ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет экономических наук

Медведев Александр Игоревич

«Моделирование поведения розничных инвесторов на основе социальных сетей»

Выпускная квалификационная работа - МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
по направлению подготовки 38.04.01 Экономика

Образовательная программа "Прикладная экономика"

Рецензент
доцент, к.э.н.
Креховец Екатерина
Владимировна

Руководитель
Доцент, к.э.н.
Мамедли Мариам
Октаевна

Москва 2022

Аннотация

В настоящее время фондовые рынки являются важным инструментом развития экономики и распределения ресурсов между отраслями. Получили популярность платформы позволяющие частным инвестором легко торговать на фондовых рынках, используя удобные мобильные приложения.Исходя из практики можно сделать вывод, что моделирование отношения розничных инвесторов к акции может быть полезным для понимания и прогнозирования фондовых рынков.Цель этой работы — проанализировать значимость индекса настроения розничных инвесторов, полученного из социальных сетей для предсказания будущих цен на выбранные акции с использованием моделей машинного обучения. В работе была проверена гипотеза о значимости влияния индекса тональности на доходности рассмотренных компаний, для 6 из 7 рассмотренных компаний влияние значимо. Для всех компаний тест Грейнджера показывает причинность для доходностей хотя бы одного лага индекса тональности. Также полученный автором классификатор на основе архитектуры BERT превосходит стандартные подходы с точки зрения accuracy.

Оглавление

Введение

В настоящее время фондовые рынки являются важным инструментом развития экономики и распределения ресурсов между отраслями. Хотя в России в последние годы не наблюдались темпы быстрого экономического роста, объем торгов на Московской бирже в области фондового рынка вырос более чем в три раза за период с 2016 года по 2021 (Сайт MOEX, 2022).

Также стоит заметить активный рост и развитие сектора частных инвестиций в России и в мире в последние годы. Получили популярность платформы позволяющие частным инвестором легко торговать на фондовых рынках, используя удобные мобильные приложения. Например, большинство банков в России создали приложения для инвесторов такие как “Тинькофф Инвестиции”, “Сбербанк Инвестиции”, в Соединенных Штатах Америки получило широкое распространение приложение “Robinhood”.

Также согласно статистике опубликованной Московской биржей мы можем наблюдать увеличение количества розничных инвесторов более чем в два раза по сравнению с аналогичными данными 2020 года, до более чем 10 % населения России (около 17 миллионов человек) на конец 2021 года, а в свою очередь их объем вложений составил 6 триллионов рублей, против 2,6 в 1 квартале 2019 года (Сайт MOEX, 2022).

В последнее время все чаще стали появляться научные работы, изучающие влияние частных инвесторов на рынок акций, например в работе (van der Beck, 2021) делается вывод что неэластичный институциональный спрос позволяет частным инвесторам создавать непропорциональные изменения в стоимости активов.

Также в последнее время получили распространение случаи, когда частные инвесторы были способны манипулировать ценами акций искусственно подогревая их стоимость и координируясь в социальных сетях, наиболее известен кейс компании “GameStop”, акции которой значительно выросли вопреки прогнозам, благодаря боль-

шому количеству частных инвесторов, координирующихся по средствам социальной сети “Reddit” (How GameStop found itself at the center of a groundbreaking battle between Wall Street and small investors, 2021). Также на эту тему публиковались научные работы, например (Semenova, Winkler, 2022), в этой статье представлен эмпирический и теоретический пример того, как «ажиотаж» среди розничных инвесторов может вызывать значительные колебания цен на активы. В исследованиях данной тематики используется следующий подход: отношение инвестора к акции или же сентимент можно получить, анализируя посты инвесторов в социальных сетях. Агрегируя сентимент инвесторов за день по отношению к определенной акции, создается индекс отношения инвесторов к акции, далее применяется эконометрический подход к определению значимости полученного индекса. Это и другие исследования будут в последствие более широко раскрыты в обзоре литературы.

Исходя из вышеописанных исследований и практики можно сделать вывод, что моделирование отношения розничных инвесторов к акции может быть полезным для понимания и прогнозирования фондовых рынков. Особенно важным становится понимание и моделирование фондовых рынков в периоды серьезных изменений во всех сферах жизни, в такое время понимание динамики цен инструментов доступных на фондовой бирже могут помочь сохранить капитал и избежать серьезных потерь. Например, 24 февраля 2022 года произошел крупнейший обвал рынка акций в истории РФ (Обвал рынка акций РФ в четверг стал рекордным за всю историю, 2022) (Российский фондовый рынок упал на 11 % после начала операции в Донбассе, 2022) на фоне политических новостей.

Стоит заметить, что множество людей предполагало такое развитие событий, в социальных сетях и специализированных источниках новостей обсуждалось высокая вероятность реализации политических рисков.

Вследствие чего можно сделать вывод что в современной обстановке, когда новостная повестка изменяется постоянно, люди выкладывают огромное количество постов со своими мыслями об акциях, за

всеми новостями уследить практически невозможно, необходим автоматизированный подход к определению сентимента (отношения) инвестора к определенному инструменту.

Ограничения исследования. В работе моделируется отношение частных инвесторов к акциям выбранных компаний на основе постов в самой популярной социальной сети для инвесторов в России – "Тинькофф Пульс", а также выборка дополнена постами из каналов, публикующих инвестиционную аналитику в мессенджере "Telegram". Данные социальные сети выбраны не случайно, в отличие от иных социальных сетей, на рассматриваемых платформах люди публикуют свои мысли именно о инвестициях, что редко можно встретить в иных социальных сетях, например в Твиттере, где в выборку неизбежно попадут посты, не относящиеся напрямую к биржевым инструментам. Тинькофф Пульс имеет миллионы пользователей и является одним из главных мест для обсуждения фондового рынка в России (Соцсети «Пульс» — 2 года: 1,4 млн пользователей, 3 млн постов, 14 млн комментариев и 19 млн лайков, 2021), также каждый пользователь имеет инвестиционный счет, что повышает релевантность данных. Актуальность исследования - необходимость анализа большой выборки текстовых данных для моделирования сентимента частных инвесторов, и использование ее для улучшения прогнозов цен акций.

В работе исследуется влияние розничных инвесторов на следующие российские акции: SBER, GAZP, YNDX, VTBR, GMKN, LKOH, TCS. Данные компании выбраны, так как они входят в 10 самых популярных бумаг среди российских частных инвесторов по состоянию на первый квартал 2022 года (Сайт MOEX, 2022). Данные стоимости акций, а также посты в социальных сетях взяты за период с начала 2020 года по апрель 2022 года.

Постановка задачи. Цель этой работы — проанализировать значимость индекса настроения розничных инвесторов, полученного из социальных сетей для предсказания будущих цен на выбранные акции с использованием моделей машинного обучения.

Основные задачи, решаемые в ходе исследования:

1. Получение данных из социальных сетей, данных биржевых котировок рассматриваемых компаний
2. Ручная разметка данных из социальных сетей
3. Обучение модели BERT на полученных данных
4. Получение прогнозов, поиск наиболее неуверенных прогнозов модели, доразметка данных и добучение модели
5. Использование финальных прогнозов модели для построения индекса настроения
6. Проведение статистических тестов для определения значимости индекса для прогнозов цен на акции

Гипотеза исследования - наличие значимой взаимосвязи между индексом тональности инвесторов полученного из социальных сетей и ценами активов на фондовом рынке.

Научная новизна и практическая ценность. В данный момент для российского рынка акций практически отсутствуют работы на данную тематику с использованием глубокого обучения, преимущество данных архитектур в определении тональности на настоящий момент доказано на большом количестве общедоступных наборов данных (Papers with code, URL).

В данной работе используется архитектура BERT (Devlin, et al., 2018), также происходит сравнение работы данной модели на тестовом наборе данных с некоторыми базовыми моделями, такими как TF-IDF. Доказывается преимущество нейросетевого подхода.

Более того в открытом доступе на данный момент отсутствуют предобученные модели для определения тональности текстов на финансовые темы. В работе предлагается модель BERT, обученная на собранном автором корпусе текстов.

Также для социальной сети “Тинькофф Пульс” отсутствует официальный API, который позволил бы легко получать последние посты, в работе автором предоставляется код для парсинга данных из данной платформы.

Также данное исследование при условии дальнейшего развития (такого как разметка большего числа данных для обучения классификатора) может быть использовано банковскими институтами для генерации инвестиционных идей.

Глава 1. Обзор литературы

В первой части данной главы будет обозрены статьи ранее исследовавшие затронутую тему, проведен их кратикий обзор, будет выделен подход к исследованиюб использовавшийся в данных работах, во второй части будет проведен краткий обзор моделей в настоящее время используемых для определения тональности текстовых данных, будет описан процесс обучения и тонкой настройки модели BERT используемой в исследовнии для определения тональности.

1.1. Литература посвященная анализу взаимосвязи между данными социальных сетей и котировками акций

1.1.1. Иностранная литература

Одной из первых работ в которой анализировалось влияние тональности на цены акций является исследование (Mittal, 2011). Автор исследует влияние твитов на индекс Dow Jones Industrial Average (DJIA).Примечательно, что автор оставляет в выборке только твиты, которые, скорее всего, содержат информацию об эмоциях. Например, собираются твиты, содержащие «I feel», «I'm». Атором были собраны данные из 476 миллионов твитов за период с июня по декабрь 2009 года. Важно отметить, что авторы собирали не только твиты, связанные с DJIA, а также выбирали и другие релевантные посты (упоминания компаний входящих в индекс).

Далее автором разрабатывается, специальный список слов (каждое слово относится к одному из четырех эмоциональных состояний: Спокойствие, Счастливое, Бдительное и Доброе) для расчета оценок тональности настроений постов. Далее строятся оценки для каждого твита, определяется к какому классу он принадлежит, и данные показатели аггрегируются на уроне дней и используются для построения индексов.

Из (Mittal, 2011) следует, что только «спокойные» (с лагом 1-5) и «счастливые» (с лагом 1) оценки тональнастости твита, согласно тесту Грейнджера вызывают изменение DJIA на уровне значимости 10 %.

Кроме того, автор сравниевает точность (accuracy) трех моделей (линейной регрессии, логистической регрессии (Berkson, 1944), SVM (Vapnik, 1995) и нейронной сети). В среднем линейная регрессия дает точность направления (т. е. будет ли доходность положительной или отрицательной) на уровне 66,6 %.

Чтобы не иметь дело с относительно сложными процедурами кросс-валидации, автором ни один из алгоритмов, включающих настройку параметров, не пользуется.

Важно отметить, что мы выбираем для своей работы тот же метод кросс-валидации для оценки ошибок модели, который предложен в сататье и навзван автором k-кратной последовательной кросс-валидацией (k-SCV). А также используем тест Гренджера для исследования причинности изменений акций компании от тональности настроения в социальных сетях.

К недостаткам работы можно отнести отсутствие сравнения полученных автором моделей с признаками тональности с моделями без них, что не позволяет сделать однозначных выводов о неслучайности результатов, а также важности признаков тональности.

Во все большем числе статей исследуется потенциальная ценность добавления в модели признаков, основанных на тональности настроения инвесторов, в дополнение к часто используемым в таких моделях наборов признаков, таких как например ценовые, технические, а также макроэкономические (Jin, et al., 2020) (Antweiler, 2004) (Chen, 2020).

Исследования показывают, что настроения в социальных сетях и других онлайн-платформах, связанных с инвестициями, отражают субъективное восприятие и характерный подход инвесторов по отношению к акциям (Kearney Liu, 2014). В исследовании (Pagolu et al., 2016), автор моделирует тональность на большом наборе данных твитов, связанных с акциями, и делает вывод о сильной корреляции между ценами и тональностью твитов. Работы (Jin et al., 2020) также про-

водят анализ тональности настроений по комментариям, связанным с акциями в Интернете, чтобы классифицировать их как бычьи или медвежьи, использовать эти данные для построения индекса тональности, чтобы использовать их в качестве дополнительных переменных в финансовых моделях. (Sonkiya et al., 2021) используют модель классификации на основе BERT для извлечения мнений из новостных статей о выбранных акциях. Затем они предсказывают будущие цены на акции, используя генеративную состязательную сеть с ценовыми, техническими и эмоциональными характеристиками, что значительно снижает ошибку на тестовой выборке по сравнению с моделями без признаков тональности.

