Исправление опечаток и грамматических ошибок в русскоязычных текстах

Бунин Дмитрий, группа 792

Научный руководитель: Сорокин А. А.

Датасет

Был взят датасет с соревнования SpellRuEval: [Sorokin, Baytin и др. (2016)].
Что исправляется:

- опечатки (*мнея* $\rightarrow$ *меня*),
- орфографические ошибки (*митель* $\rightarrow$ *метель*),
- употребление неверного слова (*компания* $\rightarrow$ *кампания*),
- намеренно неправильное написание (*ваще* $\rightarrow$ *вообще*),
- грамматические ошибки (согласование) (*он видят* $\rightarrow$ *он видит*),
- ошибки с пробелами/дефисами (*както* $\rightarrow$ *как-то*),
- смешанное использование чисел и букв в числительных (*2-ух* $\rightarrow$ *двух*),
- использование цифр вместо букв (*в4ера* $\rightarrow$ *вчера*).

Датасет

Что не исправляется:

- иностранные слова,
- неформальные сокращения (*прога* $\rightarrow$ *программа*),
- пунктуационные ошибки,
- ошибки, связанные с заглавным/строчным написанием,
- неразличение букв «е» и «ё».

Датасет I

Статистика, Обучающее множество

- Количество предложений: 2000;
- Всего исправлений: 1727;
- Распределение количеств ошибок в предложениях:

Датасет II

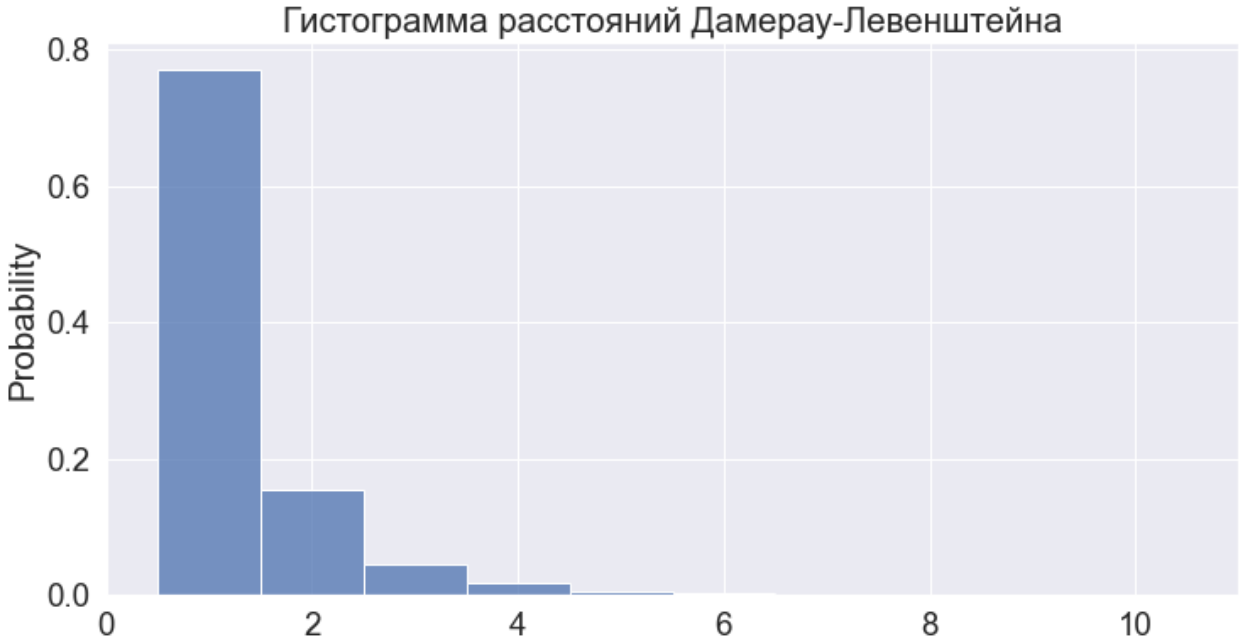
Статистика, Обучающее множество

- Распределение ошибок по выравниваниям:
  - Один к одному: 80.0%;
  - Один ко многим: 15.6%;
  - Много к одному: 4.1%;
  - Много ко многим: 0.2%.

