scolfield сегодня в 09:01

ИИ для прогнозирования тренда стоимости Bitcoin на данных Twitter. ч.1

Python *, Twitter API *, Big Data *, Искусственный интеллект

Криптовалюты уже давно стали средством инвестиции, а криптобиржы заняли прочную позицию относительно рынка ценных бумаг и Forex. Трейдеры и инвесторы хотят знать, что будет завтра, через неделю, через час. Технический анализ - хороший инструмент, но он не может предсказать эмоции и настроения людей. В тоже время социальные сети каждую секунду пополняются новыми статьями, публикациями, сториз и пр. И все это несет в себе и оптимизм, и панику и пр. настроения, влияющие на решения людей. Причем людей, пишущих в своих блогах про криптовалюты, мы будем разделять на просто людей и лидеров мнений. Так вот идея заключается в том, что происходящее на рынках является источников всей этой big data'ы постов и мнений в соц. сетях. НО! Но и наоборот все, что "новостится", обсуждается, поститься, пампится и дампится в соц. сетях влияет на рыночные котировки. Конечно киты о своих разворотах никогда никого не предупредят, на то они и киты. Но не исключено, что они как раз постараются опубликовать что-то, что подготовит почву для разворота. И та нейросеть, которую нужно построить, должна в идеале и такие сигналы интерпретировать корректно. Да и в принципе не так важно, для чего этот подход применять: для акций или криптовалют. Да и на последок, перед тем как заняться этой темой, я немного погуглил и понял, что идея конечно же не новая, и есть те, кто уже пробовал это сделать. С одной стороны это хорошо - значит кто-то еще в это верит, а с другой стороны - это не отвечает на вопрос, решаема ли задача.

В этой статье я расскажу о первой серии экспериментов для проверки гипотезы влияния данных Twitter на тренд стоимости Bitcoin. Цель не угадать ценник, а предсказать рост, убывание или относительную неизменность цены.

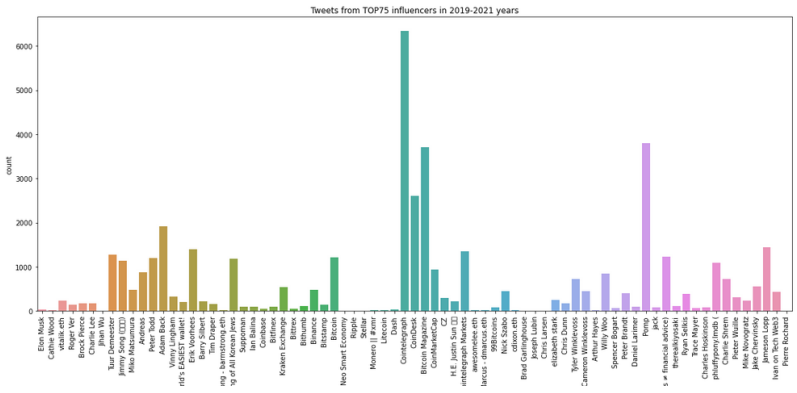
Я решил не анализировать все твитты всех пользователей, где упоминается биткойн, и ограничился лидерами мнений в количестве 75 шт. Лидеров мнений я отобрал, сматчив 2 статьи на тему влияния отдельных людей на курс битка. Вот список их учеток в Twitter (крипто энтузиасты должны узнать большинство по никнеймам):

```
@elonmusk, @CathieDWood, @VitalikButerin, @rogerkver, @brockpierce, @SatoshiLite,
@JihanWu, @TuurDemeester, @jimmysong, @mikojava, @aantonop, @peterktodd, @adam3us,
@VinnyLingham , @bobbyclee, @ErikVoorhees, @barrysilbert, @TimDraper,
@brian_armstrong,@koreanjewcrypto, @MichaelSuppo, @DiaryofaMadeMan, @coinbase,
@bitfinex, @krakenfx, @BittrexExchange, @BithumbOfficial, @binance, @Bitstamp,
@bitcoin, @NEO_Blockchain, @Ripple, @StellarOrg, @monero, @litecoin, @Dashpay,
@Cointelegraph, @coindesk, @BitcoinMagazine, @CoinMarketCap,@cz_binance,
@justinsuntron, @CointelegraphMT, @AweOlaleye, @davidmarcus, @99BitcoinsHQ,
@NickSzabo4, @cdixon, @bgarlinghouse, @ethereumJoseph, @chrislarsensf,
@starkness, @ChrisDunnTV, @tyler, @cameron, @CryptoHayes, @woonomic,
@CremeDeLaCrypto, @PeterLBrandt, @bytemaster7, @APompliano, @jack, @MatiGreenspan,
@theRealKiyosaki, @twobitidiot, @TraceMayer,@IOHK_Charles, @fluffypony,
@CharlieShrem,@pwuille,@novogratz, @jchervinsky, @lopp, @IvanOnTech, @pierre_rochar
```

