Исправление опечаток и грамматических ошибок в русскоязычных текстах

Бунин Дмитрий, группа 792

Научный руководитель: канд. физ.-мат. наук Сорокин А. А.

Введение и обоснование актуальности

Около 8% несловарных слов в ГИКРЯ являются результатом опечаток [Т. О. Шаврина и Сорокин (2015)].

Около 15% всех поисковых запросов к Яндексу содержат по крайней мере одну ошибку [А. Байтин (2008)].

Наличие опечаток может существенно влиять на качество современных нейросетевых моделей, таких как BERT [Kumar, Makhija и Gupta (2020)].

Модуль исправления опечаток — неотъемлемая часть любого современного текстового редактора.

Постановка задачи

SpellRuEval

Взята задача, поставленная участникам соревнования SpellRuEval [Сорокин, А. В. Байтин и др. (2016)], проходившего в рамках конференции «Диалог 2016».

В состязании предлагалось исправлять опечатки и грамматические ошибки в текстах из социальных медиа.

Все исправляемые грамматические ошибки были локальными, т.е. затрагивали в основном одно слово.

Постановка задачи I

Данные, Обучающее множество

- Количество предложений: 2000.
- Всего исправлений: 1727.
- Количество уникальных исправлений, неизвестных словарю: 68.
- Распределение ошибок по выравниваниям:
  - один к одному: 80.0%;
  - один ко многим: 15.6%;
  - многие к одному: 4.1%;
  - многие ко многим: 0.2%.

Постановка задачи II

Данные, Обучающее множество

- Распределение количества ошибок в предложениях:

Постановка задачи

Метрики

Обозначим S_{cor} – множество корректных исправлений предложений, а S_{sol} – множество исправлений, выполненных моделью. Метрики:

$$P = \frac{|S_{sol} \cap S_{cor}|}{|S_{sol}|},$$
$$R = \frac{|S_{sol} \cap S_{cor}|}{|S_{cor}|},$$

Целевая метрика:

$$F_1 = \frac{2PR}{P + R}.$$

Обзор существующих решений

Историческое развитие

- Расстояние Дамерау-Левенштейна.
- Модель шумного канала.
- Разработка эффективных алгоритмов поиска слов с некоторой ошибкой.
- Контекстно-зависимые методы.

Обзор существующих решений

Русский язык

До проведения SpellRuEval почти не было работ для русского языка.

Большая часть из тех, что есть адресованы более конкретным проблемам, например, исправлению поисковых запросов [А. Байтин (2008)], [Панина, А. В. Байтин и Галинская (2013)].

Решение победителей SpellRuEval [Сорокин и Т. О. Шаврина (2016)] состояло из трех основных этапов:

1. Генерация предложений-гипотез.
2. Отсечение гипотез при помощи модели ошибок и языковой модели.
3. Ранжирование оставшихся гипотез логистической регрессией.

Описание моделей и методов

Алгоритм

Предварительные действия:

1. Токенизация и нормализация.
2. Инициализация «черного списка».
3. Генерации кандидатов с признаками и объединение токенов в группы.

Итерации исправления:

1. Поиск лучшей позиции для исправления и проверка критерия останова.
2. Выбор лучшего исправления .
3. Внесение позиций без исправлений в «черный список».

Модуль для генерации кандидатов с их признаками.
Для каждого токена генерируется список кандидатов при помощи моделей:

- `LevenshteinSearcher` — поиск слов на заданном расстоянии Дамерау-Левенштейна. Реализация заимствована из библиотеки DeepPavlov [Burtsev и др. (2015)].
- `PhoneticSearcher` — поиск слов с тем же фонетическим кодом. Алгоритм заимствован из решения победителей SpellRuEval.
- `HandcodeSearcher` — собранная вручную таблица исправлений на корпусе «Тайга».

Используется объединение словарей Хагена и wiktionary.

Помимо нахождения самих кандидатов для них также вычисляются признаки:

- Признаки позиции.
- Признаки источника.
- Индикаторы текущего/оригинального кандидатов.
- Содержание пробелов/дефисов.
- Индикатор словарности.
- Индикатор объединения.

В модели генерации кандидатов есть механизм для объединения двух последовательно идущих токенов в один для их совместного рассмотрения.

Это нужно для того, чтобы уметь исправлять вставки лишних пробелов:

- *и так* $\rightarrow$ *итак*,
- *как то* $\rightarrow$ *как-то*.

