Aspect Based Sentiment Analysis

Описательная статистика датасета

Описательная статистика датасета будет фрагментарно даваться в разделах далее, здесь также кратко расскажем о датасете.

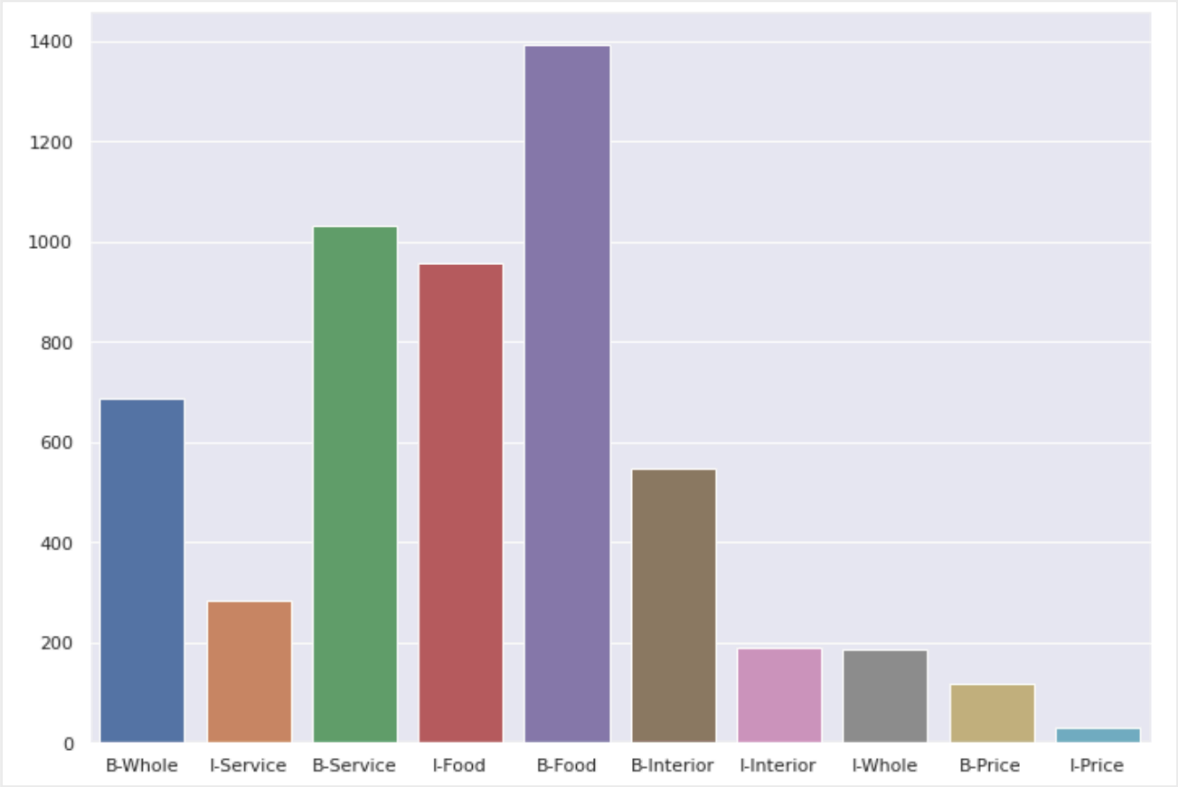
В файле train_aspects.txt лежит разметка каждого тоокена каждого предложения по тональности (neutral/positive/negative/both) и аскпекту (Food, Interior, Whole, Service), ниже приведен пример разметки некоторых токенов по обоим признакам.

	text_id	category	mention	start	end	sentiment
0	3976	Whole	ресторане	71	80	neutral
1	3976	Whole	ресторанах	198	208	neutral
2	3976	Whole	ресторане	256	265	neutral

Всего в корпусе 284 текста (train_reviews.txt)

Задача NER: предсказываем BIO-тег каждому токену

Для начала рассмотрим распределение тегов (без O):



