефонов в разных форматах
   - Наличие разных ссылок
   - Разные значения полей
   - Лишние символы (пробелы, мягкие переносы)

Задача

Очистка данных

2. **Удаление мусора**:
   - Сегменты, состоящие из символов или цифр
   - Чрезмерно длинные предложения
   - Сегменты на языке, отличном от целевого

Задача

Очистка данных

3. **Устранение ошибок ввода**:
   - Опечатки в исходном тексте, которые приводят к появлению дубликата
   - Опечатки в переводе

Задача

Очистка данных

4. **Анонимизация**:
   - Имена (NER)
   - Номера телефонов
   - Адреса электронной почты
   - Другая идентифицирующая информация

Задача

Очистка данных

5. **Устранение несоответствий**:
   - Несоответствия в параллельных корпусах (поиск коллокаций и кандидатов на их перевод)

Подход

Этапы

1. **Анализ исходных параллельных корпусов**

Формат TMX — по сути XML.

```
<tu creationdate="20161221T125309Z" creationid="SONY-S\Svetlana"
changedate="20161221T130023Z" changeid="SONY-S\Svetlana" lastusagedate="
20161221T130023Z">
  <prop type="x-LastUsedBy">SONY-S\Svetlana</prop>
  <prop type="x-Context">8141944612297908607, 15989695196343197l6</prop>
  <prop type="x-ContextContent">FPGA-based, specifically: |  | На базе
  FPGA, а именно: | </prop>
  <prop type="x-Origin">TM</prop>
  <prop type="x-ConfirmationLevel">Translated</prop>
  <tuv xml:lang="en-US">
    <seg>x86/i64-based processor, specifically:</seg>
  </tuv>
  <tuv xml:lang="ru-RU">
    <seg>На базе процессора x86/i64, а именно:</seg>
  </tuv>
</tu>
```

Подход

Этапы

2. **Общий подход к решению задачи**

- Регулярные выражения для поиска данных
- Предварительная обработка:
  - ✓ Нижний регистр
  - ✓ Пунктуация (отдельная пунктуация в числовых данных)
  - ✓ Удаление незначащего текста в начале и конце строк
  - ✓ Замена незначащих для сравнения данных на символы
- Сравнение > построение индекса (словарь), множество

Подход

Этапы

3. **Взаимодействие с пользователем**

- Создание трех файлов:
  - Файл с отфильтрованной памятью
  - Файл с удаленными сегментами
  - Файл отчета
- Вывод прогресса для пользователя в консоль
- Запуск пользователем скрипта в терминале с указанием пути к файлу и параметров

Подход

Этапы

4. **Разработка**

- Функциональный метод
  - ✓ Регулярные выражения на максималках
  - ✓ Функция для предварительной обработки текста
  - ✓ Функция для замены незначащего текста