В работе (Heston, 2017) авторы пришли к выводу, что позитивная тональность в новостях быстро повышает доходность акций, тогда как негативная вызывает более долгосрочную реакцию. (Li and Wu, 2022) используют классификатор на основе метода опорных векторов для прогнозирования направления доходности акций с входными функциями, состоящими из различных технических индикаторов и характеристик тональности настроений. Технические функции создаются с использованием библиотеки Python TA-Lib (mrjbq7, 2021), библиотеки с открытым исходным кодом для расчета технических индикаторов.

На основе вышеописанных работ, несмотря на то что зачастую авторы приходят к различным выводам, можно прийти к выводу об общей методологии проведения подобных исследований. Изначально собираются данные из социальных сетей, с помощью различных классификаторов проводится классификация тональности настроения с помощью некого метода классфификации, также можно заметить в более новых статьях, например (Sonkiya et al., 2021), (Li and Wu, 2022) авторы приходят к выводу что нейросетейвой подход к классификации текстовых данных позволяет получить лучшие оценки вероятности принадлежности тональности текста к определенному классу. Далее авторами используется эконометрический подход к определению важности признаков тональности для предсказания цен на фондовом рынке. Данная методология будет также использована в нашем иссле-

довании. Также в вышеописанных работах редко встречается сравнение модели с признаками тональности и без них, что будет проведено в этой работе.

Прогноз движения цен с использованием настроений инвесторов ставит перед исследователем проблему, а именно различную частотность признаков тональности настроений и цен акций. Новости и сообщения в социальных сетях распространяются неравномерно во времени, тогда как данные о ценах или доходностях доступны только с дневной частоностью. Поэтому нам необходимо брать цену или доходность с определенной периодичностью и агрегировать данные о тональности новостей или постов в социальных сетях в индекс с той же переодичностью.

В данной работе ежедневная доходность акций является целевой переменной, причину перехода от стоимости акций к доходности мы подробнее обсудим в главе с данными. Поэтому нам необходимо агрегировать эти настроения в подходящей форме для ежедневного измерения, чтобы ввести настроения в качестве независимых переменных в модели для прогнозов. Вышеописанные исследования выявили ряд методов для этого. Некоторые примеры из литературы включают вычисление логарифмического отношения бычьих и медвежьих сообщений в день (Antweiler, 2004), отношение экспоненциальных скользящих средних количества бычьих и медвежьих сообщений для уменьшения шума (Leow, et al., 2021) или просто соотношение количества положительных и отрицательных сообщений (Bollen, 2011). В нашей работе мы используем подход агрегирования, который включает в себя разницу между количеством постов с положительной и отрицательной тональностью и масштабирование ее по общему количеству постов в этот день, включая нейтральные посты, как это сделано в работе (Hiew, et al., 2019). Подробнее подход агрегирования можно рассмотреть в формуле (1.1).

$$\text{Sentiment Score } e_{t,s} = \frac{\text{Pos}_{t,s} - Neg_{t,s}}{\text{Pos }_{t,s} + Neu_{t,s} + Neg_{t,s}} \tag{1.1}$$

где $e_{t,s}$ - Тональность отношения к акции за определенное время;
$Pos_{t,s}$ - Количество позитивных постов за это время;
$Neg_{t,s}$ - Количество отрицательных постов за это время;
$Neu_{t,s}$ - Количество нейтральных постов за это время;

1.1.2. Российская литература

Хотя уже нами было приведено большое количество исследований данной темы для иностранных финансовых рынков, большинство из них ориентированы на развитые рынки, однако согласно исследованию (Bekaert, Harvey, 1995) можно сделать вывод о том, что анализ развивающихся фондовых рынков, таких как российский фондовый рынок, в силу таких особенностей таких как отличающиеся институты, высокая скорость развития и другие, может отличаться от анализа развитых рыноков. Из этого можео сделать вывод, что выводы описанные в иностранных исследованиях нельзя однозначно перенести на наш рынок, и требуется проведение исследования на российских данных.

Изучая работы ислледующие данную проблему на нашем рынке, такие как (Дубко, 2015),(Адрианова, 2018) можно сделать вывод, что авторы не предоставили численных выводов, изучив в своих работах только методологию проведения анализа цен рынка акций с использованием переменных полученных с помощью анализа настроения постов в социальных сетях

1.2. Статьи определяющие тональность данных с помощью BERT

BERT (Devlin, et al., 2018), представляет собой модель обработки естественного языка, предварительно обученную на очень больших массивах данных для предсказания скрытых слов, и обладающую способностью для возможной адаптации к иным задачам. Поскольку предварительно обученный BERT уже способен захватывать двунаправленный контекст из текста, для точной настройки модели для конкретной задачи NLP требуется лишь небольшой набор данных. Таким

образом, BERT является эффективным и действенным выбором в качестве модели для задач связанных с текстами, таких как задача классификации текстов использующейся в данной работе. В нашей работе мы применяем BERT для классификации тональности текста из социальных сетей и, следовательно, фокусируемся на литературе с аналогичными задачами. Человек способен определять тональность текста с довольно низкой точностью, то есть точность классификации тональности текста среднего человека, составляет всего 70–79 % (Gwet, 2014). Это указывает на то, что примерно 70-процентная точность, достигнутая в различных трех работах по классификации настроений твитов, связанных с акциями, можно считать успешным показателем (Pagolu et al., 2016). (Nansekin, Chen, 2020) изучают общественное мнение о 425 криптовалютах, используя классификатор на основе BERT с текстовыми данными, состоящими из сообщений на StockTwits, платформе социальных сетей, где трейдеры делятся идеями, данная платформа напоминает «Тинькофф пульс». В этом исследовании делается вывод о том, что добавление лексикона для предметной области в список токенов BERT способствует повышению эффективности классификации.В статье (Ortu, 2022) используют подход на основе BERT для классификации эмоций и настроений в комментариях GitHub и Reddit, связанных с биткойнами и Ethereum. Каждый комментарий в социальных сетях классифицируется на основе определенных эмоций, включая радость, гнев и печаль, а также общего настроения. (Pota et al., 2020) проводят тонкую настройку (fine-tuning) модели BERT для классификации тональностей твитов, сначала дополнительно предварительно обучая модель на задачу предсказания скрытых слов (MLM) на тексте, созданном из обильных данных Твиттера, чтобы модель изучила жаргон Твиттера, а затем дообучает модель на задачу классификации твитов. Предлагаемый подход превосходит другие традиционные модели.

Шапиро и др. (2020) изучают экономические настроения из различных источников новостей с 1980 по 2015 год с использованием модели на основе BERT с добавленным лексиконом для предметной области и обнаруживают тесную связь между извлеченными оценками

настроений и показателями настроений потребителей, а также экономическими потрясениями.

В нашем исследовании будет использован способ построения классификатора описанный в работе (Pota et al., 2020) сначала различные предобученные модели класса BERT будут дообучены на задачу MLM, на собранном автором наборе данных из социальных сетей (15 % слов маскируются, и модель пытается их предсказать). Данное этап обучения производится так как данные в «Тинькофф Пульсе» а также в экономических каналах Телеграма, отличаются от тех на которых тренировалась модель, следовательно необходимо ее дообучить понимать контекст слов на таком необычном для нее наборе данных. Далее дообученная на эту задачу модель будет дополнительно обучаться на задачу классификации текстов методом активного обучения на размеченном вручную наборе данных. Подробнее про модели, метод обучения и процесс тонкой настройки модели (fine-tuning) будет рассказано в следующей главе

Глава 2. Обзор подходов к решению NLP задач. Архитектура BERT

Данная глава рассказывает о существующих подходах к решению задач в области обработки естественного языка (NLP), проводит обзор развития подходов, описывает используемую в работе в качестве классификатора данных архитектуру BERT

2.1. Обзор развития подходов к решению NLP задач

Основная идея и область для развития в области обработки естественного языка - создание векторных представлений слов и текстов, так называемых эмбеддингов. Далее получив качественные эмбеддинги возможно применение любых уже существующих методов классификации, таких как линейные модели, метод опорных векторов и нейронные сети.

Изначально исследователи для создания векторных представлений текста использовали метод Bag of Words. Для проведения исследования таким образом создавался словарь всех слов встречающихся в выборке, потом для каждого текста создавался вектор длиной ранвой количеству слов в словаре, и считалось количество повторений каждого слова, получалось что элемент i вектора был равен количеству повторений слова i в тексте, а если слово ни разу не встречалось был равен нулю.

В дальнейшем, чтобы учесть тот факт, что для понимания значения текста различные слова представляют разную ценность, например предлоги практически не дают нам информации о значении текста исследователи стали использовать метод TF-IDF.

TF (частота терминов) указывает, как часто термин появляется в тексте, число, которое обычно нормализуется (обычно частота слов, деленная на общее количество слов в статье), чтобы предотвратить предпочтение длинных файлов (одно и то же слово может иметь боль-

шая частота слов в длинном файле, чем в коротком, независимо от того, важно слово или нет). TF представляется следующей формулой:

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{2.1}$$

Где $n_{i,j}$ обозначает количество раз слово t_i появляется в документе d_j

IDF (обратная частота документа) указывает на распространенность ключевых слов. Если у вас меньше документов, содержащих запись, тем больше IDF, а это означает, что термин имеет хорошую дифференциацию категорий. Формула IDF выглядит следующим образом:

$$IDF_i = \log \frac{|D|}{1 + |j : t_i \in d_j|} \tag{2.2}$$

Где $|D|$ количество документов в выборке, $|j : t_i \in d_j|$ - количество документов в которых встречается данное слово. Для того чтобы избежать деления на ноль когда t_i равно 0 , прибавляем 1 .

Итоговой статистикой является:

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D) \tag{2.3}$$

Таким образом модель дает высокий вес словам которые часто встречаются в данном тексте и редко в других. Из недостатков обоих методов можно выделить:

- Большую сложность вычислений (Так как размерность векторов получается большой, а сокращение приведет к потери информации о редковстречающихся словах)
- Отсутствие моделирования контекста, в различном контексте слова могут иметь различное значение
- Отсутствие моделирования похожести слов, итоговая модель не сможет понимать что два текста имеют синонимичное значение

Не смотря на это данная модель будет использоваться в работе для сравнения результатов классификации с классификатором основанном на архитектуре BERT.

Далее иследователи стали использовать подход, который позволил создавать эмбеддинги слов, векторы относительно небольшой размерности (в большинстве случаев до 1000, в то время как если мы будем создавать вектор для слова с помощью Bag of Words, мы получим вектор размерности количества слов в словаре с нулями везде кроме данного слова). Первый популярный подход такого плана являлся нейросетью Word2Vec (Mikolov, 2013). Отличительной чертой эмбеддингого подхода является обучение эмбеддингов таким образом, что косинус угла между векторами двух похожих слов стремится к 1.

$$cosinesimilarity = \left(1 - \frac{u \cdot v}{\|u\|_2 \|v\|_2}\right) \tag{2.4}$$

Модели данного плана обучаются на огромных корпусах слов, полученных из Википедии, и других источников, с задачей предсказания является ли пара слов стоящими рядом в предложении. Векторные представления слов полученные с помощью обученного на большом корпусе слов Word2Vec обладают желаемым свойством таким что векторы похожих по значению слов близки с точки зрения косинуса угла между ними.

Так широко известен пример, что если вычесть из вектора «королева», полученного с помощью Word2Vec, вектор «женщина» и добавить вектор «мужчина», то полученный вектор будет наиболее похож на вектор «король» с точки зрения косинуса угла между векторами (все векторы слов одной длины, что позволяет производить арифметические операции между ними).

Хотя и эта модель решает часть проблем, которые возникали с предыдущими моделями, она все еще не учитывает то, что в различном контексте у слов могут быть разные значения, что является несомненно важным в нашей задаче, так как некоторые слова в финансовом контексте могут иметь совершенно иное значение, чем в обычных условиях.