Датасет III

Статистика, Обучающее множество

- Распределение ошибок по расстоянию Дамерау-Левенштейна:



- Количество исправлений, неизвестных словарю: 105.

Датасет

Статистика, Тестовое множество

- Количество предложений: 2000;
- Всего исправлений: 1976.

Архитектура модели

Алгоритм

Предварительные действия:

- Токенизация, оформление «черного списка».
- Генерации кандидатов и объединения токенов в группы.

Итерации исправления:

- Поиск лучшей позиции для исправления и проверка критерия останова.
- Исправление в выбранной позиции.
- Внесение позиций без исправлений в «черный список».

Архитектура модели

Candidate Generator

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

- `LevenshteinSearcher` — поиск слов из словаря на заданном расстоянии Дамерау-Левенштейна.
- `PhoneticSearcher` — поиск слов из словаря с тем же фонетическим кодом.
- `HandcodeSearcher` — собранная вручную таблица.

Используется словарь:
`https://github.com/danakt/russian-words/`.
Существует также механизм объединения токенов.

Архитектура модели

Candidate Generator, `LevenshteinSearcher`

- Реализация модели взята из библиотеки DeepPavlov: `https://github.com/deepmipt/DeepPavlov/tree/master/deeppavlov/models/spelling_correction/levenshtein`
- Словарь хранится в виде префиксного бора.
- Для поиска используется алгоритм A^* с h_2-эвристикой из статьи [Oflazer (1996)].
- Позволяет находить ошибки ненужного слияния слов (*вобщем* $\rightarrow$ *в общем*).

Архитектура модели

Candidate Generator, `PhoneticSearcher`

- Модель фонетического кодирования слов.
- Реализация по статье [Sorokin и Shavrina (2016)].
- Позволяет находить словарные слова похожие по звучанию: (*пользовацца* $\rightarrow$ *пользоваться*).
- Позволяет исправлять ошибки с повторением гласных (*дооолго* $\rightarrow$ *долго*).

Архитектура модели

Candidate Generator, `HandcodeSearcher`

Вручную созданная таблица для предложения кандидатов.

Построение:

1. Были взяты тексты из социальных сетей, которые попали в корпус «Тайга»: `https://tatianashavrina.github.io/taiga_site/`
2. Был взят словарь, являющийся объединением словарей Хагена и wiktionary.
3. Несловарные слова были упорядочены по частоте встречаемости.
4. Было проанализировано около 5000 слов, среди которых найдено около 140 исправлений.
5. Добавлено два словарных исправления по обучающему множеству и несколько исправлений, исходя из собственного опыта.

Архитектура модели

Candidate Generator, Генерация признаков

Помимо нахождения самих кандидатов для них также вычисляются признаки:

- Признаки позиции.
- Признаки источника.
- Индикатор текущего кандидата.
- Индикатор оригинального кандидата.
- Содержание пробелов/дефисов.
- Индикатор словарности (актуально только для оригинального токена).
- Индикатор объединения.

Архитектура модели

Candidate Generator, Объединение токенов

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.

Архитектура модели

Position Selector

Модель для нахождения позиции исправления на текущей итерации и проверки критерия останова. В ее основе лежит использование языковых моделей слева-направо и справа-налево.

Архитектура модели

Position Selector, Алгоритм работы

Алгоритм работы:

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

Архитектура модели

Position Selector, Языковые модели

Для построения языковых моделей была взята библиотека KenLM.

- В качестве обучающих данных была взята часть корпуса «Тайга».
- Текст был переведен в нижний регистр, из него была удалена пунктуация.
- Были обучены 3-граммные модели, использующие модифицированное сглаживание Кнезера-Нея: [Chen и Goodman (1996)].

Архитектура модели

Position Selector, Supervised модель

Была предпринята попытка обучить supervised модель на основе признаков кандидатов.

Хоть модель и имела большую полноту при нахождении позиций с ошибками это не привело к улучшению результата.

По всей видимости, модель хоть и находила новые позиции с ошибками, но подходящих кандидатов для их исправления не было.

