Классификация текстов комментариев по классам токсичности

A Preprint

Макаров Александр Владиленович
студент НГТУ АВТФ, 3 курс
makarov.alxr@yandex.ru

22 февраля 2019 г.

Abstract

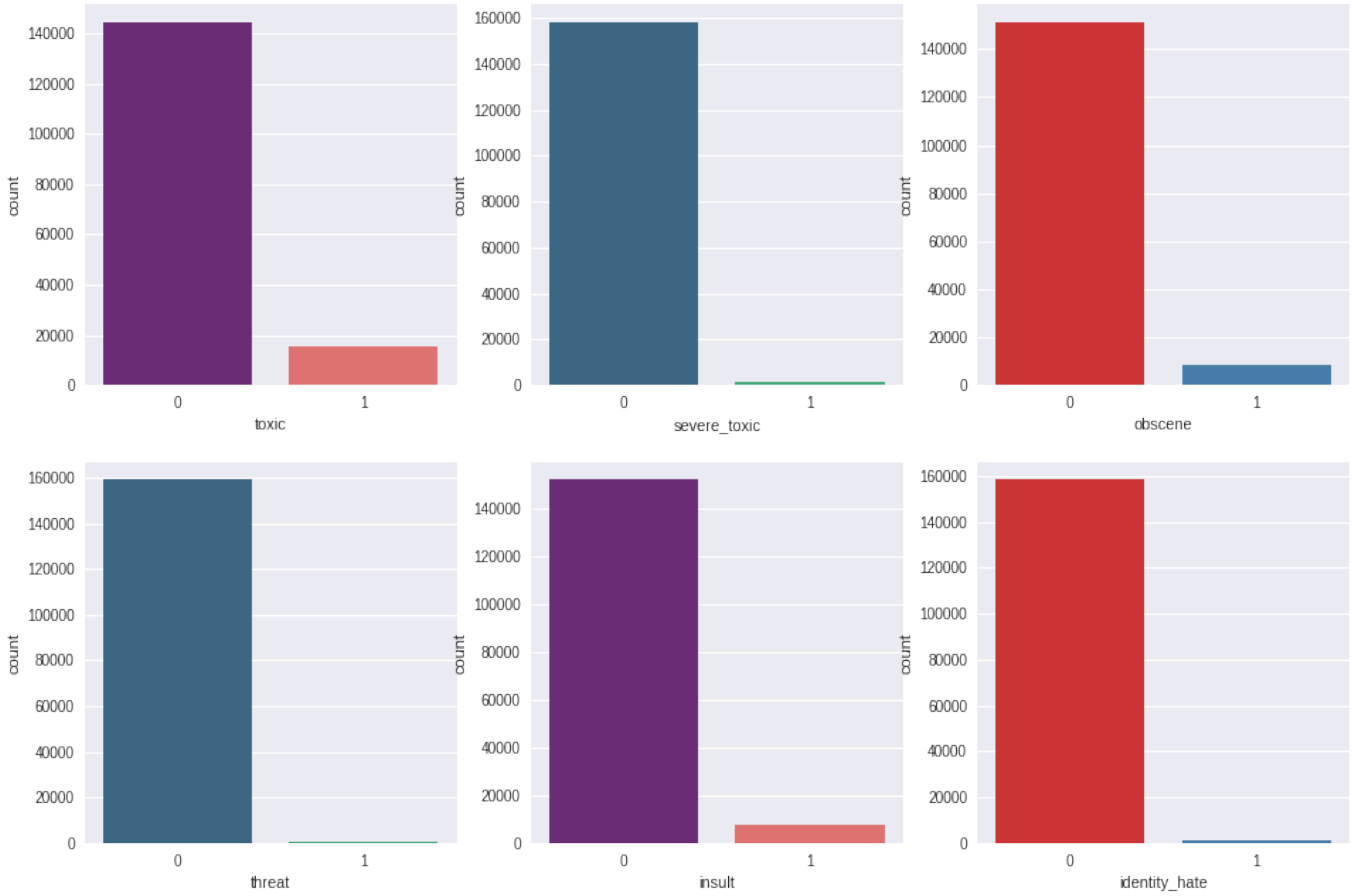
1 Введение

Анализ эмоциональной окраски текстов на естественном языке - одна из классических задач NLP. В этой статье будет произведен обзорный анализ существующих методов её решения разной сложности. Комментарии собраны из интернет ресурса "Википедия"и содержат специфические особенности, о которых поговорим в разделе "Анализ датасета".