2.2. Перенос обучения (Transfer Learning)

Однако как было ранее указано, чтобы получить качественные векторные представления слов необходим большой тренировочный на-

бор данных. Так как в большинстве случаев набор данных для конкретной исследовательской или практической задачи не является настолько обширным чтобы обучить качественную модель, например исследователь может столкнуться с проблемой, что в тренировочной выборке могут отсутствовать некоторые языковые паттерны и выражения, которые будут встречаться в тестовой выборке, тем самым произойдет переобучение модели, и ее качество на новых данных будет страдать.

Соответственно на данный момент в NLP используется принцип transfer learning (перенос обучения), он состоит в том, что модель предобучается на большом наборе данных на одну или несколько обширных задач, а в последствие предобученная модель дообучается на сравнительно небольшом наборе данных доступных иследователю (как это происходит в нашей работе) на конкретную задачу. Важным отличием от классического подхода к обучению моделей является тот факт, что те веса, что были получены в ходе минимизации функции потерь нейронной сети на исходной задачи, сохраняются и используются как стартовая точка для оптимизации весов нейронной сети под конкретную задачу. Данный подход позволяет модели лучше отражать всю совокупность текстов на которую она была изначально обучена и не переобучаться на небольшой набор данных доступный в исследованиях узких прикладных задач.

При использовании переноса обучения предварительно обученная модель строится путем обучения модели нейронной сети на огромном количестве общих данных, которые в случае NLP обычно представляют собой неструктурированный неразмеченный текст из Интернета. Затем предварительно обученная модель настраивается на размеченных данных, относящихся к конкретной задаче, которая разделяет некоторые знания с первичной моделью. Этот метод обучения был широко распространен в исследованиях связанных с компьютерным зрением в течение многих лет.

Для переноса обучения в обработке естественного языка обычно используется процесс предварительного обучения языковой модели. Задача языковых моделей, как правило, состоит в том, чтобы предска-

зать следующее слово или предложение в тексте. Языковые модели обладают двумя важными свойствами, которые делают их хорошим выбором для использования в качестве базовой модели. Во-первых, их можно обучать на неразмеченных данных и, следовательно, использовать огромное количество данных, доступных в Интернете для обучения, например статьи из Википедии, доступные в открытом доступе книги. Во-вторых, хорошо обученная языковая модель может дать нам хорошее понимание структуры и семантики языка, что необходимо и полезно для решения последующих задач, таких как классификация текстов рассматриваемая в нашем исследовании.

В современных языковых моделях созданных для использования с transfer learning весь текст отображается в последовательность векторных представлений слов, что означает, что каждое слово представляется вектором в соответствии с его контекстом. Для transfer learning в NLP используется множество архитектур. Например, ELMo использовала двунаправленный LSTM. Но наиболее популярны архитектурны трансформеров, такие как BERT и GPT-3.

2.3. Выбор модели

Рассматривая задачу определения тональности сообщений из социальных сетей и не только, можно увидить что в данный момент лучшие результаты показывают различные реализации архитектур BERT и GPT (Papers with code, URL), исходя из этого для нашей работы мы будем использовать архитектуру BERT. Несмотря на то, что GPT-3 на некоторых наборах данных получает лучшие метрики, она недоступна в открытом доступе, а также ее обучение значительно более затратно с точки зрения вычислительных ресурсов, что не позволяет использовать ее в нашем эксперименте.

Языковые модели этого типа, в отличии от создания статических векторных представлений слов, такие как создаются с помощью Word2Vec и GloVE, создают динамические векторные представления зависящие от контекста. Например, слово «среда» в двух предложениях «Среда обитания» и «Среда лучший день недели» имеет два совер-

шенно разных значения, но вектор этого слова обоих предложениях в word2vec одинаковый.

Также стоит затронуть важный аспект переноса обучения, такой как исходная предобученная модель должна быть обучена на корпусе текстов напоминающем данные на которых будет происходить обучение на конечную задачу. Большинство моделей BERT для русского языка описано на научных и литературных корпусах текста, однако присутствует в открытом доступе модель «DeepPavlov/rubert-base-cased-conversational», которая была обучена на корпусе текстов из социальных сетей, субтитров. Данную модель мы будем использовать как исходную в нашей задаче, так как язык в наших данных ближе к разговорному, чем к литературному или научному. Для того чтобы исходная модель имела большую способность к моделированию финансового языка она будет дообучена на задачу предсказания скрытого слова (оригинальная задача на которую обучаются перобученные модели BERT) на корпусе текстов собранных автором, как было описано обзоре литературы.

2.4. Модель BERT

Модели NLP были основаны на сложных рекуррентных или сверточных нейронных сетях, которые включали в себя кодировщик и декодер. Наиболее эффективные модели также соединяли кодировщик и декодер через механизм внимания. Исследователи из Google (Devlin, et al.,2018) предложили новую простую сетевую архитектуру, Transformer, основанную исключительно на механизмах внимания, полностью исключающую свертки и рекуррентные слои.Эксперименты с двумя задачами машинного перевода показывают, что эти модели превосходят по качеству, в то же время они более распараллеливаемы и требуют значительно меньше времени для обучения

2.4.1. Основная идея модели

BERT — это модель, которая побила несколько рекордов по тому, насколько хорошо модели справляются с языковыми задачами. Вскоре

после выпуска документа с описанием модели команда также открыла исходный код модели и сделала доступными для загрузки версии модели, которые уже были предварительно обучены на массивных наборах данных. Это важное событие, поскольку оно позволяет любому, кто создает модель машинного обучения, включающую обработку языка, использовать эту мощную модель в качестве легкодоступного компонента, экономя время, энергию, знания и ресурсы, которые ушли бы на обучение модели обработки языка с нуля.

Обучение для конкретной задачи происходит в три этапа:

1. Предобучение на большом массиве данных, на задачу предсказания скрытого слова. В нашем случае мы выбираем предобученную модель доступную в библиотеке transformers. Данная модель способна предсказывать вероятности пропущенного слова в предложении.

2. Fine-tuning модели для собранных данных для исследования, задача такая же как на первом этапе, однако для повышения качества работы моделя возможно заморозка некоторых слоев согласно стратегии, чтобы избежать потери информации полученной на первом этапе. Это делается так как на первых слоях содержится более обобщенная информация об языковой структуре (Vucetic et. all, 2022). Для обучения мы используем весь набор собранных нами данных, маскируя случайным образом 15% слов. На данном этапе мы также получаем модель которая способна предсказывать вероятности пропущенных слов, однако более ориентированную на наши данные. Например в фразе «Продавайте, скоро акции полетят MASK», вместо MASK модель с 91.2% процентной вероятностью предсказывает слово «вниз»

3. Обучение классификатора, в полученную после второго этапа модель добавляется линейный слой, который на вход принимает усредненное значение векторов слов в тексте, полученные из модели обученной на втором этапе, а на выходе предсказывает вероятность принадлежности к классу, в нашем случае положительному, отрицательному или нейтральному

На Рисунке 2.1, мы можем видеть наглядно схему работы итоговой модели, полученной после выполнения трех шагов.

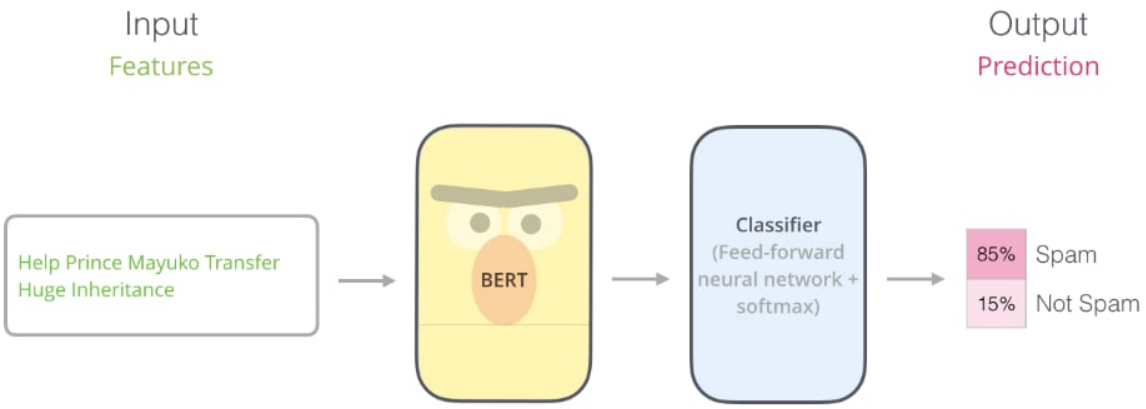

Рис. 2.1. Источник: The illustrated BERT (Alammar, 2018)

Более подробно каждый из этапов обучения будет расписан в следующих пунктах.

2.4.2. Предобучение модели

Исходные модели BERT предварительно обучаются с использованием двух задач обучения без учителя, описанных в этом разделе.

Основная идея BERT - глубокая двунаправленная модель более эффективна, чем либо модель слева направо, либо неглубокая конкатенация моделей слева направо и справа налево. К сожалению, стандартные условные языковые модели можно обучать только слева направо или справа налево, поскольку двунаправленное обусловливание позволит каждому слову косвенно «увидеть себя», а модель может тривиально предсказать целевое слово в многослойном многоуровневом коде. контекст. Чтобы обучить глубокое двунаправленное представление, мы просто случайным образом маскируем некоторый процент входных токенов, а затем предсказываем эти замаскированные токены.

На вход для обучения модели подаются тексты. В процессе обучения модели каждому слову присваивается своя кодировка в виде уникального числа, данные кодировки называются токенами, добавляются специальные токены обозначающие начало и конец предложения. Также устанавливается максимальная длина текста (максималь-

ное количество токенов), тексты длиннее обрезаются, те тексты что короче дополняются специальным padding токеном, это производится для того чтобы проводить математические операции с векторами текстов, для этого необходима одинаковая размерность входных векторов. Согласно в статье исследователей из Google, в которой был предложен и описан BERT, оптимальная длина текста 512 токенов (Devlin, et al.,2018).

Далее 15 % токенов заменяются на специальный MASK токен, на данном этапе задача модели - предсказание этого токена. Модель предсказывает вероятности каждого слова из всех попавших в выборку находится на месте замаскированного слова. Для того чтобы модель как можно лучше моделировала язык на данном этапе модель обучают на большом количестве данных, для этого требуются большие вычислительные мощности - сервер с большим количеством графических ускорителей. Источники данных на которых обучена наша исходная модель были описаны ранее.

Вторая же задача на которой предобучается BERT - предсказание следующего предложения. Обучение происходит похожим методом, однако в данном случае создается набор данных из исходных, где представлены пары предложений, в этот раз разделенные специальным «SEP» токеном. Если одно предложение следует после другого то целевая переменная равняется единице, в остальных случаях нулю. Наглядно процесс предобучения показан на рисунке 2.2.