Есть небольшой дисбаланс классов, можем предположить, что хуже всего будут предсказазываться самые малочисленные

классы - весь домен Price (B-Price, I-Price), возможно чуть лучше I-Interior, I-Whole. Конечно же, идеально, чтобы «путались» такие классы внутри своего домена: например, вместе I-Price предсказывался B-Price - это как будто лучше B-Whole тега.
Для решения данной задачи мы попробовали следующие методы:

- MultinomialNB
- Conditional Random Fields
- а также в планах было использовать tiny-RuBERT модель, но CRF нас вполне устроил своим результатом

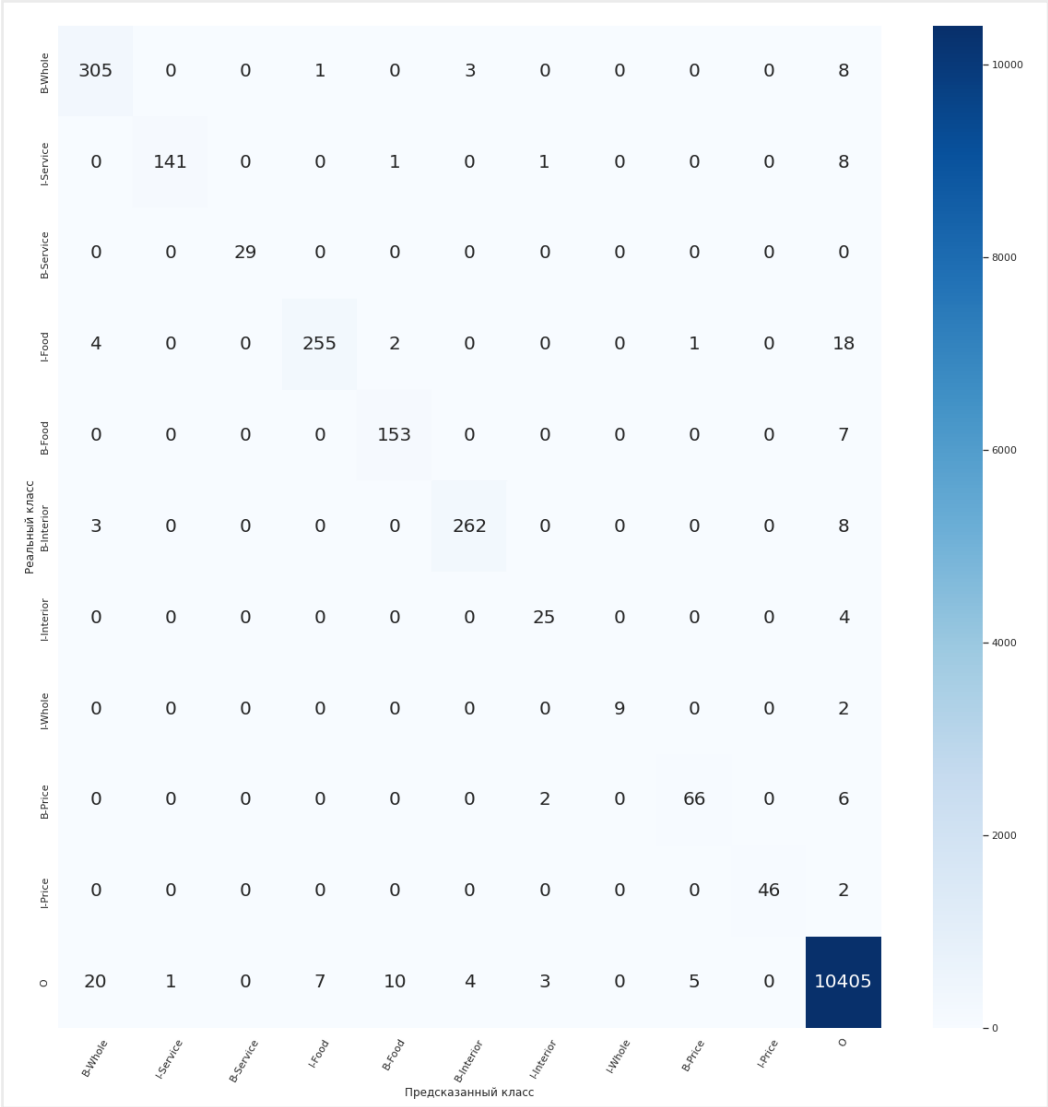
Мы оставили два самых «успешных» эксперимента, в тетради представлен код с перебором гиперпараметров для CRF, performance модели с лучшей метрикой (F1-score) со следующими параметрами:

c1 = 0.3196504797629
c2 = 0.00034474520532276035
algorithm='lbfgs'
max_iterations=100,
all_possible_transitions=True

Ниже classification report на тестовых данных:

	precision	recall	f1-score	support
B-Food	0.919	0.962	0.940	317
I-Food	0.974	0.960	0.967	273
B-Interior	0.993	0.934	0.962	151
I-Interior	0.806	0.862	0.833	29
B-Price	1.000	1.000	1.000	29
I-Price	1.000	0.818	0.900	11
B-Service	0.970	0.911	0.939	280
I-Service	0.917	0.892	0.904	74
B-Whole	0.922	0.956	0.939	160
I-Whole	1.000	0.958	0.979	48
micro avg	0.950	0.941	0.945	1372
macro avg	0.950	0.925	0.936	1372
weighted avg	0.951	0.941	0.946	1372

Посмотрим на confusion matrix:



Интерпретация результатов

В большинстве случаев классы были предсказаны верно! Всего была допущена 131 ошибка из 11696 всех тегов. Если не считать ‘O’ класс (как традиционно делается), то была допущена 131 ошибка из 1228.
Анализ ошибок модели показал следующее:

1. конечно, преобладание тегов ‘O’ несколько сместило распределение предсказаний модели, особенно, как и предполагалось ранее, небольших по количеству примеров классов
2. приятно было, что случались внутридоменные миксы, однако все еще были ситуации «не очень»: когда первый токен последовательности аспектов предсказывался как inside-тег: например, все три слова одного аспекта как I-класс, а еще хуже, когда первый токен выделялся как I, а последующий как B -

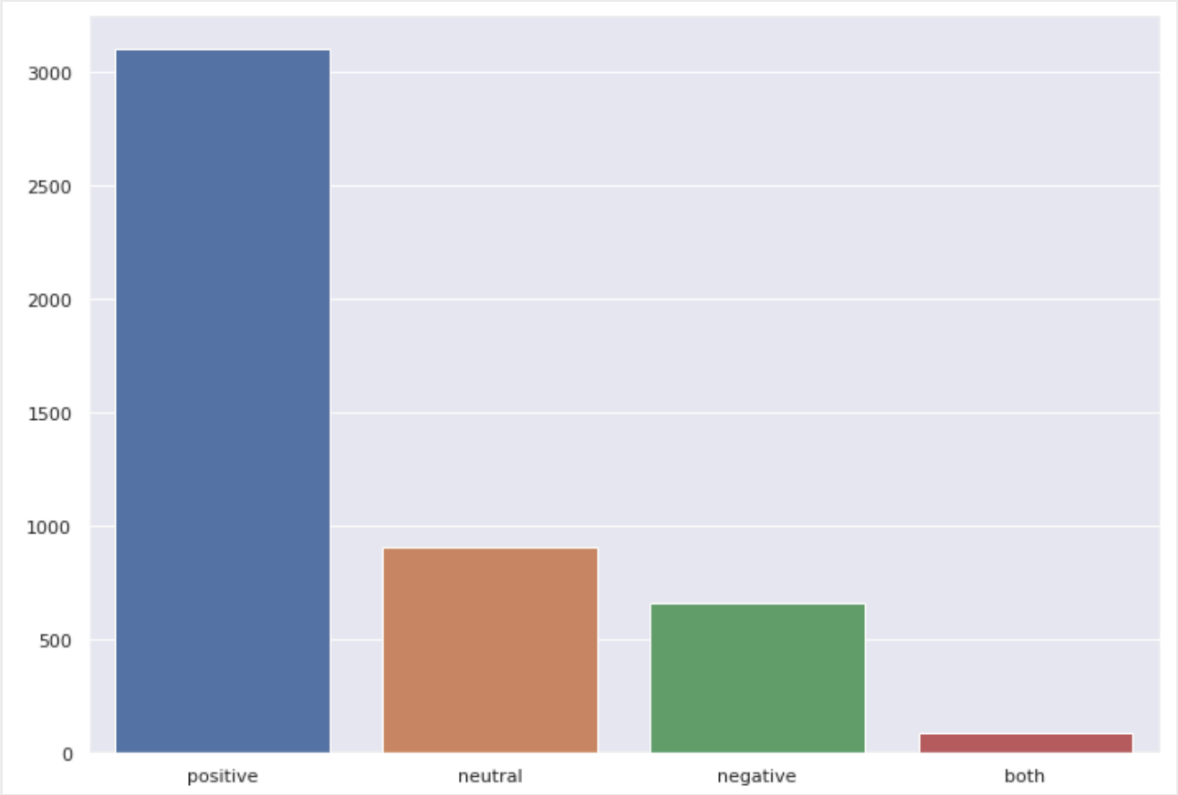
это может говорить о переобучении модели на определенные слова, эту проблему также в дальнейшем может требовать решения при более частых случаях такого рода ошибок
3. также видно, что самые маленькие классы смешивались с самыми объемными - но тут мы Америку не открыли и на confusion matrix это можно так же заметить