По временному интервалу я ограничился тремя года (2019, 2020, 2021 гг.) Для реализации я использовал библиотеки Keras, язык Python, среда Jupyter notebook, все делал на своем личном

Парсинг Twitter. API Binance.

И так, данные Twitter. Первый челлендж был в том чтобы получить именно их. Twitter отказал 2 или 3 раза в ответ на мои запросы подключиться к их официальным API. При чем немного погуглив, я понял, что это нормальная практика, и я не один такой. К тому же при работе с официальными API Twitter'a есть неприятные ограничения, поэтому я не сильно расстроился. Пришлось "взять в зубы" Selenium, XPath и спарсить Twitter самостоятельно. Я делал это первый раз, но разобрался довольно быстро. Плюс очень боялся, что в процессе меня забанят и придется покупать ip и пр. Мне так говорили люди, которые уже не раз что-то парсили, и трудно было им не верить. Но вопреки всем страхам данные за 3 года (2019, 2020, 2021 гг.) я выкачал без банов примерно за 8 часов. На выходе получил 43 842 твиттов,12 Мб. По крупному хочу отметить только одну проблему. Когда Twitter перестал открываться в России, пришлось воспользоваться VPN. Благо на то время для обучения у меня уже был скачан основной 3-х годичный датасет, и мне нужно было допарсить январь и февраль 2022 г. Я пробовал два разных VPN, и у меня постоянно крашился DOM, и парсинг останавливался. Один мой товарищ, специализирующийся на парсинге, посоветовал мне включить headless режим (это когда парсинг происходит со скрытым браузером и мы не видим глазами имитацию действий). Headless mode сразу помог. Кстати с товарищем мы познакомились, когда я искал исполнителя для скрапинга(парсинга) на Яндекс услугах. В итоге когда мы начали обсуждать детали, пришло осознание, что YouTube и мои навыки программирования будут достаточны для того, чтобы я справился со всем сам. И еще, оказалось, что не так много людей, профессионально занимающихся парсингом, парсили хотя бы раз Twitter. На графиках ниже динамика твиттов по авторам за три года:



Далее нужно было найти исторические данные изменения цены BTC, причем с гранулярностью 1ч. Единственное, где я смог это достать - оказались API известной крипто-биржи Binance (https://github.com/binance/binance-public-data/). Очень понятно и очень быстро. Спасибо Binance.

Конкатенация и сопоставление данных Twitter и Binance

И так, гранулярность, с которой я решил начать эксперименты = 1ч. А значит твитты также были распределены по часам. Итого за три года получилось: 26246 записей. Как вы догадались данные твиттов также нелинейно растянулись по временной шкале 3-х лет. И так как в одном часе могло быть пусто, а в другом наоборот написал блогер, да еще не один, да еще не одному посту, то твитты внутри часа нужно было как-то конкатенировать. И конкатенировать их внутри часа я решил по следующему правилу: автор1::: твитт_автора1 автор2::: твитт_автора2 автор3::: твитт_автора3 и т.д. Идея с ":::" заключается в том, что нейронка рано или поздно должна начать выделять на основе своих эмпирических вычислений авторов от своих твиттов. Также во избежание шума те авторы, чье количество твиттов с упоминанием BTC было < 20 за 2019-21 гг. были обезличены, т.е. их никнеймы заменены на OTHER_less20. Туда чуть было не угодил Илон Макс, т.к. оказалось, что его твиттов с упоминанием BTC было не сильно выше порога, всего 37. Он чудом сохранил индивидуальность:) Но это всего лишь гипотезы, как и вся идея, описанная в этой статье. И правило ":::" и "<20" можно и нужно менять. И еще одна важная деталь про датасет. Я сопоставил твитты T против котировок T+1. Т.е. предполагается, что нейронка по окончании часа T будет предсказывать тренд на конец часа T+1. Опять же момент для калибровки, плюс вообще никто не говорит, что час - это единственная атомарная величина. Можно пробовать с днями и другими гранулами времени.