В данной работе мы не будем подробно останавливаться на описании архитектуры модели, однако стоить заметить что в работе описано, что (Devlin, et al.,2018) модель в том варианте, что мы ее используем «base» представляет собой 12 последовательных encoder слоев из работы исследователей OpenAI в которой впервые была представлена архитектура трансформер, состоящая из одного энкодера и одного декодера (Vaswani et. all, 2017). В конченом итоге мы получаем 110 миллионов параметров для оптимизации, так как модель полностью дифференцируема, возможно нахождение векторов градиентов функции с помощью метода обратного распорстранения ошибки, соответсенно возможно обучение, тоесть минимизация функции потерь

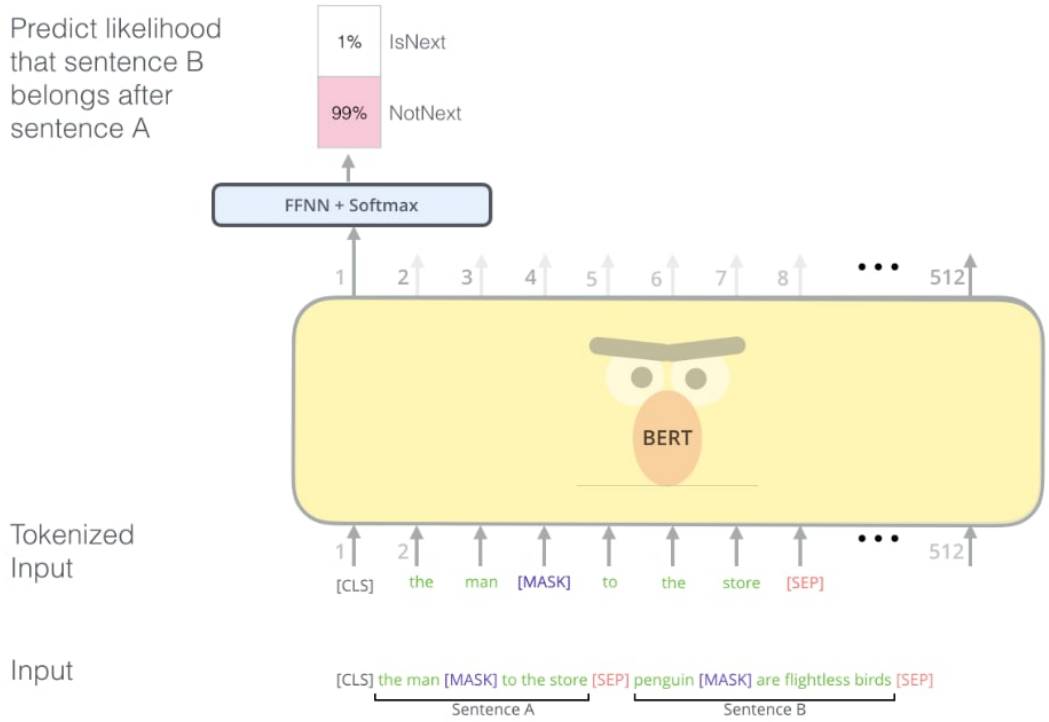

Рис. 2.2. Источник: The illustrated BERT (Alammar, 2018)

с помощью градиентного спуска. Исследователи из Google для данной задачи предлагают использование модификации стохастического градиентного спуска - Adam, в качестве функции потерь кросс-энтропию.

Метод обратного распространения ошибки

Обратное распространение ошибок, представляет собой алгоритм обучения с учителем нейронных сетей с использованием градиентного спуска. Учитывая искусственную нейронную сеть и функцию ошибок, метод вычисляет градиент функции ошибок по отношению к весам нейронной сети. Это обобщение дельта-правила для персептронов на многослойные нейронные сети с прямой связью.

ADAM

ADAM, представленный впервые в статье (Kingma, 2014) алгоритм градиентной оптимизации первого порядка стохастических целевых функций, основанный на адаптивных оценках моментов более низкого порядка. Этот метод прост в реализации, эффективен в вычислительном отношении, требует мало памяти, инвариантен к диагональному изменению масштаба градиентов и хорошо подходит для

решения задач с большими объемами данных и/или параметров. Этот метод также подходит для нестационарных целей и задач с очень шумными и/или редкими градиентами. Гиперпараметры имеют интуитивную интерпретацию и обычно требуют небольшой настройки.

ADAM дополнительно к стохастическому градиентному спуску оценивает два параметра m и ν

$m_t = \frac{\beta_1 m_{t-1}+(1-\beta_1)g_t}{a-b_1^t}; \nu_t = \frac{\beta_2 \nu_{t-1}+(1-\beta_2)g_t^2}{1-\beta_2^t}$, где $m, \nu-$ скользящие средние, $g-$ градиент текущей мини-выборки, β_1, β_2- гиперпараметры.

Кросс-энтропия

Кросс-энтропия - функция потерь которая чаще всего используется в задачах классификации в нейронных сетях, принимает большое значение когда класс предсказан неправильно и маленькое когда класс предсказан правильно. В процессе обучения нейронной сети мы минимизируем данную функцию потерь с помощью градиентного спуска или же какого-то из его вариаций таких как ADAM. Формула кросс энтропии для многоклассовой классификации:

$$-\sum_{c=1}^{N} y_c \log(p_c) \tag{2.5}$$

Где y_c - метка принадлежности к классу
p_c- вероятность класса у наблюдения предсказанная моделью

2.4.3. Тонкая настройка модели (fine tuning)

Точная настройка проста, так как механизм self-attention в Transformer позволяет BERT моделировать многие последующие задачи — независимо от того, включают ли они один текст или текстовые пары — путем замены соответствующих входов и выходов используя относительно малое количество данных.

Описанный нами второй этап тонкой настройки модели происходит по походему на первый этап сценарию. Мы берем все наши собранные данные из социальных сетей (примерно 400000 постов), кодируем как было описано выше и проводим обучение на задачу предсказания

замаскированных слов. На задачу предсказания следующего предложения модель не дообучаем так как нам для классификации постов по тональности настроения данная задача не принесет пользы, а только займет время и ресурсы.

Обучение производится на протяжении 10 эпох, как советуется в работе (Devlin, et al.,2018). Некоторые исследователи советуют заморозку части слоев во время данного этапа, например работа (Vucetic et. all, 2022), однако мы будем придерживаться оригинального способа построения модели предложенного исследователями из Google.

Для третьего этапа в модель полученную на втором этапе добавляется линейный слой, который представляет собой простую двухслойную полносвязную сеть, с функциями активации ReLU и Softmax, с количеством выходов равному 3, каждому выходу в соответствие ставится класс. На вход модель получает CLS токен, который помещается в начале каждого предложения (как было описано в главе с описанием предобучения модели) на выходе модель выдает вероятности принадлежности каждого поста к классу: позитивному, нейтральному или негативному. В соотвествие к классам поставлены числа: 0 для негативного класса, 1 для нейтрального, 2 для положительного класса.

Также так как разметка данных производилась вручную, использовался метод активного обучения, описанный в статье (Caramalau,2021). Суть метода состоит в том, что сначала размечается небольшая часть выборки, модель обучается на ней, с помощью модели мы создаем предсказания для всех собранных постов в выборке, находим те на которых модель наименее уверенна в классе (максимальная вероятность класса минимальна), размечаем их, и так продолжаем пока количество неуверенных ответов модели не станет приемлемо низким.

Глава 3. Данные

В данной главе описан процесс сбора данных, описаны получившиеся выборки, обозрены преобразования данных выполненные автором

3.1. Данные с фондовых рынков

Для нашей работы сбор данных проводился в два этапа, сначала были собраны данные о ценах акций выбранных компаний за период с начала 2020 года, до начала апреля 2022. Длина итоговой выборки получилась чуть больше двух лет. Для получения данных о ценах на акции используется бесплатная библиотека GitHub, которая достаточно проста в использовании. Код использования данной библиотеки будет приложен к работе. Библиотека возвращает цены акций («открытие», «закрытие», «максимум» и «минимум»), но в этой статье основное внимание уделяется прогнозам цены «закрытия», тоесть последней цены за день. Затем, поскольку известно, что цены акций обычно представляют собой нестационарные временные ряды, а стационарность нам необходима для построения линейных моделей и проведения тестов Грейнджера, данные преобразуются в ежедневные доходности с помощью следующего соотношения:

$$\text{return}_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{3.1}$$

Где P_t - цена закрытия текущего дня
P_{t-1} - цена закрытия предыдущего дня
$return_t$ - доходность

В последствие, после сбора текстовых данных удалось для компаний Газпром, ВТБ и TCS Group собрать данные с начала сентября 2019 года, поэтому для этих компаний были дополнительно собраны данные о доходностях.

На рисунке можно увидеть графики доходностей компаний за рассматриваемый период, визуально графики выглядят стационарными, однако можно увидеть выбросы в конце марта, начале апреля 2022

года. Соответсвенно были проведены Augmented Dickey-Fuller тесты на стационарность для каждого временного ряда (Dickey, Fuller, 1978).

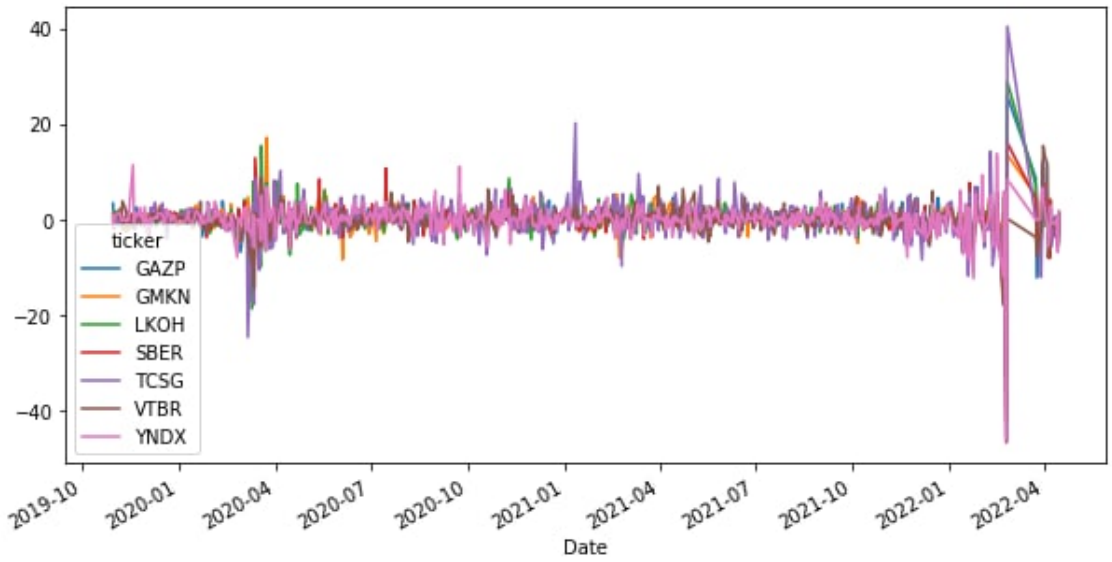

Рис. 3.1. Источник: Расчитано автором

Мы выбираем AIC в качестве критерия для выбора подходящего количества лагов в процессе AR (p). Результаты тестов мы можем видеть в таблице, как мы видим для всех временных рядов нулевая гипотеза о нестационарности отвергается на уровнях значимости в 1% и 5%

ticker	test statistic	p-value
GAZP	-6.433	0.0
GMKN	-16.1416	0.0
LKOH	-20.2849	0.0
SBER	-6.0892	0.0
TCS	-20.0126	0.0
VTBR	-10.1203	0.0
YNDX	-10.8908	0.0

Таблица 3.1. Расчитано автором

3.2. Данные из социальных сетей

Данные из «Тинькофф Пульс» были собраны с помощью запросов к Rest API, который использует данная социальная сеть. В открытом доступе простого и удоного средства для парсинга данных с данной социальной сети автору найти не удалось.

Изначально процесс сбора данных происходил с помощью библиотеки Selenium, однако для сбора большого количества данных от данного подхода пришлось отказаться в силу медленной скорости парсинга. Был найден пакет для языка Python под названием tpulse, однако в нем не было возможности парсинга более чем 30 последних сообщений для каждого тикера. Автором данной работы данный пакет был доработан для парсинга какого угодно количества сообщений по каждому тикеру, код для этого будет указан в приложении.

Сообщения в каналах Телеграм были собраны с помощью оффициального API Телеграм, сообщения собирались только из выбранных автором каналов на инвестиционную тематику, например «Твердые Цифры ВТБ», «Invest Heroes», «СберИнвестиции», в выборку попали только сообщения содержащие название рассматриваемой компании или тикер компании. В связи с этим к 370000 сообщений из «Тинькофф Пульс» удалось добавить дополнительно только 2000 сообщений.

Далее в процессе предобработки из выборки были удалены все сообщения относящиеся более чем к одной компании из списка, так как данные сообщения могут содержать различную тональность настроения инвестора по отношению к различным акциям. Например «ВТБ покупать, Газпром будет падать», из-за чего нельзя сделать однозначного вывода об тональности инвестора в данном посте.

Также в процессе предобработки из постов были удалены не несущие информации о тональности конструкции, такие как ссылки и рекламный хештег в «Тинькофф Пульс» «#миллионыдомастиньтинькофф», пустые сообщения после предобработки были также удалены.

Так как в предобученных моделях BERT присутствуют токены для эмоджи, а они явно несут информацию о тональности сообщения,

мы оставляем их в тексте. Например эмоджи изоброжающий ракету зачастую ставится при положительной тональности поста.

