Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

Факультет экономических наук

Образовательная программа «Экономика»

БАКАЛАВРСКАЯ ВЫПУСКНАЯ РАБОТА

«Влияние данных Твиттера на цены акций российских банков и их волатильность»

Выполнил
Студент группы № БЭК 165
Цвигун Аким Олегович

Научный руководитель:
Старший преподаватель
Демешев Борис Борисович

Москва 2020

Оглавление

1 Введение

Фондовый рынок является существенным фактором развития экономики: он предоставляет инструменты и механизмы для распределения ресурсов между секторами. Несмотря на небольшую стагнацию в последние три года, за последние 5 лет объем торгов на фондовом рынке Московской биржы вырос практически в два раза, с 20.6 триллионов рублей до 40.7 [*Сайт Москвоской Биржи*, URL]. Фондовый рынок становится особенно важным в периоды резких социальных потрясений: в такие отрезки грамотные инвесторы зарабатывают крупные суммы, в то время как менее подготовленные прогорают. Показательным был случай с обвалом акций российских компаний на Московской бирже 10.03.2020: на открытии торгов после трех выходных дней акции «Роснефти» и «Сбербанка» упали более, чем на 10% вследствие рекордного падения мировых цен на нефть [*Акции российских компаний обвалились на открытии Московской биржи*, 2020; *Московская биржа начала расти впервые за неделю. Почему?*, 2020]. Мировые цены на нефть, в свою очередь, обрушились после обострения ситуации с коронавирусом, COVID-19 [*Coronavirus: Oil price collapses to lowest level for 18 years*, 2020; *Цены на нефть обрушились на 30%*, 2020]. Данная динамика проявилась не только на отечественном рынке: на закрытии торгов в США 12 марта промышленный индекс Dow Jones упал на целых 9,99% [*Yahoo*, URL; *Московская биржа начала расти впервые за неделю. Почему?*, 2020].

В таких условиях становится особенно важно прогнозировать динамику цен акций лучше конкурентов. Так, обвал акций московской биржи предсказывался многими инвесторами за несколько дней до открытия [*Мосбиржа подготовилась к обвалу на российских торгах*, 2020]. Однако в современном информационном мире такое большое количество мнений и новостей, что уследить за всеми невозможно. В этой связи воз-

никает необходимость в искусственном интеллекте, способном обрабатывать крупные объемы информации и делать на их основе грамотные и корректные выводы. В работе предлагается новый способ предсказания динамики цен акций в зависимости от данных Твиттера (далее – твитов). Все твиты, содержащие название компании, анализируются на тональность несколькими моделями, результаты за день усредняются, и формируются признаки для прогнозирования цен акций компании на следующий день. Таким образом, актуальность работы обусловлена необходимостью анализа большого объема текстовой информации для улучшения предсказаний динамики цен.

Твиттер был выбран в качестве источника данных неслучайно: это социальная сеть, в которой люди высказывают свои мнения относительно происходящих событий, что встречается довольно редко, например, в «Вконтакте», в «Одноклассниках» или в «Инстаграме». «Фейсбук» не был взят потому, что он запрещает парсинг и скачивание данных при отсутствии на то письменного разрешения [*Facebook*, URL].

Исследование проводится именно в отношении российских банков, поскольку прибыль банков во многом определяется объемом обслуживания физических лиц. В свою очередь, анализ тональности твитов может отражать усредненное мнение людей насчет банка. Так, если в какой-то момент твиты приобретают негативный оттенок, это означает, что усредненное мнение людей нас чет банка ухудшилось, а следовательно, люди менее вероятно предпочтут этот банк другому, что негативно отразится на прибыли банка и, как следствие, на ценах его акций. Оценивается статистическая связь данных и котировок акций трех банков: Сбербанк, ВТБ и Росбанк.

Научная новизна и практическая значимость работы обусловлены сразу несколькими факторами. Во-первых, это отсутствие каких-либо исследований с количественными выводами зависимости фондового рынка

от российских социальных сетей. В существующих работах [А.В.Дубко, К.Буассье, 2015; Лысенко, 2018] не приводится количественного анализа влияния, что не позволяет сделать достоверное заключение о наличии взаимосвязи между общественным настроением и динамикой акций.

Следующим фактором является недостаток предобученных моделей обработки языка на русском языке. На данный момент существуют следующие предобученные архитектуры: ELMO [*ELMO*, URL], ULMFiT [*ULMFiT*, URL] и BERT [*BERT*, URL], однако каждая из них обладает существенными недостатками, о которых будет рассказано далее. В работе предлагается модель ULMFiT, предобученная на большом корпусе данных Твиттера [Ю. Рубцова, 2014].

Наконец, предыдущие исследования не использовали архитектуры глубокого обучения для анализа тональности, что, возможно, не позволяло им обнаружить сложные зависимости. В единственной работе, где используется нейронная сеть Word2Vec [Sasank (и др.), 2016], делается вывод, что использование простой модели N-gram позволяет достичь лучшего качества, чем Word2Vec. Между тем архитектуры глубокого обучения являются очень мощным инструментом анализа текстов: их преимущество перед базовыми моделями подтверждено на всех общедоступных наборах данных [*Neurohive*, URL; *Paper With Code*, URL].

Целью исследования является определение наличия зависимости динамики цен акций и их волатильности от тональности данных Твиттера. В ходе ее достижения были решены следующие задачи:

1. Предобучение модели ULMFiT для русского языка на корпусе данных Твиттера.

2. Получение различных обучающих выборок и двухступенчатое обучение моделей на имеющихся данных.

3. Получение данных Твиттера, генерация предсказаний моделей и

применение нелинейных функций к признакам.

4. Создание двух групп моделей для предсказания динамики цен акций: с использованием тональности твитов и без нее.

5. Оценка влияния признаков тональности на котировки акций при помощи нескольких тестов и дальнейший вывод.