Перевод текстов твиттов в цифры, а цифры в OHE

Дальше пошла подготовка данных к обучению. Нейросети не принимают на вход слова и символы. А значит сконкатенированные твитты нужно было преобразовать в индексы. Я воспользовался керасовским Tokenizer'ом и создал словарь, задав при этом максимальный индекс=15000. Вначале я пробовал 10000, затем 15000. Разницы в финальных результатах после обучения нейронки выявлено не было. При этом Tokenizer обнаружил 47 742 уникальных значения. Далее в целях нормализации я преобразовал индексы в OneHotEncoding (OHE). Таким образом в качестве X я получил матрицу 26246X15000 со значениями 0 и 1. А в качестве Y

образом в качестве X я получил numpy матрицу 26246X15000 со значениями 0 и 1. А в качестве Y

получилась матрица 26246X3. Почему 3 ? Потому что 3 значения было решено использовать для изменения тренда:

- 0: больше -10$ и меньше +10$
- 1: меньше -10$
- 2: больше +10$

Я решил не учитывать изменения меньшие 10$, потому что мне понравилось распределение:

```
количество 0 равно:  (5452,)
количество 1 равно:  (10612,)
количество 2 равно:  (10182,)
```

Ну и чтобы еще лучше это нормализовать, к Y был также применен OneHotEncoding. Еще раз для начинающих нейронщиков, которые читают эту статью: X - это то, что подадим на вход нейросети, а Y - то, что хотим получить на выходе. X, Y были разделены на тренировочную выборку Xtrain, Ytrain (24 806 строк) и проверочную Xtest, Ytest (1440 строк).

TimeseriesGenerator

Ну и последняя трансформация с данными перед тем, как врубить обучение. Т.к. мы имеем дело с временными рядами, то при помощи TimeseriesGenerator устанавливаем величину шага для анализа = 48 ч. Чтобы было понятней приведу фрагмент кода:

```
xLen=48    #48 hours step for analyze
valLen=1440#24hours*60days=1440hours for Test

trainLen=dataX.shape[0] - valLen #size of Train

xTrain, xTest = dataX[:trainLen],   dataX[trainLen+xLen:]
yTrain, yTest = dataTTRcategor[:trainLen], dataTTRcategor[trainLen+xLen:]

#train TimeSeriesGenerator
TRtrainDataGen = TimeseriesGenerator(xTrain, yTrain,
                                     length=xLen, stride=1, sampling_rate=1,
                                     batch_size=12)
#test TimeSeriesGenerator
TRtestDataGen = TimeseriesGenerator(xTest, yTest,
                                    length=xLen, stride=1,
                                    batch_size=12)
```

Как видно из кода батч = 12, т.е. веса при обучении модели будут меняться после 12-ти прохождений через нейронку. Резюме: когда начинается час X, то для предсказания тренда цены биткойна к концу часа X нейросеть будет анализировать твитты, написанные по тематике биткойна лидерами мнений в диапазоне с X-48 часа до X часа.

Нейросети и их обучение

Ну и наконец долгожданное обучение. Я пробовал три разных архитектуры:

- один полносвязный слой;
- полносвязные + сверточные слои;
- UNET сеть;

Обучение 50-ти эпох для каждой занимает в среднем по 3-3,5 часа.

