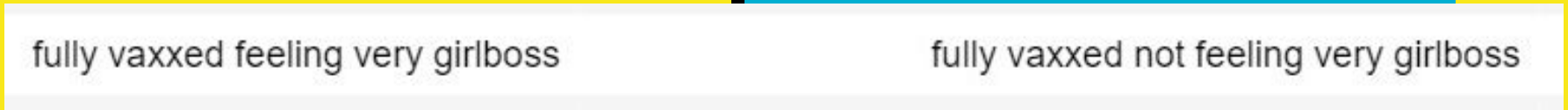

Sarcasm Detection

Алла Горбунова & Елизавета Клыкова

Данные

Источник: английские твиты из датасета iSarcasmEval (SemEval 2022)

Объем: 3468 твитов (867 или 25% саркастичных, 2601 или 75% не-саркастичных) + перифраз для саркастичных; включая перифраз – 3468 не-саркастичных, 4335 всего

Особенности: тексты размечены с точки зрения “сарказм / не сарказм” самими авторами, для каждого саркастичного текста приводится вариант без сарказма

Достоинства: объективность разметки, т. к. разметчики – сами авторы текстов

Недостатки: предложения, перефразированные из сарказма, могут быть неестественными

Задачи

Задача 1 и задача 3 из предложенных в рамках *iSarcasmEval*:

а) по тексту определить, саркастичный он или нет

Подзадача: проверить предположение о “неестественности” перефразированных несаркастичных текстов.

- сначала в класс не саркастичных текстов включим только оригинальные твиты
- потом добавим перифразы и посмотрим, как лучше

б) внутри пары “сарказм + его перифраза” определить, какой из двух текстов саркастичный

Что уже сделано

1. Проанализированы данные (Лиза)
2. Поставлены бейслайн эксперименты для всех подзадач (Алла)
3. Добавлен препроцессинг (Лиза)
4. Сделана попытка аугментации (Лиза)

Анализ данных

Проблема №1: в оригинальных твитах есть особые виды токенов, указывающие на саркастичность текста.

- ❏ @ имя пользователя
- ❏ # хештеги
- ❏ эмодзи
- ❏ гиперссылки

У перефразированных текстов этого нет –> нейросеть быстро это выучит.

Как быть? Можно считать за фичу и использовать, чтобы повысить качество моделей (например, заменять имена пользователей на специальный токен), но это нечестно и может нас подвести, если в тестовых данных это учтено.

Анализ данных

Проблема №2: оригинальные саркастичные твиты в среднем значительно длиннее, чем их перефразированные варианты.

- оригинальные: 18.3 токена (~100 символов)
- перефразированные: 13.8 токенов (~73 символа)
- в среднем: 17.8 токенов (~98 символов)

Видимо, люди в принципе стремятся перефразировать короче

Анализ данных

Проблема №3: в датасете много неочевидных случаев.

Перефразированный вариант не всегда правильный:

- *Оригинал:* Happy birthday to me!! I’m 30! 😭
- *Перифраз:* I'm 30! Id like to wish myself a happy birthday

Авторы часто добавляют лишнее к текстам:

- I could have said “this investigation was not legitimately carried out.”

Ну и наше любимое:

- *Оригинал:* Synthwave communities really love pessimism don't they haha Any upbeat energy and people get upset at ya
- *Перифраз:* yes

Baseline

Во второй задаче бейслайн поменялся относительно плана из первой презентации

В каждой подзадаче сравнивались 5 классификаторов (логистическая регрессия, наивный Байес, SVC, KNN, Decision tree) + рандом:

	best	score
а) распознавание сарказма без перифраз	naive Bayes	0.347 F1
а) с перифразами, тот же размер корпуса	decision tree	0.434 F1
а) с перифразами, максимально доступный трейн, тот же размер теста	naive Bayes	0.243 F1
б) распознавание внутри пар	SVC	0.77 acc

без перифраз

	classifier	f1	accuracy	precision	recall
0	LogisticRegression	0.058252	0.720461	0.857143	0.030151
1	GaussianNB	0.346774	0.533141	0.289562	0.432161
2	KNeighborsClassifier	0.102767	0.672911	0.240741	0.065327
3	SVC	0.029703	0.717579	1.000000	0.015075
4	DecisionTreeClassifier	0.301994	0.646974	0.348684	0.266332
5	random	0.269542	0.609510	0.290698	0.251256

перифразы + тот же размер корпуса

	classifier	f1	accuracy	precision	recall
0	LogisticRegression	0.057416	0.716138	0.600000	0.030151
1	GaussianNB	0.348000	0.530259	0.289037	0.437186
2	KNeighborsClassifier	0.044843	0.693084	0.208333	0.025126
3	SVC	0.029126	0.711816	0.428571	0.015075
4	DecisionTreeClassifier	0.331250	0.691643	0.438017	0.266332
5	random	0.291005	0.613833	0.307263	0.276382