Код можно посмотреть тут

Подход

Этапы

4. Разработка

Предварительная обработка текста

```
import re
```

```
# Регулярные выражения для поиска нужных фрагментов текста, таких как пунктуация,
# номера телефонов, адреса электронной почты и URL в разных форматах.
# Регулярные выражения разбиты на логические части и присвоены переменным,
# которые позволяют их использовать в разных комбинациях,
# в том числе для составления сложных паттернов и объектов для выполнения с ними операций далее
# (методов модуля re)
```

```
FRAG_PUNCT_OUTER = r'\[\]*()<>/?!+&€£$%#“”":;`*=|\\'
FRAG_PUNCT_INNER = r'.,'
RE_PUNCT = re.compile(r'([{punct_outer}])'.format(punct_outer=FRAG_PUNCT_OUTER))
RE_PUNCT_INNER = re.compile(r'([{punct_inner}])'.format(punct_inner=FRAG_PUNCT_INNER))
RE_PUNCT_ALL = re.compile(r'([{punct_inner}{punct_outer}])'.format(punct_inner=FRAG_PUNCT_INNER, punct_outer=FRAG_PUNCT_OUTER))
RE_NUMBERS = re.compile(r'\b[0-9]+(?:(?:\.[0-9]+)+|(?:,[0-9]+))?\b') # Ловит цифры, числа, номера версий.
RE_PHONE = re.compile(r'\+(?:\s*(?:-\s*)?\001){3,}') # Номера телефонов в международном формате с дефисом и без.
FRAG_URL_PATH_COMPONENT = r'[a-zA-Z0-9_.-]+'
FRAG_DOMAIN_NAME = r'(?:[a-zA-Z][a-zA-Z0-9-]*\.)+[a-z]{2,}'
FRAG_URL_FREEFORM = r'[/a-zA-Z0-9_.-]+'
FRAG_URL_QUERYPARAM = r'[a-zA-Z0-9_]+={}'.format(FRAG_URL_FREEFORM)
FRAG_NON_TEXT = r'(?:\001|\002|[{punct_inner}{punct_outer}]+)'.format(punct_inner=FRAG_PUNCT_INNER, punct_outer=FRAG_PUNCT_OUTER)
RE_URL_OR_EMAIL = re.compile(r'\b(?:(http)s?://{dn}(?:/(?:(?:{pathcomp}/)*{pathcomp})?(?:\?{queryparam}(?:\&{queryparam})*)?(?:#{freeform
    dn=FRAG_DOMAIN_NAME, pathcomp=FRAG_URL_PATH_COMPONENT, queryparam=FRAG_URL_QUERYPARAM, freeform=FRAG_URL_FREEFORM))
RE_LEAD_PUNCT = re.compile(r'^\s*(?:{non_text}\s+)*'.format(non_text=FRAG_NON_TEXT))
RE_TRAIL_PUNCT = re.compile(r'(?:\s+{non_text})*\s*$'.format(non_text=FRAG_NON_TEXT))
RE_ALL_PUNCT = re.compile(r'^\s*{non_text}\s*$'.format(non_text=FRAG_NON_TEXT))
```

Подход

Этапы

4. Разработка

```
def mangle_text_inner(s):
    s = s.strip().lower()

    # Отбивка пунктуации
    s = RE_PUNCT.sub(r' \1 ', s)

    # Замена чисел, включая номера версий, на заглушку
    s = RE_NUMBERS.sub(' \001 ', s)

    # Отбивка пунктуации слева от числовой замены
    s = RE_PUNCT_INNER.sub(r' \1 ', s)

    # Постобработка: замена номеров телефона на заглушку
    s = RE_PHONE.sub(' \002 ', s)

    # Постобработка: удаление чисел и пунктуации в начале и конце строки
    s = RE_LEAD_PUNCT.sub('', RE_TRAIL_PUNCT.sub('', RE_ALL_PUNCT.sub('', s)))

    return s
```