Полносвязная сеть

Слои: Dense, Flatten

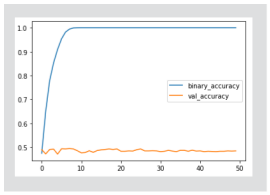
Число параметров: 1 014 503

Архитектура:

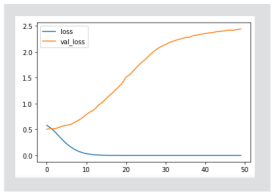
```
modelD = Sequential()
```

```
modelD.add(Dense(100,input_shape = (xLen,maxWordsCount), activation="relu" ))
modelD.add(Flatten())
modelD.add(Dense(dataTTRcategor.shape[1], activation="sigmoid"))

#Compile
modelD.compile(loss="binary_crossentropy", optimizer=Adam(learning_rate=1e-4), metr
```

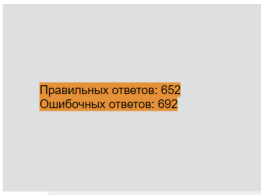
Результаты (синим - тренировочная выборка, желтым - проверочная):



точность



ошибка



предсказание тренда на проверочной выборке

Сверточная сеть

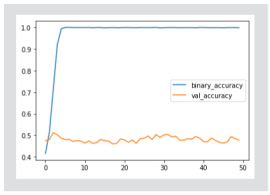
Слои: Dense, FlattenDense, Flatten, RepeatVector, Conv1D, MaxPooling1D, Dropout

Число параметров: 1 100 523

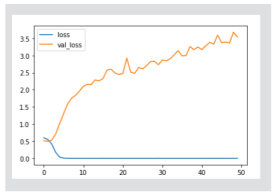
Архитектура:

```
drop = 0.4
input = Input(shape=(xLen,maxWordsCount))
x = Dense(100, activation='relu')(input)
x = Flatten()(x)
x = RepeatVector(4)(x)
x = Conv1D(20, 1, padding='same', activation='relu')(x)
x = MaxPooling1D(pool_size=2)(x)
x = Flatten()(x)
x = Dense(100, activation='relu')(x)
x = Dropout(drop)(x)
x = Dense(dataTTRcategor.shape[1], activation='sigmoid')(x)
modelUPD = Model(input, x)
```

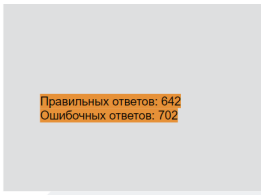
Результаты (синим - тренировочная выборка, желтым - проверочная):



точность



ошибка



предсказание тренда на проверочной выборке

UNET сеть

Слои: Dense, FlattenDense, Flatten, RepeatVector, Conv1D,MaxPooling1D, Dropout, BatchNormalization

Число параметров: 1 933 155

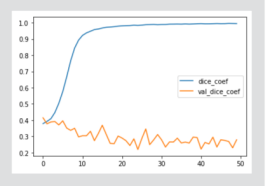
Архитектура:

```
img_input = Input(input_shape)

    # Блок 1
    x = Conv1D(32 * k , 3, padding='same')(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv1D(32 * k, 3, padding='same')(x)
    x = BatchNormalization()(x)
    block_1_out = Activation('relu')(x)
```

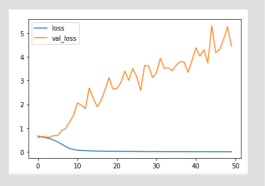
```
# Блок 2
x = MaxPooling1D()(block_1_out)
x = Conv1D(64 * k, 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv1D(64 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
block_2_out = Activation('relu')(x)

# Блок 3
x = MaxPooling1D()(block_2_out)
x = Conv1D(128 * k, 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv1D(128 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv1D(128 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
block_3_out = Activation('relu')(x)

# Блок 4
x = MaxPooling1D()(block_3_out)
x = Conv1D(256 * k, 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv1D(256 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv1D(256 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
block_4_out = Activation('relu')(x)
x = block_4_out

# UP 2
x = Conv1DTranspose(128 * k, kernel_size=3, strides=2, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = concatenate([x, block_3_out])
x = Conv1D(128 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv1D(128 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

# UP 3
x = Conv1DTranspose(64 * k, kernel_size=3, strides=2, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = concatenate([x, block_2_out])
x = Conv1D(64 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv1D(64 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

# UP 4
x = Conv1DTranspose(32 * k, kernel_size=3, strides=2, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = concatenate([x, block_1_out])
x = Conv1D(32 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv1D(32 * k , 3, padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
```

```
    x = Flatten()(x)
    x = Dense(dataTTRcategor.shape[1], activation='sigmoid')(x)

    model = Model(img_input, x)

    model.compile(optimizer=Adam(learning_rate=1e-4),
                  loss='binary_crossentropy',
                  metrics=[dice_coef])
```