Таким образом в итоговой выборке осталось около 250000 сообщений.

Далее автором было вручную размечено 3000 сообщений, был проставлен класс объекта: 0 если в сообщение негативное отношение к акции, 1 если нейтральное, 2 если положительное. Описанный в главе с обзором модели BERT метод активного обучения применялся для выбора данных для разметки.

Предобработка данных для языковой модели с разбиением на токены, добавлением специальных токенов, макскировкой слов, описанная в главе про BERT была выполняется автоматически в Python библиотеке transformers, которая использовалось для обучения языковой модели

Глава 4. Создание признака тональности

В данной главе будет описан выбор модели для классификации текстов, описаны результаты использования модели классификации с точки зрения метрики точности (accuracy), описан подход к построению индекса тональности

4.1. Выбор модели для классификации тональности

Так как тексты для обучающей выборки размечались автором, в обучающую выборку было включено примерно равное количество текстов каждого класса, при приобладании какого-либо из классов, автор при ручной разметке пропускал элементы этого класса и размечал данные далее. Соответсвенно для выбора лучшей модели мы будем использовать метрику точности, так как в таком случае она хорошо отражает качество модели. Распределение классов можно увидеть на рисунке 4.1.

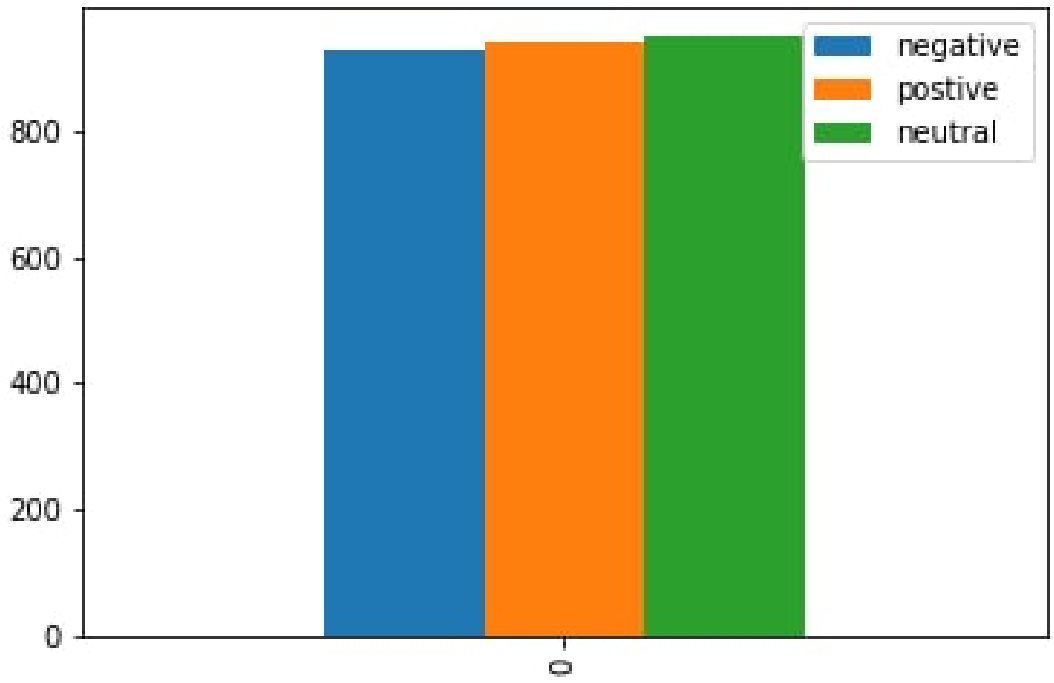

Рис. 4.1. Источник: Расчитано автором

Accuracy на тестовой выборке считается как доля тех наблюдений для которых модель предсказывает верный класс. В ходе исследования были рассмотрены следующие модели: модель описанная ruBERT-conversational описанная в предыдущих главах, модель ruBERT-converstional предобученная на большом корпусе текстов на задачу определения тональности без дальнейшей тонкой настройки под исследуюмую задачу, предыдущая модель, но дообученная только на задачу определения тональности на тренировочной выборке, без тонкой настройки на задачу MLM на всей выборке собранной автором, а также модель логистической регресии обученная на TF-IDF представлении тренировочной выборки.

На задачу MLM модель тонко настраивалась в течении 10 эпох, на задачу предсказания тональности в течении 20 эпох. Результаты моделей можно увидеть в таблице 4.1. Оценка Accuracy проводилась на отложенной выборке размером примерно в 500 наблюдений.

Model	Accuracy
ruBert-conversational-finetuned-both	69,7 %
ruBert-conversational-sentiment	52,4 %
ruBert-conversational-sent-finetuned	62,3 %
TF-IDF + logreg	47,1 %

Таблица 4.1.Источник: расчитано автором

Как мы можем видеть, лучшие результаты были показаны моделью, построение которой было описано в главе 2. Можно сделать вывод что в нашей задаче тонкая настройка на задачу MLM на всем наборе данных дает значимый прирост точности модели, также описанные результаты позволяют доказать гипотезу о том, что использование моделей глубинного обучения может улучшить качество классификации тональности сообщений. Код для обучения и дальнейшего использования модели для предсказаний будет размещен в приложении.

4.2. Построение признака тональности

Далее с помощью оптимальной модели мы делаем предсказания и выбираем для сообщения тот класс тональности, вероятность которого с точки зреня модели максимальна. Получившаяся модель уверенно классифицирует всю выборку, из примерно 250 тысяч наблюдений, количество наблюдений у которых мера неопределенности модели $(1-max(p_i)$, где p_i - вероятности принадлежности наблюдения к одному из трех классов) больше 0.3 не превышает четырех тысяч. Пример результатов оценки модели, взяты 3 случайных наблюдения, последнее сообщение сокращено для лучшего вида в таблице, так как оно было длиной в 200 слов:

Сообщение	negative	neutral	positive	label
{$YNDX} сливайте пока не поздно	0.972525	0.005213	0.022262	-1
{$LKOH} Кто тут сколько зарабатывает в среднем в месяц за ежедневный просмотр графика?)	0.001817	0.995833	0.002350	0
На следующей неделе буду докупать компанию {$LKOH}...	0.002930	0.312310	0.684760	1

Таблица 4.2.Источник: расчитано автором

Далее полученные данные для каждой компании аггрегируются за день, чтобы иметь такую же переодичность как рассматриваемые в исследование финансовые данные по следующей формуле:

$$\text{Sentiment Score } e_{t,s} = \frac{\text{Pos}_{t,s} - Neg_{t,s}}{\text{Pos }_{t,s} + Neu_{t,s} + Neg_{t,s}} \tag{4.1}$$

где $e_{t,s}$ - Тональность отношения к акции за определенное время;
$Pos_{t,s}$ - Количество позитивных постов за это время;
$Neg_{t,s}$ - Количество отрицательных постов за это время;
$Neu_{t,s}$ - Количество нейтральных постов за это время;

Как было описано выше: В нашей работе мы используем подход агрегирования, который включает в себя разницу между количеством постов с положительной и отрицательной тональностью и масштабирование ее по общему количеству постов в этот день, включая нейтральные посты, как это сделано в работе (Hiew, et al., 2019).

Также так как в дальнейшем мы будем использовать наши индексы тональности в моделях свзяанных с временными рядами проверям стационарность каждого ряда с помощью теста Augmented Dickey-Fuller.

ticker	test statistic	p-value
GAZP	-8.6365	0.0
GMKN	-27.9731	0.0
LKOH	-9.772	0.0
SBER	-11.9203	0.0
TCS	-4.6917	0.0
VTBR	-11.7069	0.0
YNDX	-4.5661	0.0

Таблица 4.3. Расчитано автором

Как мы можем видеть из результатов тестов, для всех рассматриваемых временных рядов отвергается гипотеза о нестационарности

Глава 5. Результаты

В данной главе будет рассказано про проведение тестов Грейнджера, построение линейной модели включающей признаки тональности и построение модели без признака тональности, использование данной модели для проведения статистических тестов на значимость индекса тональности настроения инвесторов.

5.1. Тесты причинности по Грейнджеру

Поскольку мы имеем дело с временными рядами, тест причинности по Грейнджеру кажется подходящим, чтобы увидеть, полезен ли один ряд для объяснения поведения другого (при условии линейной зависимости). Но важно иметь в виду, что на самом деле причинность по Грейнджеру не подразумевает действительной причинности. (Hamilton, 1994) утверждает: «Вместо того, чтобы проверять, является ли Y причиной X, причинность Грейнджера проверяет, предсказывает ли Y X».

Дизайн теста следующий:

Пусть X, Y - стационарные временные ряды.

$H_0 : x$ не является причинным по Грейджеру для y

$H_1 :$ Не H_0

1) Находится оптимальное количество лагов и строится авторегрессия следующего дизайна:

$$y_t = a_0 + a_1 y_{t-1} + \cdots + a_m y_{t-m} + e_t \tag{5.1}$$

2) Добавляются лаги x в авторегрессию с первого шага:

$$y_t = a_0 + a_1 y_{t-1} + \cdots + a_m y_{t-m} + b_p x_{t-p} + \cdots + b_q x_{t-q} + \mathrm{e}_t \tag{5.2}$$

Остаются только те x_i, которые значимы по отношению к t-статистике и при условии, что они совместно значимы по F-критерию. H_0 отклоняется, если ни один x_i не выдерживает проверки значимости.

Однако текущие реализации тестов как в R, так и в Python позволяют включать в неограниченную модель только одинаковое коли-

чество лагов зависимых и независимых переменных. Автор использует библиотеку Statmodels в Python с ее встроенными функциями (который также требует меньших вычислительных затрат, чем обобщение теста GC, поскольку последний имеет дело с гораздо большим числом оцениваемых моделей, чей информационный критерий Akaike (AIC) (Akaike, 1974) или байесовский информационный критерий (BIC) (Schwarz, 1978) следует сравнить позже, чтобы выбрать наилучшую модель для теста GC). Тем не менее, можно рассмотреть возможность модернизации существующего теста GC как способ улучшения дальнейших исследований.

Видно, что условием правильности теста GC является стационарность ряда. Действительно, в противном случае риск получения ложной регрессии неприемлемо высок. В предыдущих главах используем расширенный тест Дикки-Фуллера (ADF) (Dickey, Fuller, 1979) для проверки нестационарности типа случайного блуждания. Как мы видим из вышеописанных результатов наши временные ряды являются стационарными

Первоначальная гипотеза слегка трансформируется нами для проверки того, что изменение настроений людей (а не само настроение) добавляет прогностическую силу прогнозированию доходности. Наконец, для каждого временного ряда доходностей применяется GC-тест для лагов в диапазоне от 1 до 10 включительно. Затем BIC сравнивается между неограниченными моделями, и выбирается модель с наименьшим BIC, которая используется при оценке p-значения теста GC.

Далее на рисунке 5.1 приведены средние p-значения теста причинности по Грейнджеру для первых пяти лагов, с гипотезой, что тональность настроения частных инвесторов из социальных сетей добавляет прогнозистическую силу прогнозированию доходности. Стоит заметить, что для всех компаний кроме Яндекса наша гипотеза подтверждается для всех значений лага на уровне значимости в 5 %, для Яндекса гипотеза подтверждается только для 2 лага индекса тональности.