перифразы + увеличенный трейн

	classifier	f1	accuracy	precision	recall
0	LogisticRegression	0.013158	0.783862	1.000000	0.006623
1	GaussianNB	0.241758	0.502882	0.180921	0.364238
2	KNeighborsClassifier	0.057143	0.762248	0.208333	0.033113
3	SVC	0.000000	0.780980	0.000000	0.000000
4	DecisionTreeClassifier	0.157480	0.691643	0.194175	0.132450
5	random	0.194444	0.665706	0.204380	0.185430

классификация в парах

	classifier	accuracy
0	LogisticRegression	0.729885
1	GaussianNB	0.442529
2	KNeighborsClassifier	0.333333
3	SVC	0.701149
4	DecisionTreeClassifier	0.402299
5	random	0.459770

+ ПРЕПРОЦЕССИНГ

без перифраз

	classifier	f1	accuracy	precision	recall
0	LogisticRegression	0.067633	0.721902	0.875000	0.035176
1	GaussianNB	0.346614	0.527378	0.287129	0.437186
2	KNeighborsClassifier	0.018868	0.700288	0.153846	0.010050
3	SVC	0.029703	0.717579	1.000000	0.015075
4	DecisionTreeClassifier	0.296703	0.631124	0.327273	0.271357
5	random	0.291005	0.613833	0.307263	0.276382

перифразы + тот же размер корпуса

	classifier	f1	accuracy	precision	recall
0	LogisticRegression	0.092166	0.716138	0.555556	0.050251
1	GaussianNB	0.351779	0.527378	0.289902	0.447236
2	KNeighborsClassifier	0.056604	0.711816	0.461538	0.030151
3	SVC	0.046948	0.707493	0.357143	0.025126
4	DecisionTreeClassifier	0.433735	0.729107	0.541353	0.361809
5	random	0.248619	0.608069	0.276074	0.226131

+ 0.086

перифразы + увеличенный трейн

	classifier	f1	accuracy	precision	recall
0	LogisticRegression	0.026144	0.785303	1.000000	0.013245
1	GaussianNB	0.243478	0.498559	0.181230	0.370861
2	KNeighborsClassifier	0.012579	0.773775	0.125000	0.006623
3	SVC	0.000000	0.782421	0.000000	0.000000
4	DecisionTreeClassifier	0.179775	0.684438	0.206897	0.158940
5	random	0.164875	0.664265	0.179688	0.152318

классификация в парах

	classifier	accuracy
0	LogisticRegression	0.764368
1	GaussianNB	0.436782
2	KNeighborsClassifier	0.425287
3	SVC	0.770115
4	DecisionTreeClassifier	0.505747
5	random	0.436782

+ 0.04

Препроцессинг

- Переводим спецсимволы *html* в читаемый вид
- Заменяем имена пользователей на @username
- Ссылки и хештеги заменяем на #link и #hashtag соответственно
- Пунктуацию и эмодзи сохраняем
- Не лемматизируем

Аугментация

Саркастичных твитов примерно в 3 раза меньше, чем не-саркастичных:

- попробуем заменить значимые слова (>3 символов) на синонимы из WordNet
- и замаскировать часть слов, заменив их на [unk] (для Берта)
- ! опасность испортить данные (важно для задач, связанных с семантикой)

Пример замены:

- *The only thing I got from college is a caffeine addiction*
- *→ The lone thing I got from #unk is a caffein dependence*

Аугментация

Увеличение количества саркастичных примеров за счет замены слов на синонимы ожидаемо дает прирост качества. На корпусе из 1734 саркастичных и 2601 не-саркастичных (без перифраза):

	classifier	f1	accuracy	precision	recall
0	LogisticRegression	0.593128	0.740484	0.803922	0.469914
1	GaussianNB	0.650575	0.649366	0.543186	0.810888
2	KNeighborsClassifier	0.257919	0.621684	0.612903	0.163324
3	SVC	0.635879	0.763552	0.836449	0.512894
4	DecisionTreeClassifier	0.611594	0.690888	0.618768	0.604585
5	random	0.420438	0.542099	0.428571	0.412607

+ 0.3

Трудности

Дизайн эксперимента для оценки добавления перифраз:

- трейн корпус: увеличенного размера
  или такого же размера с перифразами в нужном соотношении
- тест корпус: ровно тот же самый, что в варианте без перифраз
  или с перифразами, но такого же размера, что в первом варианте

Метрики распознавания в парах:

- accuracy (a.k.a. нашёл верно / неверно)
- или индекс саркастичного текста в паре
- или ??? (в оригинальном соревновании как-то получают f1)

Решение собственно языковой задачи vs. решение соревнования – от этого зависит, фича или баг наши неязыковые элементы (имена пользователей, ссылки, теги)

Дальнейшие действия

- подумать, что делать с хештегами и мусором в перифразах
- определиться с (не)использованием перифраз
- попробовать аугментацию с использованием векторных представлений
- нейросетевая классификация:
  - CNN, LSTM – варианты эмбеддингов
  - BERT

Важные ссылки

Github-репозиторий проекта

Доска в Trello (ссылка-приглашение)