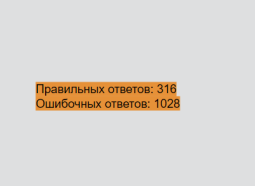
Результаты (синим - тренировочная выборка, желтым - проверочная):



точность



ошибка



предсказание тренда на проверочной выборке

Ну вот пока так. Удовлетворяющего результата пока нет. Признаться честно я возлагал очень не малые надежды на UNET. Эта архитектура считается в кругах опытных нейронщиков, как best practice. Но, как видите, результаты она дала еще более дикие, чем обычная полносвязка, которая тоже в свою очередь пока дает эффект, равносильный бросанию монетки. Нейронками я увлекаюсь не так давно, поэтому в планах попробовать применить: Q-learning, Сети с вниманием, Генетические алгоритмы и др. Не обещаю, что вторая часть выйдет скоро, но триггером для ее написания должна стать положительная динамика в результатах.

Больше всего хотелось бы получить советы и мнения от нейронщиков и датасаентистов, решавших что-то близкое и не очень, опытных и не совсем. Но также буду благодарен любой обратной связи в комментариях!

Если есть идеи по сотрудничеству и/или, как допилить эту идею)) - велкам в личку! Доведя нейронку до ума, можно будет подумать на тему SaaS продукта. Пару слов о себе: я продакт/проджект в банкинге с ИТ бэкграндом > 17 лет. Программирование, нейронки, хакатоны - это все мои хобби.

Всем спасибо, что дочитали до конца!

Теги: deep learning, machinelearning, python, parsing data, neural networks, twitter, bitcoin, cryptocurrency, trading, prediction

Хабы: Python, Twitter API, Big Data, Искусственный интеллект

4 Карма 2 Рейтинг
@scolfield
Пользователь

Комментарии 4

ПОХОЖИЕ ПУБЛИКАЦИИ

23 сентября 2021 в 16:49
Triton: Open Source язык для ядер Deep Learning
+9 3.6K 19 0

4 мая 2021 в 15:16

Facebook Prophet + Deep Learning = NeuralProphet

+5 3.4K 20 0

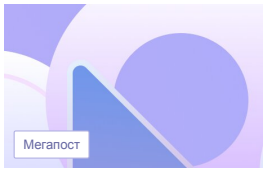
2 ноября 2018 в 07:05

Transfer Learning: как быстро обучить нейросеть на своих данных

+9 30K 98 8 +8

МИНУТОЧКУ ВНИМАНИЯ

Разместить



ИТ-рынок сегодня: наём, зарплаты, прогнозы

Экспресс-тур по городу искусственного интеллекта

Четыре всадника самореализации в IT

КУРСЫ

Основы языка Python
20 апреля 2022 · 15 000 ₽ · GeekBrains

Программирование на Python для детей
28 апреля 2022 · 14 700 ₽ · GeekBrains

Углубленный курс по Python
15 мая 2022 · 55 080 ₽ · GeekBrains

Data Science Bootcamp
18 апреля 2022 · 285 000 ₽ · Elbrus Coding Bootcamp

Python-разработчик
19 апреля 2022 · 100 000 ₽ · Яндекс Практикум

Больше курсов на Хабр Карьере

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

сегодня в 06:08

Почему все врут, правда о кривде

+37 10K 33 58 +58

вчера в 13:53

[Личный опыт] Турция: как здесь живется сейчас

+32 20K 53 52 +52

вчера в 20:10

Возможна блокировка Википедии в России, — предупредил участник Википедии Станислав Козловский

+30 7K 14 49 +49

сегодня в 08:44

Сейчас плохо, но все может быть еще хуже

+25 9K 17 16 +16

вчера в 20:50

5 необычных клавиатур весны 2022 года: современные геймерские, ретро и клавиатура на основе Raspberry Pi

+25 4.5K 8 20 +20

Сортируем квартиры по фотографиям: как мы сделали фильтр «бабушкин ремонт»

Турбо