Алгоритм заключается в последовательном просмотре токенов слева направо и их жадном объединении если получаем словарное слово убрав пробел или заменив его на дефис.

Модуль предназначен для выбора позиции исправления и проверки критерия останова.

В его основе лежит использование n-граммных языковых моделей с модифицированным сглаживанием Кнезера-Нея [Chen и Goodman (1996)], обученных слева направо и справа налево на корпусе «Тайга» [Т. О. Шаврина и Шаповалова (2017)].

Реализация взята из библиотеки KenLM [Heafield (2011), Heafield и др. (2013)].

Алгоритм работы:

1. Для каждой позиции и каждого кандидата находится логарифм вероятности от этих моделей.
2. Считается агрегирующий показатель — среднее гармоническое логарифмов вероятностей от левой и правой моделей. Для каждой позиции находится разница показателей между лучшим кандидатом и текущим.
3. В качестве позиции для исправления позиция берется та, которая
   - не находится в «черном списке»,
   - разница показателей в которой максимальна и превышает некий порог.
4. Вычисленные вероятности и метрика записываются в признаки кандидатов.

Описание моделей и методов

Модуль выбора исправления

Модуль предназначен для выбора корректного исправления из списка в заданной позиции.

Используется модель RuBERT [Devlin и др. (2019)] [Kuratov и Arkhipov (2019)] или Conversational RuBERT и сгенерированные на предыдущих шагах признаки.

Ранжирование на основе предсказания замаскированных токенов в модели BERT.

Алгоритм:

1. Токенизация предложений при помощи WordPiece-токенизатора, использующегося в BERT.
2. Токенизация кандидатов.
3. Для каждого подтокена в кандидате измерить логарифм его вероятности, отключив механизм внимания для других подтокенов.
4. Все полученные логарифмы вероятностей для одного кандидата усреднить.

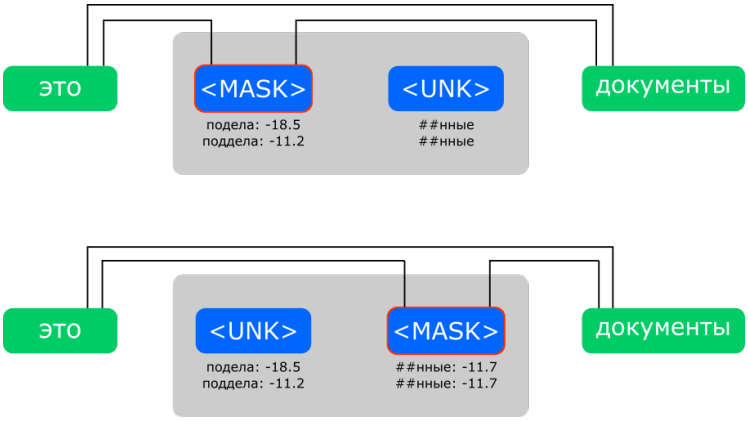

Рис. 1: Иллюстрация работы ранжирования при помощи BERT.

Было выяснено, что ранжирование только при помощи BERT имеет низкое качество.

Протестировав модель на обучающем множестве, удалось собрать выборку, содержащую данные, попадающие в модуль выбора исправления.
Признаки:

- признаки из модуля генерации кандидатов,
- признаки из модуля выбора позиции,
- признаки после ранжирования BERT.

В качестве моделей над признаками были испытаны ranking SVM [Joachims (2002)], CatBoost [Dorogush, Ershov и Gulin (2018)].

Результаты

Сравнение с другими моделями

Модель	**Точность**	**Полнота**	F_1
Яндекс.Спеллер	83.09	59.86	69.59
JamSpell	44.57	35.69	39.64
SpellRuEval Baseline	55.91	46.41	50.72
SpellRuEval Winner	81.98	69.25	75.07
Только BERT	48.76	57.89	52.94
Лучшая модель	**87.70**	**73.23**	**79.81**
Быстрая модель	86.04	70.80	77.68

Таблица 1: Сравнение результатов различных моделей.

Результаты

Потребление ресурсов

- Скорость лучшей модель: ≈ 0.6 предл./с.
- Скорость быстрой модели ≈ 4.8 предл./с.
- Расход памяти: ≈ 7.5ГБ.

Процессор: Intel Core i5-3210M (2 ядра, 4 потока, 2.5ГГц).

Такой большой расход памяти вызван хранением словаря в виде префиксного бора и может быть уменьшен в более качественной реализации.

Заключение

Спасибо за внимание!