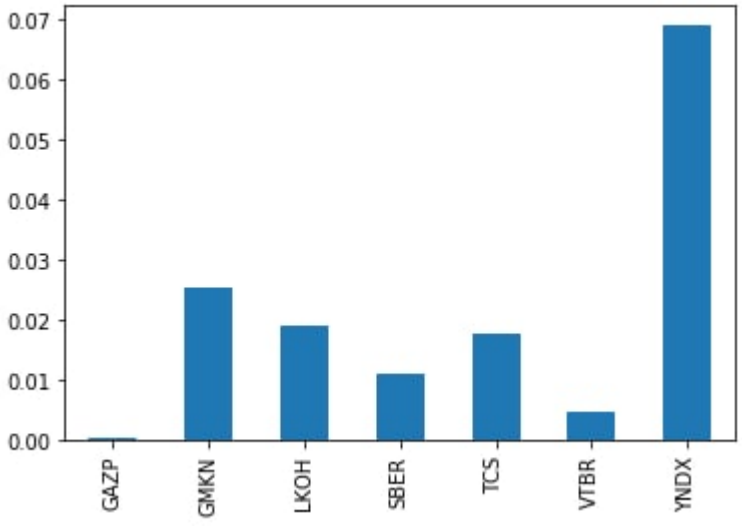

Рис. 5.1. Источник: Расчитано автором

Стоит также заметить, что обратная гипотеза, что доходность позволяет прогнозировать индекс тональности не подтверждается ни для одной компании, как мы можем наблюдать на рисунке 5.2 поэтому для дальнейшего анализа будет некорректно использовать модель векторной авторегресии.

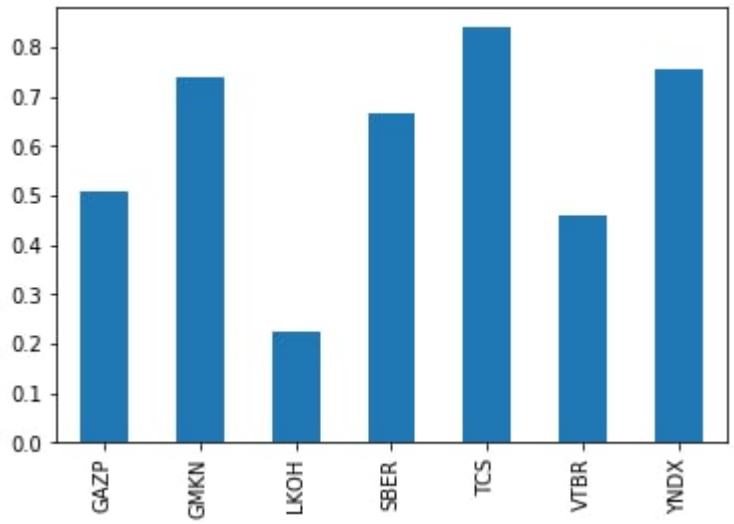

Рис. 5.2. Источник: Расчитано автором

Также с помощью критерия BIC были выбраны оптимальные величины лагов для модели используемой в тесте для каждой компании, и посчитано p-значение для теста Грейнджера для этого количества лагов. Результаты можно увидеть в таблице. Также можно заметить, что согласно критерию BIC выбираются низкие значения оптимально-

го лага, в то время как AIC выбирал максимальные значения лагов. Основное различие между AIC и BIC заключается в выборе модели. Они предназначены для конкретных целей и могут давать отличные результаты. AIC имеет бесконечные и относительно большие размеры. AIC приводит к сложным признакам, тогда как BIC имеет более конечные размеры и согласованные атрибуты. Мы выбираем критерий BIC, так как в последствие мы будем строить модель для прогнозирования с выбранным количеством лагов, и хотим избежать переобучения.

	Lag 1	Lag 2	Lag 3	Lag 4	Lag 5	Optimal Lag
GAZP	0.0001	0.0001	0.0003	0.0003	0.001	2
GMKN	0.1243	0.0002	0.0004	0.001	0.0014	3
LKOH	0.0031	0.0057	0.0149	0.0275	0.0447	1
SBER	0.0007	0.0034	0.0047	0.0151	0.0325	1
TCS	0.0115	0.0339	0.0073	0.0158	0.0198	1
VTBR	0.0002	0.0012	0.0023	0.0064	0.0139	2
YNDX	0.015	0.0338	0.0715	0.0919	0.133	2

Таблица 5.1. Расчитано автором

Как мы можем заметить из таблицы, гипотеза о том, что тональность настроения частных инвесторов из социальных сетей добавляет прогнозистическую силу прогнозированию доходности все менее вероятна с повышением числа лага для всех компаний, что согласовывается с представлениями о том, что более старое мнение из социальных сетей меньше влияет на доходность акций компании.

5.2. Создание линейной модели для прогнозирования доходностей

Так как тест Грейнджера показал, что признаки тональности полезны для прогнозирования доходностей, следующим разумным шагом будет построение модели, осуществляющей прогнозирование. Будут построены модели с лагами индекса тональности и без, а также

проведены тесты на значимость индекса тональности для каждой компании.Подобный подход необходим для того чтобы избежать ложных результатов так как возможно что признаки тональности уже содержат в себе часть информации из других признаков, а нам в свою очередь требуется влияние именно самих признаков тональности. Линейная модель выбирается в силу простоты интерпретации, так как нам требуется оценить значимость тональности частных инвесторов для прогнозирования дохдностей компаний, а не получить лучшие прогнозы.

5.2.1. Подготовка данных

На этапе подготовки данных, для каждой компании создается набор признаков, в который помимо признаков тональности включается набор признаков часто используемый в задаче предсказания доходности, так мы включаем лаги доходности, выбирая количество лагов используя критерий BIC описанный в предыдущих главах, также мы включаем признаки дня недели, месяца и года наблюдения, широко известен факт, что такие признаки могут влиять на доходность (Сиделев, 2018).

В силу того, что хотя дни недели и прочие признаки такого типа представляют собой числа, на самом деле они являются категориальными признаками, поэтому для каждого значения таких признаков создается бинарный признак принимающий значение 1, когда у наблюдения категориальный признак равен этому числу, и 0 иначе. Также во всех случаях первый бинарный признак удался, так как информацию о том что признак принимает данное значение можно получить через остальные бинарные признаки.

Так как согласно критерию BIC число лагов доходности для всех компаний $N_L = 2$ итоговое количество регрессоров в моделях получислось равным

$$N_R = (N_{months} - 1) + (N_{years} - 1) + (N_{workdays} - 1) + 2 + N_{L_{sent}} = 20 + N_{L_{sent}} \tag{5.3}$$

где N_i обозначает колчество признаков данного типа. Такое большое число признаков может привести к мультиколлинеарности, поэтому для каждой компании была изучена матрица корреляций признаков, признаков с корреляцией > 0.14 не обнаружено

5.2.2. Модель

Для каждой компании мы рассматриваем два класса линейных моделей: линейную регрессию с признаками тональности (лагами индекса тональности выбранными в таблице 5.1) и без них.

$$\mathrm{y}_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 xsentiment_{t-1} ... + \beta_{3+i} xsentiment_{t-i} + \beta_{i+4} x_{other} + ... + error_t \quad (5.4)$$

$$\mathrm{y}_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 x_{other} + ... + error_t \quad (5.5)$$

Для того, чтобы убедится, что оценки коэффициентов их дисперсии были несмещенными автором проводится тест Ljung-Box (G. M. Ljung, 1978) выполняется для каждой из моделей, чтобы подтвердить отсутствие автокорреляции в остатках. Также чтобы убедиться в отсутствии мультиколлинеарности для каждого из регрессоров строилась статистика VIF, VIF превышающего 10 не было обнаружено ни в одном из экспериментов. Также были посчитаны статистики Skrewness и Kurtosis, для остатков моделей, все выбранные модели не обладают нормальностью остатков, соответственно F-статистика будет считаться с помощью модификации.

Так как наша выборка за два года и 3 месяца, и включает в себя только рабочие дни, она включает в себя 597 наблюдений, что является довольно небольшим числом. Для решения проблем связанных с малым количеством данных (недообучение модели, различные смещения оценок коэффициентов, из-за малой отложенной выборки смещение оценок ошибок модели) мы используем кросс валидацию временных рядов, предложенную в уже обозренной ранее работе (Mittall, 2011). Графически данный подход показан на Рисунке 5.3. Алгоритм получения прогнозов следующий:

1. Модель строится на первых N наблюдениях
2. Предсказывают следующие Q наблюдений

3. Оцениваются коэффициенты модели, интересующие нас метрики
4. Добавьте наблюдения Q в набор поездов, начиная с Q-го, чтобы модель строилась на наблюдениях $[Q, N + Q]$
5. Шаги 1-4 повторяются пока не модель не обучится на всей выборке

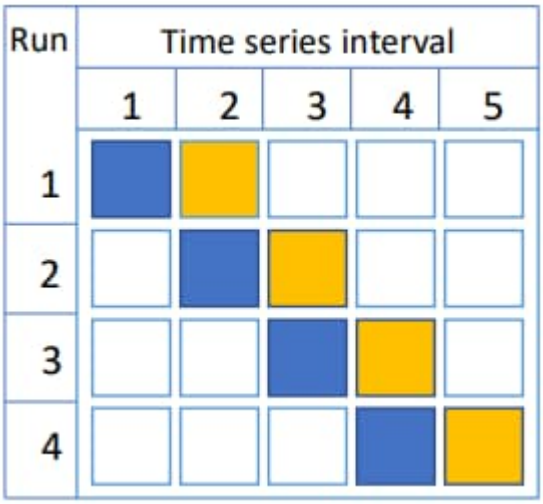

Рис. 5.3. Источник: Расчитано автором

Синие прямоугольники иллюстрируют наборы для обучения, а желтые — для тестирования. На картинке 4 итерации процесса. Эта процедура известна как k-SCV с небольшой модификацией: каждый раз, когда добавляются новые наблюдения, первые удаляются, что делается для большей гибкости модели.

Итоговые оценки коэффициентов и интересующих нас метрик усредняются из k построенных моделей. Данный алгоритм реализован в программном пакете для языка Python scikit-learn.

Метрики ошибки прогноза модели используемые в работе - MSE и MAPE. Данные метрики чаще всего используются в работах которые были обозрены в списке литературы. MSE также является стандартной функцией ошибки модели линейной регрессии, а MAPE обладает полезным свойством - понимание на сколько мы ошибаемся в процентах

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{5.6}$$

Где $\hat{Y}_i$ - предсказания модели
Y_i - Истинные значения

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{\left|Y_i - \hat{Y}_i\right|}{Y_i} \tag{5.7}$$

5.2.3. Статистические тесты для проверки значимости тональности

t-тест

В нашей работе для проверки гипотезы о значимости тональности частных инвесторов для прогнозирования доходностей акций российских компаний будет применено два типа тестов. Первый тест - проверка значимости коэффициента перед лагами индекса тональности. Тестовая статистика для данного теста имеет распределение Стьюдента со степенями свободы равными количеству наблюдений минус количество коэффициентов. Нулевая гипотеза теста состоит в том, что $\beta = 0$, альтернативная - $\beta \neq 0$. Статистика считается следующим образом.

$$t = \frac{\beta - 0}{SE_\beta} \tag{5.8}$$

Где β - рассматриваемый коэффициент SE_β - стандартное отклонение коэффициента Вероятность принятия нулевой гипотезы:

$$P(t) = 2\min\{G(t); 1 - G(t)\} \tag{5.9}$$

Где $G(t)$ - функция распределения статистики

F-тест

Также в работе используется F-тест для проверки ограничений на параметры регресии. Проверяем является ли незначимым исключение индекса тональности из регрессии. Так как было описано выше, в нашей работе не выполняется условие нормальности остатков, необходимо использовать одну из асимптотических модификаций теста. Одной из таких модификаций является F-тест с использованием статистики Вальда. Статистика Вальда для классической линейной регрессии выглядит следующим образом.

$$W = \frac{MSE_S - MSE_L}{MSE_L/n} \tag{5.10}$$

Где MSE_S - в нашем случае MSE оцененый методом кросс-валидации «короткой модели» (без лагов сентимента)
А MSE_L - MSE «длинной модели» соответственно наоборот включающей в себя лаги сентимента

Итоговая статистика имеет распределение Фишера и считается по следующей формуле:

$$F = \frac{n - k_{ur}}{r} W/n \tag{5.11}$$

Где n- число наблюдений, r - число ограничений, k_{ur} - число регрессоров в полной модели, включая свободный член

5.3. Результаты тестов

Для каждой компании из выборки были построены две вышеописанные модели, оценены все необходимые параметры с помощью k-SCV кросс-валидации. В результате у всех моделей был хотя-бы один значимый коэффициент лага индекса тональности. Однако для большинства моделей часть лагов, не смотря на результаты теста причинности по Грейнджеру оказались незначимы. Также те лаги что оказались незначимыми обычно являются более поздними, то есть отражающими значение индекса тональности инвесторов несколько дней назад.