Работа устроена следующим образом. Глава 2 посвящена обзору литературы, затрагивающей тему влияния новостных источников на динамику акций компаний. Затем, в главе 3, рассматриваются современные подходы к решению задач обработки языка (Natural Language Processing -– далее NLP): имеющиеся архитектуры, их ограничения по отношению к русскому языку, после чего выбирается оптимальная модель для исследования и изучается ее устройство. Глава 4 посвящена сбору данных. В ней содержится описание методов сбора данных и их предобработки для предобучения модели, получения размеченных данных и их предобработки для ее дальнейшего обучения, данных Твиттера, содержащих названия банков и данных по ценам акций. После этого, в главе 5, происходит генерация предсказаний тональности твитов и формирование признаков для моделей предсказания цен акций. В ней также рассматриваются различные аспекты обучения, такие как учет инфляции. Глава 6 содержит описание выбора модели без признаков тональности и с ними. Затем проводятся различные тесты для сравнения моделей, после чего происходит подведение итогов анализа и формирование выводов. Наконец, в заключительной части работы подводятся общие итоги исследования и рассматриваются потенциальные направления для дальнейших исследований в данной области.

2 Обзор литературы

В данной главе рассматривается литература, в которой изучается зависимость фондового рынка от новостных источников. На первом шаге рассматривается литература 20 века, в которой проводится исследование зависимости котировок акций как от социальных медиа, так и от средств массовой информации, для более полного формирования картины в области применения методов обработки языка к финансовому рынку. На втором шаге проводится обзор работ, изучающих влияние непосредственно Твиттера на фондовый рынок. Последний шаг посвящен обзору исследований зависимости фондового рынка от социальных сетей в области русского языка.

2.1 Литература 20 века

Влияние новостей на котировки акций стало изучаться еще в 1970-е [Niederhoffer, 1971; Schwert, 1981], однако широкой огласки исследования не получили. В силу отсутствия требуемых вычислительных мощностей и неразвитости области NLP вплоть до начала 2000-х годов целью большей части исследований являлись не численные выводы, а способы нахождения зависимости фондового рынка от новостей: [Cootner, 1964; Fama, 1991; Jensen, 1978] в своих работах приводили аргументы в пользу наличия данной взаимосвязи. Butler et al. [Butler, Malaikah, 1992] исследуют доходность акций на рынке Саудовской Аравии и Кувейта в период с 1985-1989. В работе они приходят к выводу, что большое влияние на формирование доходности акций крупных компаний оказывают новости, затрагивающие их индустрию.

French et al. [R.French, Roll, 1986] исследовали причины большей волатильности цен активов в часы работы биржи, чем в другое время.

В качестве основной причины авторы выделили появление публичной информации, которое чаще происходит в рабочее время. Появление новой информации зачастую приводит к росту неопределенности, что, в свою очередь, приводит к росту волатильности. Тем не менее, статья не содержит каких-либо численных доказательств влияния публичной информации на цены активов.

Опять же ввиду слабых компьютерных мощностей исследования, нацеленные на получение числовых выводов, специализировались не на анализе самих новостей, а на их численных характеристиках: длина заголовков новостей, дамми-переменная, отвечающая за наличие как минимум одного из семнадцати основных макроэкономических заявлений или число новостей, содержащих слова-тригеры (например, “dividend”) [Ederington, Lee, 1993; Mitchell, Mulherin, 1994]. Однако даже такие «игрушечные» признаки оказались весомыми: в своей работе M.L. Mitchell и J. H. Mulherin [Mitchell, Mulherin, 1994] пришли к выводу, что перечисленные выше признаки являются значимыми при прогнозировании активности рынка. Тем не менее, более сложных зависимостей авторам найти не удалось: зависимость между признаками, полученными с помощью анализа текстов примитивными моделями, и активностью на рынке, хоть и носила робастный характер, оказалась практически незначительной.

Thompson et al. [Thompson, Olsen, Dietrich, 1987] изучали доходность акций компаний на следующий день после публикации новостей о дивидендах компании. Автор пришли к выводу, что доходность в такие периоды существенно отличается от доходности в обычные дни (когда отсутствуют новости о дивидендах), приведя соответствующие значения t-статистики и F-статистики.

Kim et al. [Kim, E.Verrecchia, 1994] также исследовали динамику фондового рынка в момент публикации объявлений о дивидендах, но вме-

сто доходности акций объектом исследования была ликвидность рынка. Авторы показали, что вследствие асимметрии информации в момент сообщения бид-аск спред (разница между ценой, которую готов заплатить покупатель и той, по которой продавец готов продать) растет, что приводит к снижению рыночной ликвидности.

2.2 Литература, изучающая статистическую связь социальных сетей с котировками акций

С ростом популярности социальных сетей анализ длинных новостей стал заменяться анализом коротких записей. Сложность анализа новостей заключается в том, что даже среднестатистический человек не всегда может понять тональность новости (положительная / отрицательная) -– следовательно, простые на тот момент архитектуры определяли положительность новостей с большой ошибкой. Помимо этого, зачастую новости могут быть положительными для одной стороны и негативными для другой, что также вносит ошибку в предсказания. Анализировать социальные сети проще -– негативные записи, как правило, содержат грубую или нецензурную лексику. Более того, когда человек выражает свое мнение, как правило, он делает это однозначно -– а новости могут содержать как позитивные, так и негативные аспекты.

В 2000-е, с бурным развитием машинного обучения начали разрабатываться архитектуры для решения NLP задач. Одной из них стала BoosTexter [Schapire, Singer, 2000] -– система для классификации текстов, использующая алгоритм AdaBoost. Она легла в основу модели OpinionFinder (OF) [Wilson (и др.), 2005]. Модель использовала два классификатора BoosTexter и была обучена на корпусе текстов MPQA [*The MPQA Opinion Corpus*, URL]. Модель OpinionFinder, продемонстрировав свою мощь на различных задачах NLP [Banea (и др.), 2008; He (и др.), 2008],

легла в основу нескольких важных исследований.