Архитектура модели

Candidate Scorer

Модель ранжирования кандидатов на основе BERT [Devlin и др. (2019)] и других признаков.

Архитектура модели

Candidate Scorer, `BertScorer`

Ранжирование на основе предсказания замаскированных токенов при помощи модели. Используется Conversational RuBERT.

Алгоритм:

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

Архитектура модели

Candidate Scorer, Supervised модель

- Было выяснено, что ранжирование только при помощи BERT имеет весьма низкое качество, ему не хватает модели ошибок.
- Протестировав модель на обучающем множестве, удалось собрать выборку, где было известно какие кандидаты и с какими признаками попадают в Candidate Scorer.
- В качестве supervised моделей были испытаны модели: ranking SVM, CatBoost.
- Признаки:
  - признаки из candidate generator,
  - признаки из position selector,
  - признаки BERT:
    - среднее логарифов вероятностей подтокенов,
    - сумма логарифмов вероятностей подтокенов,
    - количество подтокенов.

Результаты

Напоминание целевых метрик

Пусть имеется текст из n предложений, тогда обозначим g_i – множество корректных исправлений предложения i, а e_i – множество наших исправлений.

$$R = \frac{\sum_{i=1}^{n} |g_i \cap e_i|}{\sum_{i=1}^{n} |g_i|},$$

$$P = \frac{\sum_{i=1}^{n} |g_i \cap e_i|}{\sum_{i=1}^{n} |e_i|}.$$

Целевая метрика:

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

Результаты

Сравнение с другими моделями

Spell Checker	**Precision**	**Recall**	F_1
Yandex.Speller	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
Лучшая модель	**86.56**	**72.37**	**78.83**
Быстрая модель	85.30	70.80	77.38

Таблица: Сравнение результатов различных моделей.

Результаты

Сравнение различных вариантов модели

Spell Checker	Precision	Recall	F_1
BERT	47.16	55.82	51.12
SVM	84.86	66.95	74.85
SVM+comb.	84.67	69.59	76.39
SVM+BERT	86.35	69.48	77.01
SVM+BERT+comb.	86.35	**72.37**	78.74
CatBoost	85.42	67.61	75.48
CatBoost+comb.	85.30	70.80	77.38
CatBoost+BERT	**87.08**	69.59	77.36
CatBoost+BERT+comb.	86.56	**72.37**	**78.83**

Таблица: Сравнение результатов вариаций модели.

Результаты

Скорость работы

Лучшая модель: $\approx$ 0.6 предл./с.

- Candidate Generator: 5.3%;
- Position Selector: 9.7%;
- Candidate Scorer: 84.9%.

Быстрая модель: $\approx$ 4.8 предл./с.

- Candidate Generator: 43.8%;
- Position Selector: 50.4%;
- Candidate Scorer: 5.4%.

Планы работ

1. Провести испытания с новым словарем.
2. Поэкспериментировать над некоторыми гиперпараметрами:
   - вариация BERT;
   - вариация языковых моделей;
   - размер вручную заданной таблицы.
3. Писать текст диплома.

Заключение

Спасибо за внимание!

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

Литература I

Chen, Stanley F. и Joshua Goodman (1996). «An empirical study of smoothing techniques for language modeling». 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». 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.

Oflazer, Kemal (1996). «Error-tolerant Finite-state Recognition with Applications to Morphological Analysis and Spelling Correction». В: *Computational Linguistics* 22.1, с. 69—89. ISSN: 08912017. DOI: 10.1184/r1/6287645.v1. arXiv: 9504031 [cmp-lg].

Sorokin, A. A., A. V. Baytin и др. (2016). «SPELLRUEVAL: The first competition on automatic spelling correction for Russian». В: *Komp'juternaja Lingvistika i Intellektual'nye Tehnologii*, с. 660—673. ISSN: 20757182.

Sorokin, A. A. и T. O. Shavrina (2016). «Automatic spelling correction for Russian social media texts». В: *Komp'juternaja Lingvistika i Intellektual'nye Tehnologii*, с. 688—701. ISSN: 20757182.