Автоматчиеское определение ритма стихотворных произведений на русском языке

Гусев Илья

Московский физико-технический институт
(государственный университет)

Факультет инноваций и высоких технологий
Кафедра компьютерной лингвистики

Москва, 2017

Содержание

Тема исследования

- Автоматчиеское определение ритма стихотворных произведений русского языка.
- Разбивается на несколько подзадач:
  - Сбор корпуса, его очистка
  - Акцентуация
  - Метрическая классификация
  - Определение рифм
  - Создание простого корпус-менеджер в вебе

Постановка задачи

- Создание полезного ресурса (как для филолгов, так и для лингвистов и программистов).
- Разбор методов и алгоритмов для каждой из существующих подзадач, разработка новых.
- Сделать так, чтобы всем этим мог воспользоваться максимально широкий круг людей
- По возможности использование языконезависимых методов.

Ресурсы

- `rupoem.ru` - поэзия 19-20 века, стихи в основном из общественного достояния -> можно легально использовать. Порядка 16000 стихотворений, 200 авторов, 2кк токенов, 13кк символов.
- `stihi.ru` - современная поэзия, использовать можно только нелегально :) Зато 37кк стихов.

Сбор - Scrapy

- Python фреймворк для парсинга сайтов.
- Поддержка CSS селекторов и XPath
- Много форматов экспорта

Очистка - руками

- Аккуратная обработка тегов.
- Эвристики для поиска дубликатов
- Вытасикиваем:
  - Автор
  - Название
  - Даты создания (если есть)
  - Текст
  - Темы (есть примерно у 25% корпуса)

Токенизация

- Самый простой вариант - по пробелам
- Для английского - `nltk.word_tokenizer`
- Для русского - по проблеам + словарь слов с дефисом.

Ресурсы

- Полная акцентуированная парадигма по А. А. Зализняку - 3кк словоформ, есть главные и вторичные ударений, без фонем (`http://www.speakrus.ru/dict/`)
- Wiktionary - для многих языков, с фонемами (нужно дополнительно парсить или через API вытаскивать). (`https://ru.wiktionary.org`)
- CMUDict, классический фонетический словарь для английского языка. (`http://www.speech.cs.cmu.edu/cgi-bin/cmudict`)

Пути решения

- Ограничиться словарём - строим над ним бор, в терминальных вершинах храним ударения. Проблема - что делать с незнакомыми словами? В целом, достаточно для многих стихов.
- End-to-end классификатор из графем сразу в ударения. Деревья решений, RF, RNN. Зачастую не очень высокая точность, изначально использовал именно этот подход.
- Grapheme-to-phoneme + классификатор на фонемах + деление на слоги, основанное на фонемах. Фонетический разбор может также пригодиться при дальнейшем анализе (для определения рифм и клитик).

Задача G2P

Фонемы я беру из IPA(International phonetic alphabet, Международный фонетический алфавит). Шаги:

- Выравнивние обучающей выборки (EM-алгоритм или динамика с заданной матрицей сопоставлений). Пример: **'ааронов ец '** -> **'ɐɐronəvʲɪts'**
- Обучение трансляции (WFST для n-gram пар (g, p) или RNN(LSTM, GRU) Seq2Seq, гибридные подходы)
- Трансляция незнакомых слов.

Таблица: Оценка (LSTM256+BLSTM256)->LSTM128, разбиение 0.9-0.1

	Wiki-Ru	CMUDict
WER	0.09	0.27
PER	0.02	0.03

Задача определения ударений по фонемам

Строим для каждого примера обучающей выборки маску ударений, примеры: **замок** -> **zɐmok** -> **[0, 1, 0, 1, 0]**, **корова** -> **kɐrovə** **[0, 0, 0, 1, 0, 0]**. Обучаем RNN для Seq2Seq, получаем неплохие результаты.