Настоящим открытием стала работа Bollen J., Mao H. и Zeng X.J. “Twitter mood predicts the stock market” в 2011 году [Bollen, Mao, Zeng, 2011]. Авторы исследовали зависимость индекса Dow Jones Industrial Average (DIJA) от шести направлений тональности твитов за последнюю неделю, спокойствие (calm), тревожность (alert), уверенность (sure), энергичность (vital), доброта (kind) и радость (happy), полученных при помощи модели Google-Profile of Mood States и тональности твитов в классификации «положительный – отрицательный», измеренной с помощью OpinionFinder. Результаты оказались действительно впечатляющими: со средней точностью прогнозирования (Accuracy) направления изменения цены акций (рост / падение) в 87.6% средняя абсолютная ошибка в процентах (далее – MAPE) предсказаний значений DIJA снизилась на целых 6%. На уровне значимости 10% важными оказались результаты OF за последний день, тональность спокойствия за второй – шестой предыдущие дни и неожиданно тональность радости, полученной шесть дней назад; на уровне значимости 5% -– только значение спокойствия со второго по пятый предыдущие дни (рис. 1). Тот факт, что тональность радости шестидневной давности важнее ее же за любой из последних дней, наталкивает на мысль, что результат для данного признака мог получиться случайным. Вполне логично, что при небольшом объеме данных (авторы использовали показатели DIJA и данные Твиттера с 28.02.2008 по 19.12.2008, то есть за 295 дней) как минимум один из 49 векторов признаков получится коррелирующим с целевой переменной на таком высоком уровен значимости (10%). Тем не менее, динамика p-значений для тональности спокойствия за последнюю неделю получилась адекватной -– начиная со значения за второй день назад, p-значение признаков монотонно убывало, что подтверждает наличие зависимости между данной характеристикой тональности и значением DIJA.

Продолжением работы стало исследование Anshul Mittal и Arpit Goel “Stock Prediction Using Twitter Sentiment Analysis” [Mittal, A.Goel, 2011]. Авторы отметили несовершенство предыдущей работы в первую очередь в области оценки качества и измерения метрик. Тот факт, что для оценивания модели авторы использовали только одно разбиение «трейн-тест» с 2-х месячным периодом отложенной выборки, означает, что влияние 49 коэффициентов измерялось на выборке в 61 наблюдение, что вполне могло привести к случайным или желаемым, но не действительным результатам. Более того, результаты метрик Accuracy и MAE по такой же причине могли получиться нерепрезентативными. В этой связи авторы предложили новый метод оценивания модели, последовательную кросс-валидацию (sequential cross-validation). В классической кросс-валидации выборка разбивается на k частей (фолдов), и для каждого фолда модель обучается на оставшейся выборке и тестируется на данном фолде, после чего получившиеся значения метрик усредняются. Соответственно, такой метод оценивания не применим для временных рядов, поскольку в нем не учитывается дата наблюдения. В последовательной кросс-валидации выборка делится на k фолдов, но в отличие от классической кросс-валидации в качестве обучающей выборки берется не вся оставшаяся часть, а только та, которая идет до k-го фолда, а в качестве отложенной -– k-й фолд. Такой подход, напротив, может быть использован в случае с временным рядом, так как для каждого фолда обучение проходит строго на более ранних данных. Результаты, полученные путем последовательной кросс-валидации, оказались хуже, чем в [Bollen, Mao, Zeng, 2011]: авторы получили Accuracy, равную 75.56% (вместо предыдущих 87.6%), однако такой подход является более корректным (и более «честным»). Серьезный недостаток работы заключается в итоговой модели. Ее результаты (рис. 2) очень схожи с предсказаниями модели «такой же как вчера»: если сдвинуть предсказания (синяя линия) на 1 шаг влево,

то полученный тренд будет практически идеально реплицировать истинный. Это означает, что модель очень сильно зависит от значения целевой переменной на предыдущем шаге, а остальные регрессоры оказывают на нее незначительное влияние.

Большое исследование провели Vu [Vu (и др.), 2013]. Изучалось влияние тональности данных Твиттера на цены акций четырех компаний: Apple, Google, Microsoft и Amazon. Для каждой компании отбирались твиты, содержащие их названия или названия их продуктов: iPhone, MacBook и т.д. для Apple; Gmail, Chrome и т.д. для Google; Xbox, Explorer и т.д. для Microsoft; AWS и т.д. для Amazon. Затем данные предобрабатывались несколькими способами так, чтобы удалить все потенциально зашумленные данные. Вследствие этого бóльшая часть твитов была удалена. В качестве модели оценивания их тональности использовалась Twitter Sentiment Tool [Go, Bhayani, Huang, 2009], модель, разработанная специально для измерения тональности твитов. В результате исследования авторы получили интересный вывод – для трех компаний, Apple, Google и Amazon признак тональности их твитов оказался значимым на уровне значимости 5%, для Microsoft -– незначительным. По мнению авторов, причиной могло стать небольшое число твитов, релевантных для Microsoft, за день: более чем за 20 дней данные для Microsoft после процедуры предобработки отсутствовали в принципе.

Sasank et al. в работе “Sentiment Analysis of Twitter Data for Predicting Stock Market Movements” [Sasank (и др.), 2016] продолжили исследование зависимости для компании Microsoft. В отличие от предыдущей работы [Vu (и др.), 2013], выводом стала выраженная зависимость между направлением движения акций компании Microsoft и общественным настроением. В данном исследовании впервые была использована модель «глубокого обучения» для оценивания тональности твитов (общественного настроения) – нейронная сеть Word2Vec. Несмотря на большое чис-

ло параметров и общественное признание, наилучший классификатор, построенный на Word2Vec, получился хуже аналогичного, построенного на N-gram -– модель, очень похожая на мешок слов (Bag of Words), где словарь составляют не обычные слова, а последовательности N слов подряд. Точность предсказаний направления движения акций в работе получилась равной 70.5%, однако авторы не привели ни p-значение для коэффициентов, ни точность предсказаний без признаков тональности твитов -– возможно, сравнимую точность можно было бы получить без использования результатов N-gram.

Kirlić et al. также продолжили изучать влияние тональности твитов на цены акций Microsoft в работе "Stock market prediction using Twitter sentiment analysis" [Kirlić (и др.), 2018], однако на этот раз для оценивания общественного настроения использовались только записи, содержащие слово "Microsoft". Авторы получили значение корреляции тональности твитов и цен акций, равное 0.7815. Таким образом, с улучшением общественного настроения росли и котировки акций Microsoft.