Дальнейшие шаги, возможности

1. Можно попробовать на задаче NER-классификации предобученный BERT на русском языке, я бы даже брала его дистиллированную версию для более легкого и быстрого файн-тьюнинга. Там может помочь с проблемой дисбаланса класса взвешенная кросс-энтропия, возможно, маленький learning rate, scheduler также мог бы положительно повлиять на лучшую сходимость модели.
2. Можно больше внимания уделить контексту аспектов: учитывать чуть больше контекста справа/слева, однако это идея может и негативно сказаться на качестве модели
3. Можно подумать что-то про лемматизацию всего текста, однако также есть риски того, что модель по итогу будет иметь чуть хуже обобщающую способность

Задача классификации: оценка тональности

Для начала посмотрим на распределение тегов на обучабщем датасете:



Видим сильный «перевес» класса положительной тональности, понимаем, что возможен риск переобучения моделей в стороны полного предсказания только позитивного класса, а также возможно, что самый малочисленный класс может очень плохо предсказываться

Дальнейшие шаги, возможности

1. Была попытка использования предобученных эмбеддингов Word2Vec, но она не привела к положительным результатам, способных улучшить качество, однако этот подход можно развивать далее: мир не сошелся на Word2Vec предобученных векторах - можно попробовать взять предобученный FastText или обучить на своем корпусе Word2vec/FastText (кажется более успешной первая идея с предобученный FastText'ом)
2. Можно эмбеддить не только слово или словосочетание, но и +/- n соседей рядом, это может помочь повысить финальную метрику
3. Можно создавать «костыли» с эмбеддингами, например, сделать какой-то датасет с парами «самых частотных»

представителей каждого класса и на тестовой выборке искать косинусной близостью самый близкий эмбеддинг из сохраненный и приписывать класс сохраненного эмбеддинга. Это может иметь обратную сторону медали: так можно найти полные антонимы, а в задаче определения тональности это может быть опасно, поэтому для большей подстраховки стоит брать топ-n самых близких эмбеддингов. А еще можно задуматься, как в целом находить «самых типичных представителей класса» - можно попробовать сделать кластеризацию на 4 класса и взять центроиды и посмотреть, кто они, и их и сохранить как самых типичных и по ним определять классы на тестовой выборке.