Таблица: Оценка BLSTM128, разбиение 0.9-0.1

	Wiki-Ru LSTM	CMUDict LSTM
WER	0.01	0.10

Задача разбиения на слоги

Используем теорию 'волны сонорности'. Утверждается, что все звуки обладают определённой 'звучностью', и по этой самой 'звучности' их можно разделить на несколько категорий - гласные, апроксиманты, носовые, и т.д. Слоги всегда имеют ядро - гласную, кооторая имеет максимальную 'звучность'. А слогораздел проходит в локальных минимумах функции звучности слова. Пример: **aftəvɐrʲiʃkə** -> **[6, 0, 3, 6, 0, 6, 4, 6, 0, 3, 6]** -> **[6]** + **[0, 3, 6]** + **[0, 6]** + **[4, 6]** + **[0, 3, 6]** -> **a-ftə-vɐ-rʲi-ʃkə**.

Простые метры

- **Ямб** - 2-сложная стопа, шаблон: -+, пример: 'мой дЯдя сАмых чЕстных прАвил'
- **Хорей** - 2-сложная стопа, шаблон: +-, пример: 'бУря мглОю нЕбо крОет'
- **Дактиль** - 3-сложная стопа, шаблон: +−, пример: 'вЫрыта зАступом Яма глубОкая'
- **Амфибрахий** - 3-сложная стопа, шаблон: -+-, пример: 'не вЕтер бушУет над бОром'
- **Анапест** - 3-сложная стопа, шаблон: −+, пример: 'тАм, в ночнОй завывАющей стУже'

Сложные метры

- **Дольник** - переменная стопа, длина интервалов между + отличается не более чем на 1, пример: 'Я зажглА завЕтные свЕчи'
- **Тактовик** - переменная стопа, длина интервалов между + отличается не более чем на 2, пример: 'вот далмаАт пришЁл ко мне лукАвый'

$$\text{простые метры} \subset \text{дольник} \subset \text{тактовик}$$

Наивный алгоритм

- Задаём каждый из метров шаблоном на языке, подозрительно похожем на регулярные выражения. Пример для ямба: **(us)*(us)(u)?(u)?**, где u - unstressed, s - stressed, компилируем их.
- Сравниваем их с заданной строкой, в которой мы проставили ударения акцентуатором.
- Односложные слова при сравнении игнорируем.
- Для остальных слов считаем ошибкой, если s в шаблоне стоит не том месте, где у нас стоит ударение.
- Таким образом получаем количество ошибок для кажого метра, выбираем метр с наименьшим.
- Делаем для всех строк, берём тот метр, количество строк которого максимально. Не забываем навесить коэффициенты на дольник и тактовик.

Рифма

Пока только для графем. На эвристиках - совпадение гласной, совпадение ударного и последующего за ударным слога.

Генератор на n-грамах

Размечаем тексты, в качестве элемента n-граммы считаем кортеж (слово, ударение), обучаем модель на корпусе. При генерации пересекаем модель с ограничениями заданного метра. Для поддержки ограничений по рифме 'разворачиваем' модель.

Ссылки

- `http://poetry-corpus.ru` - Корус на 16к стихов доступен онлайн. Движок - Django, полнотекстовой поиск через ElasticSearch, определение ритма, ударений, рифмы, генерация, слоистая разметка.
- `https://github.com/IlyaGusev/rupo/` - Модуль для Python, всё что было выше, можно ставить через pip.

G2P TODO

- Больше ресурсов
- Другие архитектуры сетей, использование CTC loss
- Пересечение с n-грамной моделью

Акцентуатор TODO

- Можно обучать одновременно с G2P
- Не делал оценку других классификаторов
- Использовать большой словарь Зализняка для обучения
- Разрешение омонимов - морфология и синтаксис

Рифмы TODO

- Использовать графемы
- Обучиться на размеченном по ритму корпусе

Метрический классификатор TODO

Тысячи их!

- Вертикальная оценка (между строками), матрица весов
- Клитики - сбор корпуса?
- Сбор эталона для оценки качества - самое важное
- Впилить уже фонетику

Генератор TODO

Этим занимается Даня Анастасьев.

- Нейронные сети для генерации - обучение языковой модели, морфология.

Спасибо за внимание!