2.3 Литература, посвященная анализу зависимости для русского языка

Исследования, посвященные статистической взаимосвязи тональности данных русскоязычного Твиттера и фондового рынка, практически отсутствуют. Существует несколько причин этого фактора, основная -– недостаточная развитость области NLP для русского языка. В работе Дубко [А.В.Дубко, К.Буассье, 2015] описан только способ проведения исследования. Лысенко [Лысенко, 2018] и Андрианова [Андрианова, Новикова, 2018] также рассмотрели лишь методы предсказания цен фондового рынка, используя анализ тональности. Работы с приведением соответствующих численных выводов отсутствуют в принципе.

Между тем, анализ развивающихся фондовых рынков, к которым относится российский фондовый рынок, существенно отличается от анализа развитых рынков вследствие значительно более быстрого роста, несовершенства институтов и иных особенностей [Bekaert, Harvey, 1995]. В этой связи использование методов нахождения зависимости между тональностью общественного мнения и ценами акций, используемых в иностранных исследованиях, может быть некорректным применительно к России.

Таким образом, исследование влияния тональности данных российских социальных сетей на фондовый рынок является своего рода авангардом. Помимо этого, как уже было отмечено, бóльшая часть исследований в области английского языка содержит уязвимые места, такие как некорректный способ оценивания модели или отсутствие сравнения с моделями без используемых признаков, которые отдельно рассмотрены в данной работе.

3 Современные подходы к решению NLP задач

В главе рассматриваются современные подходы к решению задач NLP: имеющиеся архитектуры, их ограничения по отношению к русскому языку и исследованию. Затем выбирается оптимальная для исследования модель (ULMFiT) и изучается ее устройство и принцип работы.

3.1 Краткий экскурс в NLP

Прежде чем проводить обзор архитектур NLP, необходимо рассмотреть некоторые особенности данного раздела машинного обучения. Нач-

нем с основополагающей идеи всех современных подходов -– численных представлений слов. Для того, чтобы компьютер мог работать со словами, они должны быть представлены в виде векторов. Долгое время для получения векторного представления слова использовался подход Bag of Words (BOW). Идея метода проста: создается словарь имеющихся слов длины n. Слово под номером i в словаре представлено вектором, где i-й элемент равен единице, а все остальные -– нулю. Такой подход имеет множество недостатков, главным является большая размерность векторов, что делает модель крайне ресурсозатратной. Если уменьшить число слов в словаре, многие нечасто встречающиеся слова, которые могли бы позволить сделать однозначный вывод (о тональности текста в нашем случае), будут проигнорированы. С другой стороны, с ростом размера словаря растет размерность векторов, что приводит к значительному увеличению требуемых вычислительных мощностей. Более того, BoW игнорирует похожесть слов: все пары слов имеют косинусное сходство, равное нулю. Это означает, что подход никак не учитывает близость слов, а значит, не может корректно обобщать текст.

Изучение двух вышеописанных проблем привело к идее эмбеддингов слов (word embedding) -– векторов с размерностостью 100-1000 для каждого слова. Подход был предложен еще в работах [Bengio, Ducharme, Vincent, 2001; Bengio, Ducharme, Vincent, Jauvin, 2003], однако до воскрешения глубокого обучения в 2006 оставался непопулярным. Такой подход позволяет не только существенно сократить размерность векторов, но и учитывать близость слов: близкие слова должны иметь схожие вектора. Схожесть векторов можно измерять многими способами, наиболее популярным является косинусное сходство $(1 - \frac{u \cdot v}{||u||_2||v||_2})$. Модель Word2Vec, предложенная в работе [Mikolov (и др.), 2013], стала первой широко используемой архитектурой, основанной на эмбеддингах. Исследование показало, что при обучении модели на большом наборе данных (в работе

использовался набор данных на 1.6 миллиарда слов) с получившимися векторами можно даже проводить арифметические операции: вектор слова «король» минус вектор слова «мужчина» плюс вектор слова «женщина» был наиболее близок к вектору слова «королева».

Однако эмбеддинговый подход, подобно человеку, требует предобучения: перед использованием модели на требуемых данных она должна быть обучена на большом наборе данных. Концепция использования параметров, полученных с помощью предобучения на большом наборе данных, на целевой задаче называется перенос знаний (transfer learning). Идея переноса знаний заключается в имитации способности человека приобретать знания при решении одной задачи, а затем использовать эти знания для решения другой. Концептуальное различие с традиционным машинным обучением заключается в том, что в классическом подходе две модели обучаются отдельно без сохранения или передачи знаний от одной к другой. В переносе знаний, напротив, происходит сохранение знаний (например, весов или признаков), полученных при обучении первой модели, а затем их использование для обучения второй. В этом случае задача, которую решает первая модель, называется исходной задачей (source task), а задача, которую решает вторая модель – целевой задачей (target task) [Faltl, Schimpke, Hackober, 2019].

Почему перенос знаний необходим? Проведем простую аналогию с человеком. Допустим, нам требуется научиться выделять определенную информацию из текстов на языке, который нам неизвестен. Если мы будем обучаться этому языку только на требуемых данных, мы столкнемся с двумя проблемами. Во-первых, в абсолютном большинстве случаев требуемый набор данных не настолько большой, чтобы на нем можно было выучить язык. Соответственно, обучаясь только на целевом наборе данных, произойдет переобучение -– явление, при котором модель хорошо работает на наблюдениях из обучающей выборки, но относительно плохо

делает предсказания на объектах из отложенной выборки. Это связано с тем, что в процессе ее обучения в обучающей выборке обнаруживаются некоторые случайные закономерности (в нашем случае это языковые особенности), которые отсутствуют в генеральной совокупности (в самом языке), в результате чего модель теряет способность к обобщению, настраиваясь под специфичный набор данных. Во-вторых, в тренировочной выборке могут отсутствовать различные слова и языковые конструкции. Следовательно, в случае их попадания в отложенную выборку, модель не сможет их правильно обработать, даже если по смыслу они идентичны словам и выражениям, в изобилии присутствующим в тренировочной выборке. В этой связи необходимо, чтобы модель умела правильно анализировать как можно больше различных языковых конструкций

Важным аспектом предобучения является сходство лексики выборки для предобучения и целевого набора данных. В случае, если модель предобучена на литературном языке, а используется на разговорном, который преобладает, например, в социальных сетях, модель, опять же, будет иметь дело с многими конструкциями, которые она раньше не встречала, и может интерпретировать их неправильно или, что хуже, переобучиться на них.