Исходники доступны по ссылке: `https://github.com/Mr-Geekman/bd-research`.

Ответы на вопросы

Примеры работы модели

1. здесь есть только один по настоящему напряженный момент все остальные давольно гладкие
   здесь есть только один по-настоящему напряженный момент все остальные довольно гладкие
2. да я ваще за образ жизни а ладах с природой
   да я вообще за образ жизни а ладах с природой
3. он на столько крут что я готова приклонятся перед ним
   он настолько крут что я готова приклонятся перед ним
4. остаеться только кричать его имя в без результат ных поисках его в этой пустоте
   остается только кричать его имя в без результатных поисках его в этой пустоте

Ответы на вопросы

Применение модели

1. Создание веб-сервиса по аналогии с Яндекс.Спеллером.
2. Предварительная обработка текста перед применением методов, чувствительных к опечаткам.

Литература I


Burtsev, Mikhail и др. (2015). «DeepPavlov: Open-Source library for dialogue systems». Англ. В: *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of System Demonstrations*, с. 122—127. DOI: 10.18653/v1/p18-4021.


Chen, Stanley F. и Joshua Goodman (1996). «An empirical study of smoothing techniques for language modeling». Англ. В: *Proceedings of the 34th annual meeting on Association for Computational Linguistics* -. Т. 213. Morristown, NJ, USA: Association for Computational Linguistics, с. 310—318. ISBN: 9781626239777. DOI: 10.3115/981863.981904. URL: http://portal.acm.org/citation.cfm?doid=981863.981904.


Devlin, Jacob и др. (2019). «BERT: Pre-training of deep bidirectional transformers for language understanding». Англ. В: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1.Mlm, с. 4171—4186. arXiv: 1810.04805.


Dorogush, Anna Veronika, Vasily Ershov и Andrey Gulin (2018). «CatBoost: Gradient boosting with categorical features support». Англ. В: *arXiv*, с. 1—7. ISSN: 23318422. arXiv: 1810.11363.


Heafield, Kenneth (2011). «KenLM : Faster and Smaller Language Model Queries». Англ. В: *Proceedings of the Sixth Workshop on Statistical Machine Translation* 2009, с. 187—197. URL: http://www.aclweb.org/anthology/W11-2123%7B%5C%%7D5Cnhttp://kheafield.com/code/kenlm.


Heafield, Kenneth и др. (2013). «Scalable modified Kneser-Ney language model estimation». Англ. В: *ACL 2013 - 51st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference* 2, с. 690—696.


Joachims, Thorsten (2002). «Optimizing search engines using clickthrough data». Англ. В: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, с. 133—142. DOI: 10.1145/775047.775067.

Литература II


Kumar, Ankit, Piyush Makhija и Anuj Gupta (2020). «Noisy Text Data: Achilles' Heel of BERT». Англ. В: с. 16—21. DOI: 10.18653/v1/2020.wnut-1.3.


Kuratov, Yuri и Mikhail Arkhipov (2019). «Adaptation of deep bidirectional multilingual transformers for Russian language». Англ. В: *arXiv*. ISSN: 23318422. arXiv: 1905.07213.


Байтин, А. (2008). «Исправление поисковых запросов в Яндексе». В: *Российские Интернет Технологии*.


Панина, М. Ф., А. В. Байтин и И. Е. Галинская (2013). «Автоматическое исправление опечаток в поисковых запросах без учета контекста». В: *Компьютерная Лингвистика И Интеллектуальные Технологии: По Материалам Ежегодной Международной Конференции «Диалог»* 12(19), с. 556—567.


Сорокин, А. А., А. В. Байтин и др. (2016). «SpellRuEval: The first competition on automatic spelling correction for Russian». В: *Компьютерная Лингвистика и Интеллектуальные Технологии*, с. 660—673. ISSN: 20757182.


Сорокин, А. А. и Т. О. Шаврина (2016). «Автоматическое исправление опечаток и орфографических ошибок для русскоязычных социальных медиа». В: *Компьютерная Лингвистика и Интеллектуальные Технологии*, с. 688—701. ISSN: 20757182.


Шаврина, Т. О. и О. Шаповалова (2017). «К методике создания корпусов для машинного обучения: «Тайга», синтаксический корпус и парсер». В: *По материалам Международной Конференции «Корпусная Лингвистика – 2017»*, с. 78—84.


Шаврина, Т. О. и А. А. Сорокин (2015). «Моделирование расширенной лемматизации для русского языка на основе морфологического парсера TnT-Russian». В.