2 Анализ датасета

Данная задача является классической задачей multi-label классификации. Для успешного обучения требуется сбалансированный набор данных.
К сожалению, большая часть датасета содержит в себе примеры, не относящиеся ни к одному классу - всего их 0.898%
При этом, чем к большему количеству классов относится пример, тем меньшее количество таких примеров:

Количество примеров	
Количество классов	Количество примеров
0 классов	143346
1 класс	6360
2 класса	3480
3 класса	4209
4 класса	1760
5 классов	385
6 классов	31



А некоторые сочетания классов не встречаются вовсе, некоторые же - только один раз.

Средняя длина сообщения - 394 символа.

Кроме того, тексты содержат большое количество опечаток и нестандартных обозначений. В текстах встречаются общеупотребимые сокращения, URL, даты и числа.
Самые популярные слова с яркой негативной окраской: fuck, nigger, suck, shit, hate, fat, die.

3 Линейная регрессия на TF-IDF и n-grams

Самым частым маркером оскорбительных комментариев являются табуированные слова: ругательства, оскорбления, уничижительные выражения, слова с яркой негативной окраской и т.п.
Появляется возможность выделения текстов в группы на основе только знаковых слов, с чем должна неплохо справляться линейная регрессия. В качестве входных значений выделяются n-grams по словам и символам с окном, равным трём.
Для оценки эффективности работы алгоритма возмём AUC-ROC (о причинах этого решения - пункт 4.3.2)
Результаты на тестовой выборке:

Класс	AUC-ROC	micro f1 score	macro f1 score
toxic	0.89	0.92	0.63
severe$_toxic$	0.96	0.98	0.63
obscene	0.91	0.95	0.66
threat	0.92	0.98	0.54
insult	0.92	0.96	0.68
identity$_hate$	1	0.99	0.81

4 Нейросетевй подход

Поскольку данные сырые, pipeline состоит из трёх главных частей:

1. Препроцессинг исходного текста
2. Получение эмбеддингов
3. Нейросеть

4.1 Препроцессинг

Для исправления опечаток используется перебор по словарю с заменой слова на ближайшее по расстоянию Левенштейна. Помимо явных ошибок, тексты изобилуют сокращениями и нестандарными написаниями слов, распространённых в интернете. Для того, чтобы привести их к одной форме, проще всего использвать кастомные фильтры, например, на основе регулярных выражений.

4.2 Эмбеддинг

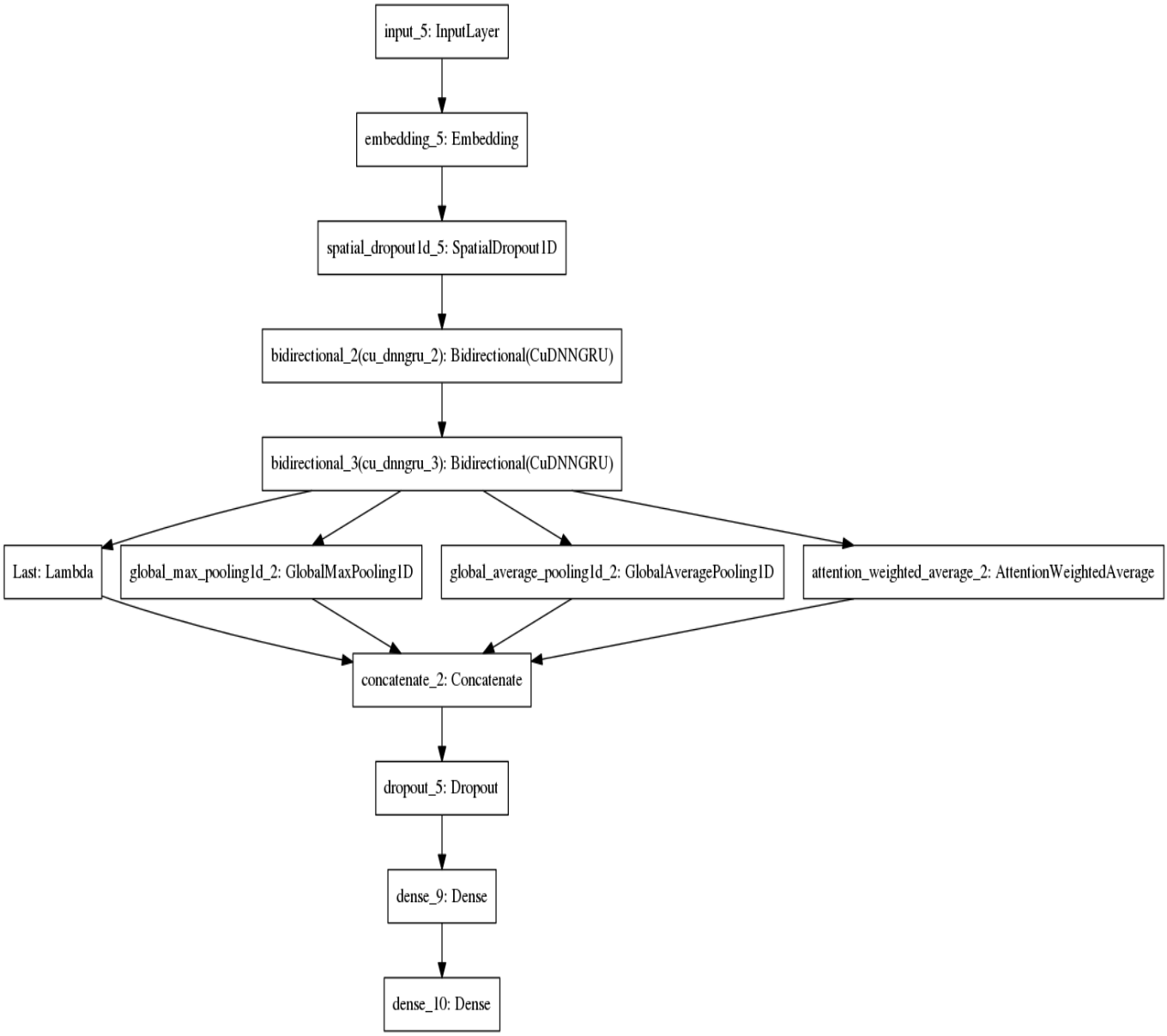
Для представления текстов в векторной форме используется некий эмбеддинг. Задача довольно локальна, и его можно получить в процессе обучения, используя Embedding layer, изначально инициализированный случайными значениями. Однако, если использовать предобученный эмбеддинг, дообучая его вместе с моделью. Это ускоряет сходимость модели и, в среднем, улучшает итоговый результат.
Сразу оговоримся, что такие глубокие модели как BERT (и любые другие transformer-like модели) для получения эмбеддингов не рассматривались, отчасти потому, что в нашем случае это overhead метод.
Таким образом, для английского языка остаются ещё как минимум три проверенные модели:

1. Word2Vec
2. GloVe
3. Fasttext

Первые два обладают существенным недостатком - в качестве атомарного элемента они используют слово. В случае с комментариями из интернета, даже после предобработки текстов остается очень много артефактов и нестандартных написаний. Это приводит к искусственному увеличению количества векторных представлений одних и тех же слов.
Fasttext использует n-grams, что обеспечивает ему большую устойчивость. Даже при нескольких вариантах написания одного и того же слова их векторные представления будут находится очень близко в общем пространстве признаков. Именно поэтому далее в качестве инициализатора для эмбеддинг слоёв используется fasttext [1]

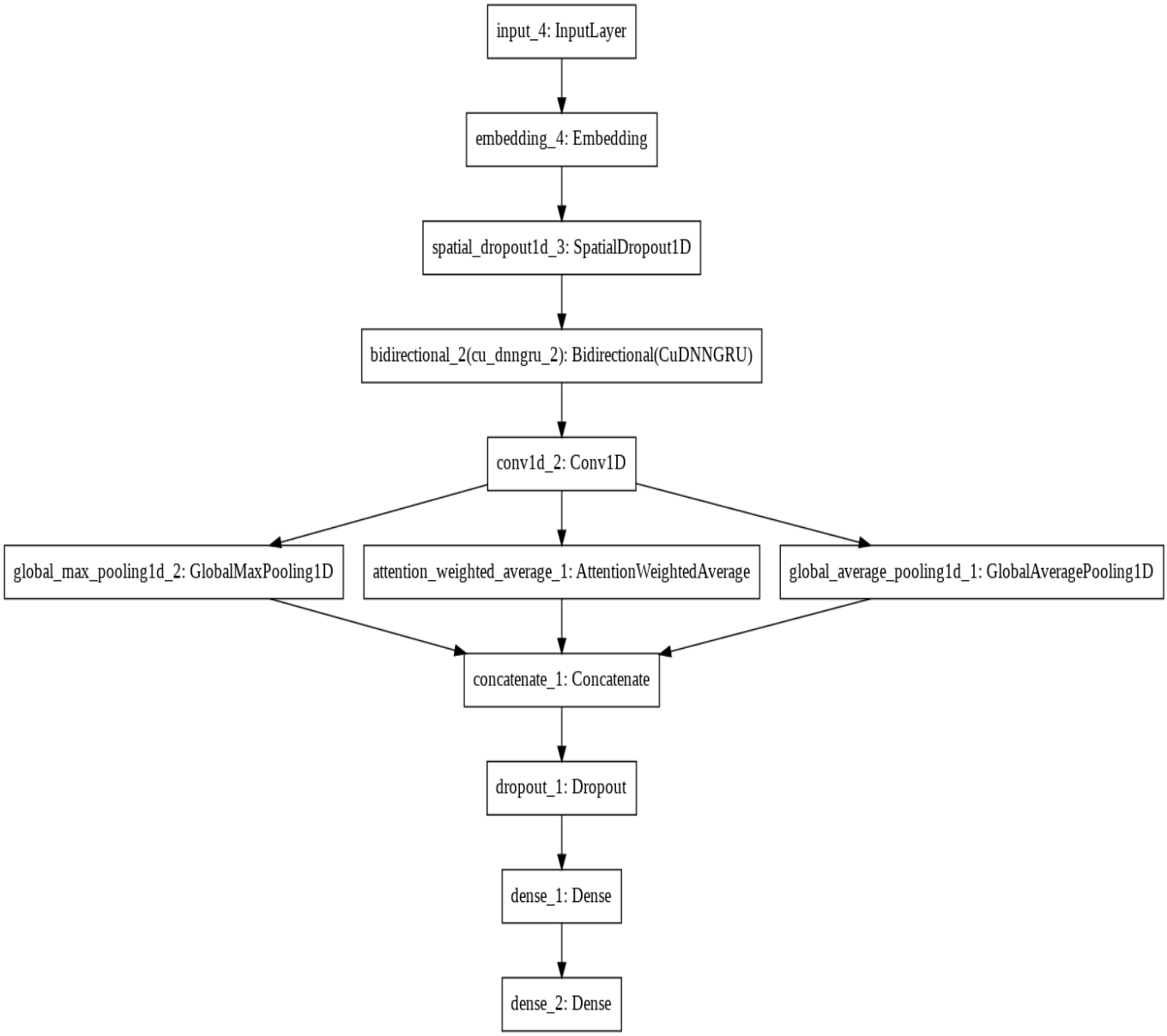
4.3 Выбор модели нейросети

Условия датасета приводят к тому, что
1. Классическое решение - использование Text-RNN. Для суммирования результатов применяем pooling слои. Кроме того, для выделения ключевых фич уместно использовать Attention слой, а для увеличения связности - последнее скрытое состояние реккурентного блока.



2. Исследования последних лет показывают эффективность применения свёрточных сетей для классификации текстов. Опираясь на работу Ye Zhang и Byron C. Wallace [3], было выдвинуто предположение, что обобщающей способности большого количества карт различного размера окажется достаточно для получения высокоуровневых признаков:

3. Попытка объеденить вышеназванные модели - реккурентная нейросеть со свёртками и пулингом:



Все эти модели достаточно просты, чтобы не сильно переобучаться на данных. Реккурентные сети, свёртки и пулинг - классические инструменты для классификации текстов.

4.3.1 Выбор функции потерь