3.2 Обзор современных архитектур

NLP развивается очень быстро: так называемые State of The Art (SOTA) модели, оптимальные в рамках конкретных задач, сменяют друг друга по несколько раз за год. Поскольку использование NLP в работе необходимо для определения тональности твитов, требуется рассмотреть текущие SOTA архитектуры в области анализа тональности. Список SOTA моделей взят с платформы [*Papers With Code*, URL], в которой

доступны SOTA архитектуры по всем задачам машинного обучения.

Практически все современные модели в силу большого числа параметров требуют для обучения использование GPU – дополнительного графического процессора. Сервис GPU доступен на платформе Google Colab, однако для обучения текущих SOTA моделей с нуля (без использования переноса знаний) предоставляемого размера GPU недостаточно. В этой связи лучшая моделью с точки зрения метрики Accuracy, T5 – Text-to-Text Transformer, архитектура от Google, не рассматривалась при выборе модели для предсказания тональности, поскольку для данной модели отсутствует предобученная версия для русского языка.

Следующий класс моделей – производные BERT-a: ALBERT, RoBERTa, SpanBERT, DistilBERT и непосредственно оригинальная BERT – все они также являются ресурсозатратными, однако для исходной BERT существуют предобученные версии для русского языка [*BERT*, URL]. Тем не менее, обе предобученные для русского языка модели обучены на языке, по стилю схожим с литературным – на корпусе текстов Википедии и новостных источников. Язык, используемый в социальных сетях и, в частности, в Твиттере, является разговорным и существенно отличается от литературного: в нем присутствуют многие конструкции, которые не встречаются в используемом наборе данных: например, ни в новостях, ни в Википедии не используются фразы приветствия («Добрый день» / «Здравствуйте» и т.д.), однако их использование в социальных сетях в большинстве случаев намекает на положительность твита. Более того, язык в Твиттере содержит множество жаргонизмов, ругательств и прочей лексики, которая также не встречается ни в новостных источниках, ни в Википедии.

Другим большим классом моделей являются рекуррентные нейронные сети: ULMFiT, Block-spabank LSTM, bmLSTM, Bi-CAS-LSTM и так далее. В работе мы не будем углубляться в рекуррентные нейронные

сети, подробнее про них, и, в частности, про архитектуру LSTM написано в [Olah, URL]. Вкратце, рекуррентные сети -– это нейронные сети, в которых предсказание на данном шаге зависит от предсказаний на предыдущих шагах. Таким образом, в задачах NLP каждое новое слово обрабатывается с учетом полученной от предыдущих слов информации – это позволяет учитывать контекст слов. Наилучшей архитектурой среди рекуррентных сетей является ULMFiT, Universal Language Model Fine-tuning for Text Classification. Для данной модели существуют предобученные версии для русского языка [*FastAI*, URL], однако они обладают тем же недостатком, что и предобученные версии BERT. Тем не менее, ULMFiT обладает уникальным преимуществом: в силу небольшого по сравнению с остальными архитектурами числа параметров данную модель можно предобучить самостоятельно, используя сервис Google Colab. В этой связи для проведения исследования была выбрана архитектура ULMFiT.

3.3 Идея ULMFiT

Одна из центральных концепций ULMFiT – индуктивный перенос знаний (inductive transfer learning). Отличие индуктивного переноса знаний от обычного заключается в двух аспектах. Во-первых, исходная задача должна отличаться от целевой. Во-вторых, при решении целевой задачи должны быть доступны размеченные данные – не только набор текстов, но и целевые значения.

Оба требования соблюдаются в нашем конкретном случае, где исходной задачей является предсказание следующего слова в последовательности (language modeling), а целевой задачей – анализ тональности, и наборы данных, на которых мы будем обучать конечные модели, размечены.

Процесс обучения модели можно разделить на три этапа (см. рис. learning). Первый шаг -– предобучение языковой модели (language model) на большом корпусе данных общего характера (general-domain pretraining) – корпус статей Википедии, новостных источников или корпус данных разговорной речи. Языковая модель – модель, предсказывающая следующее слово в последовательности. После предобучения модель с определенной точностью предсказет следующее слово: например, наиболее вероятным продолжением фразы «о возможной» в используемом для предобучения наборе данных Твиттера является слово «отставке» – такое продолжение фразы корректно в 10.33% случаев. На этом этапе модель изучает общие черты языка.

На втором этапе происходит тонкая настройка (fine tuning) языковой модели (language model) на целевом наборе данных. Тонкая настройка -– процесс обучения уже предобученной модели на целевом наборе данных, как правило, сравнительно меньшнм набора данных, использовавшегося для предобучения. В данном случае это обучение на требуемой размеченной выборке. Следуя идее индуктивного переноса знаний, знания, полученные на первом этапе, должны использоваться для целевой задачи. Однако набор данных целевой задачи в большинстве случаев отличается от набора данных исходной задачи. В этой связи языковой модели требуется настроиться под специфику данных целевой задачи, не теряя знаний, полученных на первом этапе, поэтому происходит тонкая настройка с последовательной заморозкой слоев. После тонкой настройки модель, как и после первого шага, способна предсказывать следующее слово в предложении, однако теперь модель это делает, исходя из специфики набора данных целевой задачи. Например, в отличие от результата, полученного на первом шаге, наиболее вероятным продолжением фразы «о возможной» для одного из используемых для обучения наборов данных (выборка с комментариями о банках) является слово «продаже». После-

довательность «о возможной» имеет такое продолжение в 100% случаев.