Подход

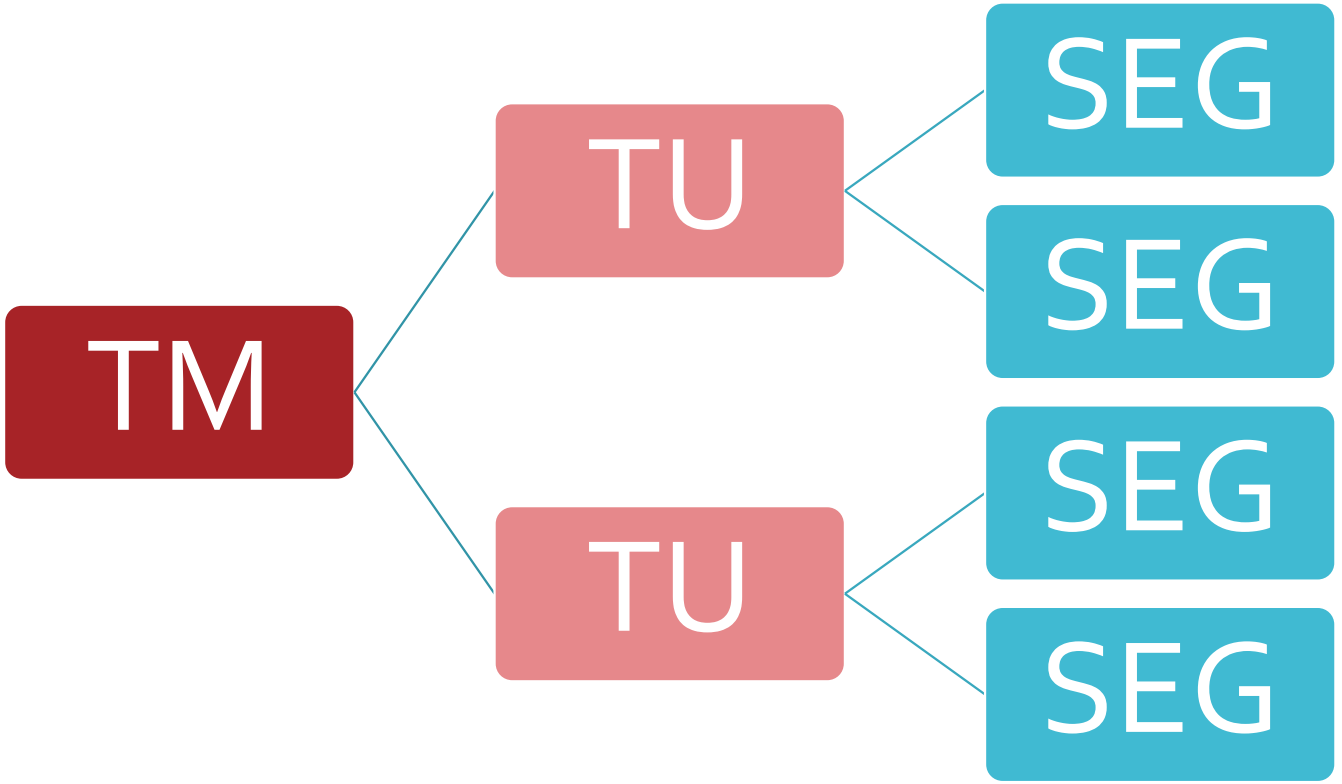
Этапы

4. Разработка

```
def mangle_text(s):

    fragments = []

    # Отдельно ищем URL и адреса электронной почты или URL во всех возможных форматах
    while True:
        m = RE_URL_OR_EMAIL.search(s)

        if m is None:
            break

        # Когда находим, обрабатываем текст до и после как фрагменты, после чего соединяем обратно в строку.
        text_before = s[:m.start()]
        if text_before.strip():
            fragments.append(mangle_text_inner(text_before))

        # Заменяем на разные заглушки, в зависимости от типа совпадения.
        if m.group(2) == 'www':
            prefix = '\003'
        elif m.group(1) == 'http':
            prefix = '\004'
        else:
            prefix = '\005'

        # Присоединяем заглушку ко всему найденному паттерну
        fragments.append(prefix + m.group(0))

        # Проверяем строку дальше
        s = s[m.end():]

    # Если строка содержит какие-то символы кроме пробельных, обрабатываем ее и добавляем к фрагментам
    if s.strip():
        fragments.append(mangle_text_inner(s))

    s = ' '.join(fragments)

    # Разбиваем на слова и снова соединяем, чтобы избавиться от лишних пробелов
    words = s.split()
    s = ' '.join(words)

    return s
```

Подход

4. **Разработка (с куратором)**
   - Объектно-ориентированный метод



Подход

4. Разработка (с куратором)

Сложности

Основные сложности

1. Комплексная архитектура, много связей
2. Продумывание и обработка всевозможных исключений
3. Запись в XML
4. Запись в HTML
5. Отладка в ООП
   - Специальный параметр отладки
   - Запись особого тега в файл для проверки

Сложности

Код

1. Запуск кода

```
SvetlanasMacbook-3:Code Svetlana$ python3 cleaner.py -r -v basler.tmx
Loading TMX...
Filtering...
* Deleted (Duplicate): 5962
* Deleted (No Text Content): 128
* Filtered: 60126
Writing results...
Done.
SvetlanasMacbook-3:Code Svetlana$
```

2. GitHub здесь

Следующий этап

Что дальше?

1. Расширение интерфейса (опции критериев удаления)
2. Удаление дубликатов с опечатками (расстояние 1 символ)
3. Поиск опечаток в переводе
4. Извлечение терминологии
5. Составление двуязычных глоссариев…

Изученные материалы

Классный учебник по Python

- Справочник по языку Python3

Работа с XML-файлами в Python

- Создание и сборка XML-документов

Регулярные выражения

- Регулярные выражения - от простого к сложному

Интерфейс командной строки (аргументы для скрипта)

- Модуль argparse

Вывод информации в терминал

- Объекты stdin, stdout, stderr модуля sys в Python