Данные результаты согласуются с высказанным предположением, что чем более дальний лаг тем меньше должно быть его влияние на сегодняшние доходности. Более того, то что часть лагов выбранных в ходе GC теста оказались незначимыми подтверждает необходимость включения в модель дополнительных регрессоров, для получения более точных оценок влияния индекса отношения инвесторов. А также данные результаты подтверждают гипотезу о значимости индекса тональности частных инвесторов для предсказания доходностей по акциям рассматриваемых компаний. Результаты t-тестов для всех рассматриваемых регрессий можно найти в таблице 5.2. Также можно увидеть что все значемые коэффициенты положительные, что тоже согласуется с гипотезой о том, что частные инвесторы могут значимо влиять на рынок акций.

Тикер	Коэффициент	t-стат.	p-value
GAZP	sent_lag_1**	3.255	0.001
	sen_lag_2*	1.990	0.047
GMKN	sent_lag_1	1.485	0.138
	sent_lag_2**	3.962	0.000
	sent_lag_3	-1.349	0.178
LKOH	sent_lag_1**	3.468	0.001
SBER	sent_lag_1**	2.988	0.003
TCS	sent_lag_1*	2.498	0.013
VTBR	sent_lag_1**	3.522	0.000
	sent_lag_2	-0.342	0.732
YNDX	sent_lag_1**	2.266	0.024
	sent_lag_2	0.539	0.590

Таблица 5.2. t-тесты. Расчитано автором. * - значимость на уровне 5%, ** - значимость на уровне 1%

Переходим к результатам F-тестов. Как мы можем видеть, что гипотеза о незначимости исключения лагов индекса тональности отвергается для всех рассматриваемых компаний, кроме Яндекса. Данная ситуация может интерпретироваться как то, что модель классификации тональности могла плохо предсказать тональность сообщений частных инвесторов о Яндексе, так как на всех этапах самые плохие результаты значимости были у индекса тональности Яндекса. Однако такое объяснение выглядит не очевидным, исследователь предполагает, что такие результаты могут быть связаны с тем, что акции данной компании могут быть больше зависимыми от каких-либо фундаментальных показателей, а также пользоваться большим институциональным спросом. Результаты F-тестов доступны в таблице 5.3.

ticker	p_value	stat value
GAZP	0.0	11.1
GMKN	0.0	54.1
LKOH	0.0	284.0
SBER	0.0	46.2
TCS	0.0	192.5
VTBR	0.0	69.2
YNDX	1.0	-7.9

Таблица 5.3. F-тесты. Расчитано автором

Заключение

В нашей работе мы провели исследования влияния частных инвесторов на рынок акций в России. Было выбрано 7 самых популярных компаний среди физических лиц по данным Мосбиржи. В качестве оценки тональности отношения инвесторов к акциям были выбраны их сообщения в социальных сетях.

Далее были проверены несколько гипотез: классификация тональности сообщений в социальных сетях может быть улучшена с помощью использования методов глубинного обучения. Данная гипотеза оказалась верна, действительно полученный автором классификатор на основе архитектуры BERT превосходит стандартные подходы с точки зрения accuracy.

Также в силу отсутствия предтренированных на российских финансовых сообщениях архетектур глубинного обучения в открытом доступе, полученная автором модель представляет ценность для дальнейших исследований. В качестве развития работы возможно будет при наличие бюджета разметить большее количество данных и обучить модель на них. Это позволит улучшить качество модели, а соответственно и индекса тональности.

Более того, в работе была проверена гипотеза о значимости влияния индекса тональности на доходности рассмотренных компаний, для 6 из 7 рассмотренных компаний влияние значимо. Для всех компаний тест Грейнджера показывает причинность для доходностей хотя бы одного лага индекса тональности. В качестве дальнейшего развития работы возможно в будующих исследованиях применить нелинейные преобразования к прогнозам модели классификации, так как возможно доходность акций зависит нелинейно от индекса тональности сообщений частных инвесторов в социальных сетях

Также фокусом данной работы была значимость индекса тональности, поэтому в качестве модели для прогнозов применялась линейная модель, однако в будующих работах возможно создание торговой стратегии на основе данного индекса и его нелинейных преобразова-

ний, и следовательно использование более сложных моделей машинного обучения для данной задачи.

Библиографический список

1. Akaike H. A new look at the statistical model identification // IEEE transactions on automatic control. 1974. № 6 (19). С. 716–723.
2. Antweiler W., Frank M. Z. Is all that talk just noise? The information content of internet stock message boards // The journal of finance. 2004. № 3 (59). С. 1259–1294.
3. Beck P. van der Flow-driven ESG returns // SSRN Electronic Journal. 2021.
4. Berkson J. Application of the logistic function to bio-assay // Journal of the American Statistical Association. 1944. № 227 (39). С. 357–365.
5. Caramalau R., Bhattarai B., Kim T.-K. Visual Transformer for task-aware Active Learning // arXiv [cs.CV]. 2021.
6. Cortes C., Vapnik V. Support-vector networks // Machine learning. 1995. № 3 (20). С. 273–297.
7. Devlin J. et al. BERT: Pre-training of deep bidirectional Transformers for language understanding // arXiv [cs.CL]. 2018.
8. Dickey D. A., Fuller W. A. Distribution of the estimators for autoregressive time series with a unit root // Journal of the American Statistical Association. 1979. № 366 (74). С. 427.
9. Gwet K. L. Intrarater Reliability Hoboken, NJ, USA: John Wiley & Sons, Inc., 2014.С. 340–356.
10. Hiew J. Z. G. et al. BERT-based financial sentiment index and LSTM-based stock return predictability // arXiv [q-fin.ST]. 2019.
11. Kearney C., Liu S. Textual sentiment in finance: A survey of methods and models // International review of financial analysis. 2014. (33). С. 171–185.
12. Kingma D. P., Ba J. Adam: A method for stochastic optimization // arXiv [cs.LG]. 2014.

13. Li X., Wu P. Stock price prediction incorporating market style clustering // Cognitive computation. 2022. № 1 (14). С. 149–166.
14. Ljung G. M., Box G. E. P. On a measure of lack of fit in time series models // Biometrika. 1978. № 2 (65). С. 297–303.
15. Mingzheng L. et al. A Chinese stock reviews sentiment analysis based on BERT model // Research Square. 2020.
16. Mittal A., Goel A. Stock Prediction Using Twitter Sentiment Analysis // Stanford.edu [Электронный ресурс]. URL: http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf (accessed: 12.05.2022).
17. Pagolu V. S. et al. Sentiment analysis of Twitter data for predicting stock market movements IEEE, 2016.
18. Pota M. et al. An effective BERT-based pipeline for Twitter sentiment analysis: A case study in Italian // Sensors (Basel, Switzerland). 2020. № 1 (21). С. 133.
19. Semenova V., Winkler J. Social contagion and asset prices: Reddit's self-organised bull runs 2021.
20. Sonkiya P., Bajpai V., Bansal A. Stock price prediction using BERT and GAN // arXiv [q-fin.ST]. 2021.
21. Vaswani A. et al. Attention is all you need // arXiv [cs.CL]. 2017.
22. Xue J. et al. Public discourse and sentiment during the COVID 19 pandemic: Using Latent Dirichlet Allocation for topic modeling on Twitter // PloS one. 2020. № 9 (15). С. e0239441.
23. А.В.Дубко, К.Буассье. — 2015.
24. Андрианова Е. Г., Новикова О. А //Cloud of Science. — 2018. — т. 5
23. Частные инвесторы вложили в 2021 году в российские ценные бумаги на Московской бирже 1,35 трлн рублей // Московская Биржа

[Электронный ресурс]. URL: https://www.moex.com/n39552/?nt=106 (accessed: 12.05.2022).
24. Статистика объемов торгов Московская Биржа // Московская Биржа [Электронный ресурс]. URL: https://www.moex.com/ru/ir/interactive-analysis.aspx (accessed: 12.05.2022).
25. Тинькофф — финансовые услуги для физических и юридических лиц // Тинькофф Банк [Электронный ресурс]. URL: https://www.tinkoff.ru/about/news/03092021-social-network-pulse-2-years-1.4-mln-users-3-mm-posts-14-mln-comments-19-mln-likes/ (accessed: 12.05.2022).
26. // Rbc.ru [Электронный ресурс]. URL: https://quote.rbc.ru/news/article/621396ea9a79472c01aba5ed (accessed: 12.05.2022).
27. Московская Биржа - Инфографика // Московская Биржа [Электронный ресурс]. URL: https://www.moex.com/s2184 (accessed: 12.05.2022).
28. // Interfax.ru [Электронный ресурс]. URL: https://www.interfax.ru/business/824299 (accessed: 12.05.2022).
29. Papers with code - sentiment analysis // Paperswithcode.com [Электронный ресурс]. URL: https://paperswithcode.com/task/sentiment-analysis (accessed: 12.05.2022).

Приложение 1. Парсинг «Тинькофф Пульс»

```
!pip install requests && pip install pytz
!pip install pyopenssl

import requests
import sqlite3

from time import sleep
from datetime import datetime
from pytz import timezone
import pandas as pd

s = requests.session()
s.keep_alive =False

ls = []
def insert(data, ls):
    ls.append(data)
    return ls

def get_data_from_api(url: str, ticker: str, cursor_number:
 ↪  str):
    link = url.format(ticker, cursor_number)
    session = requests.Session()
    data = session.get(link, headers=headers, stream=True)
    raw_data = data.json().get('payload').get('items')

```

```
    for i in tqdm(raw_data):
        my_data = {
            'ticker': ticker,
            'id': i.get('id')[:7],
            'text': i.get('text'),
            'instruments': ',
             ↪  '.join(get_instruments(i.get('instruments'))),
            'likesCount': i.get('likesCount'),
            'nickname': i.get('nickname'),
            'commentsCount': i.get('commentsCount'),
            'date': i.get('inserted'),
            'parse_date': datetime.today().strftime('%Y-%m-%d
             ↪  %H:%M:%S')
        }
        insert(my_data, ls)


def get_cursor(url: str, ticker: str, cursor_number: str) ->
 ↪  str:
    link = url.format(ticker, cursor_number)
    session = requests.Session()
    data = session.get(link, headers=headers, stream=True)
    cursor_number =
     ↪  data.json().get('payload').get('nextCursor')
    return cursor_number


def get_instruments(lst: list):
```

```
    data = []
    for item in lst:
        line = f"{item.get('briefName')} !!
        ↪ {item.get('ticker')} !! {item.get('price')} !!
        ↪ {item.get('lastPrice')}"
        data.append(line)
    return data


def convert_inserted(line):
    utc_time = datetime.strptime(line,
    ↪ '%Y-%m-%dT%H:%M:%S.%f%z')\
        .replace(tzinfo=timezone('utc')).strftime('%Y-%m-%d
        ↪ %H:%M:%S')
    return utc_time


from tqdm import tqdm

link = 'https://www.tinkoff.ru/api/invest-gw/social/v1 \
        /post/instrument/{}?limit=50 \
        &appName=invest&platform=web&cursor={}'
def get_data_from_ticker(ticker: str) -> None:
    max_cursor = '99999999999'
    for _ in tqdm(range(1, 2000)):
        get_data_from_api(link, ticker, max_cursor)
        sleep(0.01)
        max_cursor = get_cursor(link, ticker, max_cursor)
```

```
        sleep(0.5)
        print('Всего данных:', len(ls))

ls = []

securites = ['SBER', 'GAZP', 'YNDX', 'LKOH', 'TCS']
for security in securites:
    get_data_from_ticker(security)

df = pd.DataFrame(ls)
df.to_csv('out.csv')
```