Третий шаг – обучение классификатора целевой задачи. Поскольку целью является предсказание тональности текста, а не следующего слова, в обученную языковую модель добавляются два линейных слоя, чтобы результатом работы модели было распределение вероятностей по классам (в нашем случае – положительный / нейтральный / отрицательный текст для двух моделей и положительный / отрицательный для третьей).

Далее каждый этап будет рассмотрен более подробно для понимания устройства и принципа работы ULMFiT.

3.4 Предобучение на данных общего характера

Модель для предобучения на данных общего характера, языковая модель, решает задачу предсказания следующего слова в предложении (Рис. 1.1). Архитектура будет рассмотрена на примере прямого прохода нейронной сети (forward pass) для последовательности слов «прямой проход нейронной». Обозначим длину словаря (количество уникальных токенов, для которых модель будет обучать эмбеддинги) за $dictLen$. На вход подается последовательность трех (в общем случае n) слов. Процедура получения их эмбеддингов осуществляется через one-hot кодирование: каждому слову ставится в соответствие вектор длины dictLen, в котором на месте, соответствующем номеру слова в словаре, стоит единица, а все остальные – нули. Для того, чтобы получить эмбеддинги слов, имеющуюся матрицу размерности $(3 \times dictLen)$ требуется умножить на матрицу эмбеддингов -– матрицу размерности $(dictLen \times 400)$, в которой i-я строчка является эмбеддингом i-го слова в словаре. Таким образом, первый слой сети, матрица эмбеддингов, имеет размерность $(dictLen \times 400)$.

Следующие три слоя – три объединенные (stacked) рекуррентные ней-

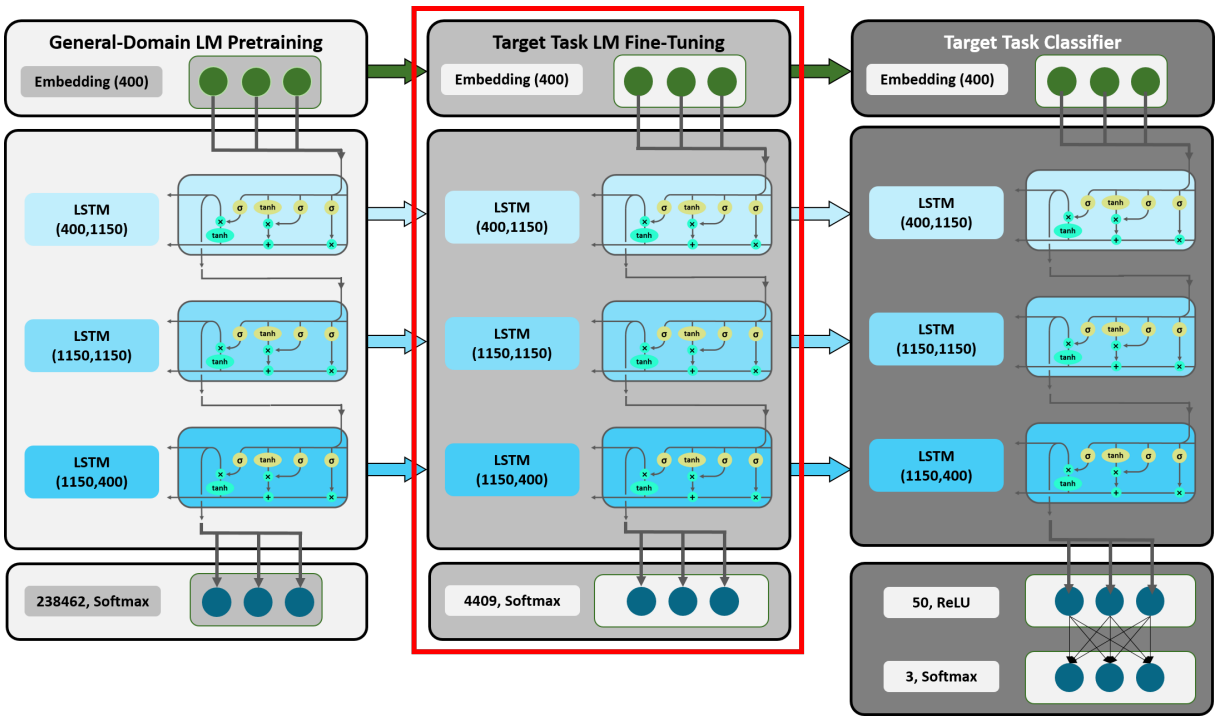

Рис. 1.1: Структура ULMFiT [Faltl, Schimpke, Hackober, 2019].

ронные сети LSTM. Данная структура называется AWD-LSTM (Average Weight-Dropped Long Short Term Memory) [Merity, Keskar, Socher, 2017] и будет рассмотрена более детально позже. Первые две клетки имеют 1150 скрытых нейронов (соответственно, для каждого слова на выходе из обеих клеток получается вектор длины 1150), а последняя – 400 нейронов. Это сделано для того, чтобы размерность вектора, подающегося на вход AWD-LSTM, совпадала с размерностью выходного вектора. Важно отметить, что тот факт, что клетки являются объединенными, означает, что каждый токен проходит сразу через три клетки: результат обработки k-го токена в последовательности для всех клеток не зависит от $(k+1)$-го токена. Соответственно, для клетки i и токена k на вход подается результат обработки k-го токена для $(i-1)$-й клетки (для первой клетки – эмбеддинг k-го токена) в качестве входных данных (input) и выход и состояние памяти после обработки $(k-1)$-го токена i-й клеткой в качестве скрытого состояния и состояния памяти соответственно. Таким образом, на выходе из AWD-LSTM получается матрица размерности

(3×400) ($n \times 400$ в общем случае), в которой каждая из 3-х (n) строчек может быть интерпретирована как вектор, содержащий всю необходимую информацию о первых k токенах для предсказания $(k+1)$-го.