Определим функцию ошибки, с помощью которой будем обучать модели. Для этого формализуем задачу, которую должна решить нейросеть. Для каждого комментария модель выдвигает шесть (число классов) гипотез - принадлежит ли пример к каждому классу. Таким образом, мы можем считать, что нейросеть генерирует некое распределение вероятностей. В этом случае лучший результат будет означать увеличение правдоподобия, из чего выводится функция ошибки - известная нам как logloss или кросс энтропия.

4.3.2 Выбор метрики

Выбор метрики - одно их самых важных условий, поскольку на этом основании определяется, какая модель лучше справляется с задачей. Т.к. метрика должна адекватно отражать качество решения конкретной задачи, при выборе следует учесть следующее:

1. Классы сильно несбалансированны.
2. Для конвертации вещественного значения, предсказанного нейросетью, в бинарную метку необходимо выбрать границу, разделяющую два класса (например, 0.5).
3. Для двух разных моделей на одной стадии обучения это пороговое значение будет различаться.

Опираясь на эти условия, была выбрана метрика AUC-ROC. Её рассчет не привязан к пороговому значению, кривая хорошо показывает себя в задачах с несбалансированными классами и AUC-ROC гораздо меньше подвержена завышению качества на маленьких датасетах, чем AUC-PR[2]

5 Результаты тестов

Модели проверялись на тестовом датасете и обучались, пока ошибка не переставала уменьшаться быстрее 5e-4 (т.н. механизм ранней остановки).

Модель	AUC-ROC
Average RNN	0.9852
Text CNN	0.9843
Average RCNN	0.9847

Список литературы

[1] English word vectors trained using fastText. https://fasttext.cc/docs/en/english-vectors.html

[2] Jesse Davis, Mark Goadric, The Relationship Between Precision-Recall and ROC Curves

[3] Ye Zhang, Byron C. Wallace. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification https://arxiv.org/pdf/1510.03820.pdf