Приложение 2. Fine-tuning BERT MLM

Ссылка на данные, URL

```
#Для обучения необходим мощный графический ускоритель, в
↪ работе использовался Nvidia Tesla V100
!nvidia-smi

!pip install transformers
#!pip install pytorch-lightning
!pip install datasets

data_path = './data/final_out_diploma.csv'
sample_path = './data/sample.csv'
json_path = './data/train.json'
uns_path = './data/uns42.csv'
```

```
12
13 from datasets import load_dataset
14
15 dataset_mlm = load_dataset('csv', data_files= data_path,
    ↪  split = 'train')
16
17 from transformers import AutoTokenizer, AutoModelForMaskedLM
18 import torch
19
20 tokenizer = AutoTokenizer.from_pretrained("DeepPavlov/rubert
    ↪  \
21                                     -base-cased-conversational")
22
23 #Загружаем модель с checkpoint, в прошлый раз обучение
    ↪  прервалось
24 model = AutoModelForMaskedLM.from_pretrained("./results_clf/
    ↪  \
25
    ↪  checkpoint-103").to('cuda')
26
27 def preprocess_function(examples):
28     return tokenizer(examples['text'], truncation=True,
        ↪  max_length = 512, padding = 'max_length')
29
30 tokenized_data = dataset_mlm.map(preprocess_function, \
31     remove_columns=dataset_mlm["train"].column_names)
32
33 from transformers import DataCollatorForLanguageModeling
```

```
34
35 data_collator =
    ↪ DataCollatorForLanguageModeling(tokenizer=tokenizer,
    ↪ mlm_probability=0.15)
36
37 from transformers import AutoModelForMaskedLM,
    ↪ TrainingArguments, Trainer
38
39 training_args = TrainingArguments(
40
41     output_dir="./results",
42     evaluation_strategy="epoch",
43     save_strategy='epoch',
44     learning_rate=2e-5,
45     num_train_epochs=5,
46     weight_decay=0.01,
47     per_device_train_batch_size = 16,
48     save_total_limit=1,
49     metric_for_best_model='loss',
50     greater_is_better=False,
51     load_best_model_at_end=True,
52     fp16=True,
53     fp16_full_eval=False
54
55 )
56
57 trainer = Trainer(
58     model=model,
```

```
    args=training_args,
    train_dataset=tokenized_data["train"],
    eval_dataset=tokenized_data["test"],
    data_collator=data_collator

)

trainer.train()

#Сохраняется в итоге лучшая модель с точки зрения loss
```

Приложение3. Обучение и прогнозы с помощью BERT. Задача классификации

```
def preprocess_labels(label):
    if label == 'negative' or label == -1:
        label = 0
        return label
    if label == 'neutral' or label == 0:
        label = 1
        return label
    if label == 'positive' or label == 1:
        label = 2
        return label
    else:
        label = 3
    return label
```

```
14
15 def preprocess_data(data, label):
16     data = data.dropna(axis=0)
17     data['label'] = data[label].apply(lambda x:
       ↪  preprocess_labels(x))
18     data = data[['text','label']]
19     data = data[data.label.isin([0,1,2])]
20     return data
21
22 import torch
23 from torch.utils.data import Dataset
24 from torch.utils.data import random_split
25
26 class TPDataset(Dataset):
27
28   def __init__(self, task, dtype, data_path, tokenizer,
     ↪  labels, max_token_len: int = 256):
29     self.data_path = data_path
30     self.tokenizer = tokenizer
31     self.max_token_len = max_token_len
32     self.labels = labels
33     self.task = task
34     self.dtype = dtype
35     self._load_data()
36
37   def _load_data(self):
38     if self.task == 'train':
39         if self.dtype == 'csv':
```

```
            data = pd.read_csv(self.data_path)
            self.data = preprocess_data(data, self.labels)
        if self.dtype == 'json':
            data = pd.read_json(self.data_path)
            self.data = preprocess_data(data, self.labels)
    else:
        self.data = pd.read_csv(self.data_path)

def add_new_data(self, new_data_path):
    data = pd.read_csv(new_data_path)
    data = preprocess_data(data)
    self.data = self.data.append(data)

def __len__(self):
    return len(self.data)

def __getitem__(self, index):
    item = self.data.iloc[index]
    text = str(item['text'])
    labels = torch.tensor(item['label'], dtype=torch.long)
    tokens = self.tokenizer.encode_plus(text,
                                        add_special_tokens =
                                        ↪ True,
                                       # return_tensors =
                                       ↪ 'pt',
                                        truncation = True,
                                        max_length =
                                        ↪ self.max_token_len,
```

```
                                        return_attention_mask=True
    if self.task == 'train':
        return {'input_ids': tokens.input_ids,
         'attention_mask': tokens.attention_mask,
         'labels': labels}
    else:
        return {'input_ids': tokens.input_ids,
         'attention_mask': tokens.attention_mask}


full_ds = TPDataset(task = 'eval',dtype = 'csv',data_path =
 data_path, tokenizer = tokenizer, labels = 'text')
TP_ds = TPDataset(task = 'train',dtype = 'csv',data_path =
 uns_path, tokenizer = tokenizer, labels = 'label')
tokenizer =
 AutoTokenizer.from_pretrained("DeepPavlov/rubert-base-cased-conver
train_size = int(0.9 * len(TP_ds))
test_size = len(TP_ds) - train_size
train_ds, test_ds = torch.utils.data.random_split(TP_ds,
 [train_size, test_size],
 generator=torch.Generator().manual_seed(1488))
cl_dataset = {'train': train_ds, 'test': test_ds}
from transformers import DataCollatorWithPadding

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
from transformers import AutoModelForSequenceClassification,
 TrainingArguments, Trainer
```

```
model =
 ↪  AutoModelForSequenceClassification.from_pretrained("./results_clf_
 ↪  num_labels=3)
import numpy as np
from datasets import load_metric
metric = load_metric("accuracy")
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=1)
    return metric.compute(predictions=predictions,
     ↪  references=labels)

training_args = TrainingArguments(
    evaluation_strategy = 'epoch',
    output_dir="./results_clf_final",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=20,
    weight_decay=0.01,
    save_strategy = "epoch",
    load_best_model_at_end = True,
    #fp16=True,
    #fp16_full_eval=False,
    #label_names = attributes,
    save_total_limit=1,
    logging_strategy = 'epoch',
    metric_for_best_model = 'accuracy'
```

```
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=cl_dataset["train"],
    eval_dataset=cl_dataset["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics
)
trainer.train()
import mlflow
mlflow.end_run()
predictions = trainer.predict(full_ds)
from torch.nn import Softmax
sm = Softmax(dim=1)
preds = np.array(sm(torch.tensor(predictions[0])))
data_out['negative'], data_out['neutral'],
 ↪  data_out['positive'] = preds[:,0], preds[:,1], preds[:,2]
data_out['pred_unsertanty'] = (1 - np.max(preds,axis=1))
data_out.to_csv('./data/labeled.csv')
```

Приложение 4. Создание индекса тольности

```
def preprocess_labels2(label):
    if label == 0:
```

```
        label = -1
        return label
    if label == 1:
        label = 0
        return label
    if label == 2:
        label = 1
        return label

def preprocess_data(data):
    data.labels = data.labels.apply(lambda x:
    ↪ preprocess_labels2(x))
    return data
df_labeled = pd.read_csv('labeled.csv')
df_labeled['score'] = df_labeled['likesCount'].apply(lambda
↪ x: 1 + math.log(1 + x))
import numpy as np
df_labeled['labels'] =
↪ np.argmax(np.array(df_labeled[['negative','neutral','positive']]),
↪ axis=1)
def one_ticker(x):
    i = len(x.split(" !! "))
    return i
df_labeled_ot =
↪ df_labeled.loc[df_labeled['instruments'].map(one_ticker)
↪ <= 4]
df_labeled_ot = df_labeled_ot.drop(columns = 'Unnamed: 0')
df_labeled_ot.ticker.value_counts()
```

```
df_labeled_ot["date"] = pd.to_datetime(df_labeled_ot["date"])
preprocess_data(df_labeled_ot)
daily_sentiment = df_labeled_ot.pivot_table(index='date',
    columns="ticker", values='labels',
    aggfunc='first').resample("D").mean()
```

Приложение 5. Парсинг финансовых данных

```
tickers =
    ['GAZP.ME','GMKN.ME','LKOH.ME','SBER.ME','TCSG.ME','VTBR.ME','YNDX
import yfinance as yf

stockdata = []
for tickerSymbol in tqdm(tickers):
    tickerData = yf.Ticker(tickerSymbol)
    tickerDf = tickerData.history(period='1d',
        start='2019-08-27', end='2022-4-18')
    tickerDf['ticker'] = tickerSymbol
    tickerDf.columns = map(lambda x: x.lower(),
        tickerDf.columns)
    tickerDf['return'] = tickerDf['close'].pct_change() * 100
    stockdata.append(tickerDf)

df_returns = pd.DataFrame()
for i in stockdata:
    df_returns = df_returns.append(i.reset_index()[1:])
df_returns =
    df_returns.pivot_table(index='Date',columns='ticker',values='retur
```

```
full_data = daily_sentiment.merge(df_returns, left_index=True, right_index=True, suffixes=('_sent', '_return'))
```

Приложение 6. Granger-causality tests

```
from statsmodels.tsa.stattools import grangercausalitytests

maxlag=15
test = 'ssr_chi2test'

def grangers_causation_matrix(data, variables, test='ssr_chi2test', verbose=False):

    df = pd.DataFrame(np.zeros((len(variables), len(variables))), columns=variables, index=variables)
    for c in df.columns:
        for r in df.index:
            test_result = grangercausalitytests(data[[r, c]], maxlag=maxlag, verbose=False)
            p_values = [round(test_result[i+1][0][test][1],4) for i in range(maxlag)]
            if verbose: print(f'Y = {r}, X = {c}, P Values = {p_values}')
            min_p_value = np.min(p_values)
            df.loc[r, c] = min_p_value
    df.columns = [var + '_x' for var in variables]
```

```
    df.index = [var + '_y' for var in variables]
    return df

gcmatrixes = []
df_lags = pd.DataFrame()
for i in daily_sentiment.columns:
    attributes = [col for col in full_data.columns if i in
    ↪ col]
    matrix =
    ↪ grangers_causation_matrix(full_data[attributes].dropna(),
    ↪ attributes)
    gcmatrixes.append(matrix)
    gc =
    ↪ grangercausalitytests(full_data[attributes[::-1]].dropna(),
    ↪ maxlag=15, verbose=False)
    pvals = {i: np.mean([gc[i][0][key][1] for key in
    ↪ gc[i][0].keys()]) for i in range(1,maxlag)}
    aics = [gc[i][-1][1].bic for i in range(1,maxlag)]
    df = pd.DataFrame({i: pvals}).T
    df['optimal_lag'] = np.argmin(aics) + 1
    df_lags = df_lags.append(df)
```

Приложение 7. Создание признаков

```
def processing_for_fin(df, sent_lag = 1, returns_lag = 1):

    days_of_week =
    ↪ pd.get_dummies(df.reset_index()['index'].dt.dayofweek,
```

```
                                prefix='weekday',
                                drop_first=True)
    days_of_week.index = df.reset_index()['index']

    months =
     ↪  pd.get_dummies(df.reset_index()['index'].dt.month,
                                prefix='month',
                                drop_first=True)
    years = pd.get_dummies(df.reset_index()['index'].dt.year,
                                prefix='year',
                                drop_first=True)

    months.index = df.reset_index()['index']
    years.index = df.reset_index()['index']

    if sent_lag:
        for i in range(sent_lag):
            df[f'sent_lag_{i+1}'] = df.iloc[:,0].shift(i+1)
    for j in range(returns_lag):
        df[f'return_lag_{j+1}'] = df.iloc[:,1].shift(j+1)

    df = pd.concat((df, days_of_week), axis=1)
    df = df.drop(columns = df.columns[0])
    df = pd.concat((df, months), axis=1)
    df = pd.concat((df, years), axis=1)
    df = df.dropna()

    y = df.iloc[:,0]
```

```
    X = df.iloc[:, 1:]
    return y, X
```

Приложение 8. F-test

```
from scipy import stats
def f_test(mse_ur, mse_r, X, q):
    n = len(X)
    k_ur = len(X.columns) + 1
    W = (mse_r - mse_ur) / mse_ur
    stat = (n-k_ur) / q * W
    p_val = 1 - stats.f.cdf(stat, dfn=q, dfd=n-k_ur)
    return {'p_value': np.round(p_val,2), 'stat value': stat}
```