Результатом последнего слоя в архитектуре должно быть вероятностное распределение следующего слова для каждой из последовательностей из первых k токенов. Иными словами, это вероятность каждого токена из словаря быть продолжением последовательностей «прямой», «прямой проход» и «прямой проход нейронной». Чтобы получить вектор длины dictLen (это требуется для того, чтобы иметь значение для всех токенов из словаря), матрицу (3×400) надо умножить на матрицу размерности $(400 \times dictLen)$. Теперь, когда мы имеем значение для каждого слова, для получения вероятностного распределения используется функция Softmax. Данная функция, также называемая мягким максимумом, является обобщением логистической функции в многомерном случае: для каждой координаты вектора v результат применения Softmax равен: $softmax(v) = \frac{1}{\sum_{k=1}^{K} e^{v_k}} \begin{pmatrix} e^{v_1} \\ e^{v_2} \\ \dots \\ e^{v_K} \end{pmatrix}$. Она является дифференциируемой и позволяет получить вероятностное распределение для вектора. Соответственно, последний слой сети имеет размерность $(400 \times dictLen)$. Матрица весов на данном слое называется матрицей декодирования (decoding matrix).

3.5 Подготовка к тонкой настройке (fine-tuning)

3.5.1 Процесс сопоставления для матрицы эмбеддингов

При рассмотрении архитектуры сети для предобучения мы имели словарь длины $dictLen$. Однако в наборе данных, для которого мы хо-

тим научиться предсказывать тональность, есть слова, не присутствовавшие в наборе данных для предобучения, и наоборот, отсутствует часть слов, присутствовавших в исходном наборе данных. Соответственно, новая длина словаря равна $newDictLen$, вследствие чего требуется изменить матрицы эмбеддингов и декодирования. В обеих матрицах строки, соответствующие словам, отсутствующих в целевом наборе данных, удаляются; для новых токенов, (тех, которые отсутствуют в наборе данных для предобучения) строки в обеих матрицах инициализируются векторами из средних значений для каждого столбца, которые, в свою очередь, близки к нулю (Рис. 1.2).

3.5.2 Метод обратного распространения ошибки во времени переменной длины

Для обучения рекуррентных нейронных сетей используется метод обратного распространения ошибки во времени (backpropagation through time) [*A Gentle Introduction to Backpropagation Through Time*, 2017]. Рассмотрим отличие данного метода от классического метода распространения ошибки. Все объекты тренировочной выборки (тексты в нашем случае) объединяются, формируется один длинный текст. Огромный размер этого текста делает невозможным применение обратного распространения ошибки с точки зрения объема памяти. Вместо этого текст разбивается на несколько выборок с фиксированным размером – в данном случае на 64 выборки. Как показано на рисунке 1.3, мини-выборки создаются путем обрезания каждой из выборок и объединения каждого столбца – в данном случае, выборки обрезаются на части из 70 токенов. Во время обучения на каждой итерации на вход подается одна мини-выборка. Иными словами, 64 текста длиной в 5 (k в общем случае) токенов обрабатываются параллельно: для каждой из 64 последовательностей предсказывается следующий токен, считается функция потерь и обновляются

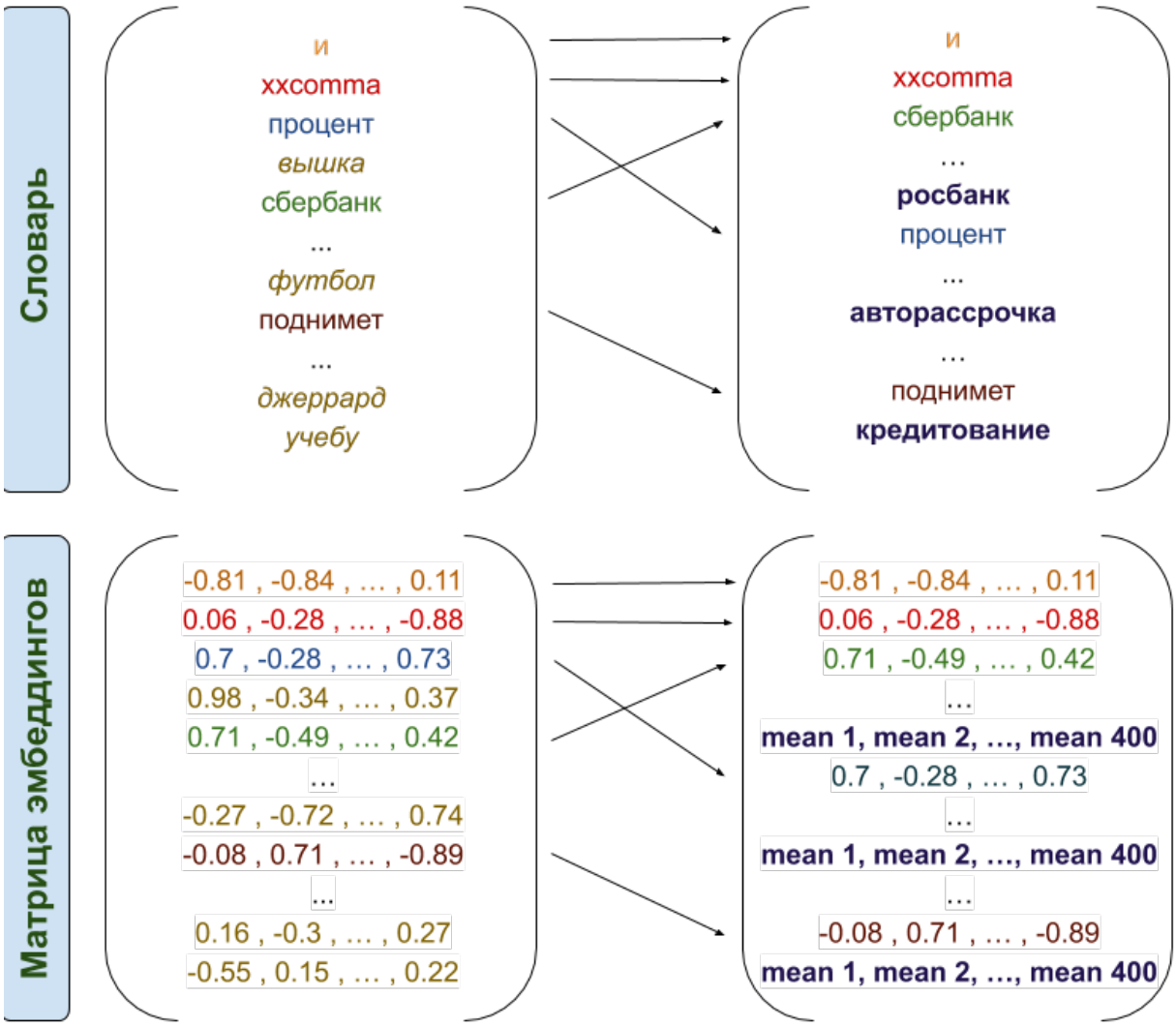

Рис. 1.2: Процесс сопоставления для матрицы эмбеддингов.

веса в соответствии с алгоритмом обратного распространения ошибки. После этого начинается новая эпоха, в которой обрабатываются следующие 64 последовательности и так далее.

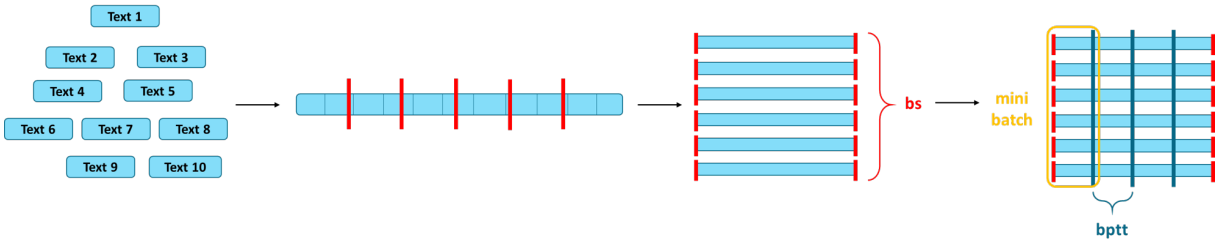

Рис. 1.3: Процесс создания мини-выборок [Howard, 2018].

Merity et al. [Merity, Keskar, Socher, 2017] расширили данную тех-

нику, введя обратное распространение ошибки во времени переменной длины. Основной недостаток предыдущего метода заключался в том, что спустя определенное число эпох модель учится на абсолютно одинаковых мини-выборках. Обратное распространение ошибки во времени переменной длины решает эту проблему путем добавления определенной степени случайности. Во время усечения каждой из 64 выборок (до 5 токенов в данном случае) и превращения их в мини-выборки, в 95% случаев параметр длины усечения останется прежним (5 в данном случае), а в оставшихся 5% выбирается случайно из нормального распределения. Такая модификация позволяет алгоритму учиться на разных последовательностях и, как следствие, более эффективно использовать имеющиеся данные.

В ULMFiT также присутствует паддинг (padding): ко всем текстам с длиной, меньшей значения усечения, применяется паддинг -– добавление в конец текста токенов паддинга. Это применяется для того, чтобы все тексты, которые проходят через алгоритм обратного распространения ошибки, были одинаковой длины.

3.5.3 Adam оптимизатор

В качестве метода минимизации функции потерь при обучении в ULMFiT используется алгоритм Adam (adaptive moment estimation). Howard и Ruder [Howard, Ruder, 2018] отмечают, что данный метод, на их взгляд, является наилучшим в контексте задач NLP при условии корректного выбора гиперпараметров. Adam является расширением стохастического градиентного спуска (SGD) как метода адаптивного обучения. Он оценивает первый и второй моменты градиента, чтобы приспособить значение скорости обучения для каждого параметра нейронной сети. Моменты оцениваются скользящими средними градиента и квадрата градиента с поправкой на определенное отклонение: $m_t =$

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

Идея градиентного спуска часто изображается с помощью мячика, скатывающегося по склону, который в конечном итоге должен оказаться в самой низкой впадине (то есть в глобальном минимуме). Используя данную аналогию, идею Adam можно описать с помощью скатывающегося тяжелого мяча с трением [Ruder, 2017]. Как показано на рисунке (Рис. 1.4), данный алгоритм позволяет преодолеть локальные минимумы и остановиться возле глобального минимума в силу своей массы.

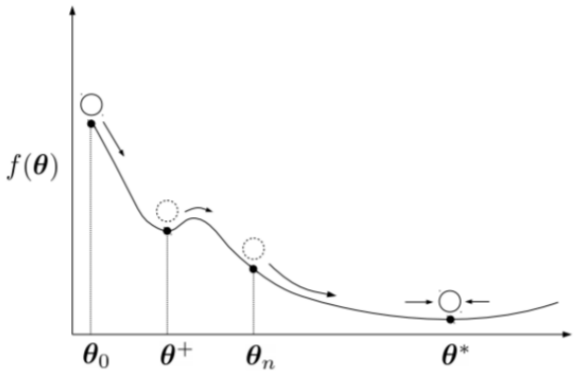

Рис. 1.4: Принцип действия Adam оптимизатора [Heusel (и др.), 2018].

Как уже было отмечено, Howard и Ruder [Howard, Ruder, 2018] предложили специфические значения для гиперпараметров β_1 и β_2 алгоритма. В частности, вместо стандартного значения 0.9 для β_1, авторы используют значение между 0.7 и 0.8.

3.5.4 Дропаут

Ключевая идея архитектуры AWD-LSTM состоит в использовании различных модификаций дропаута в разных местах модели. В процессе обучения нейронной сети и обновления весов для минимизации функции потерь часть весов синапсов может изменяться так, что они исправляют ошибки весов предыдущих синапсов. Это приводит к сложным взаимозависимостям, которые негативно влияют на обобщающую способность модели и, следовательно, приводят к переобучению. Одним из методов

борьбы с переобучением является построение большого числа нейронных сетей, обученных на том же самом наборе данных и усреднение их результатов. Поскольку данный метод, как правило, неосуществим в реальности, для борьбы с переобучением используется дропаут.

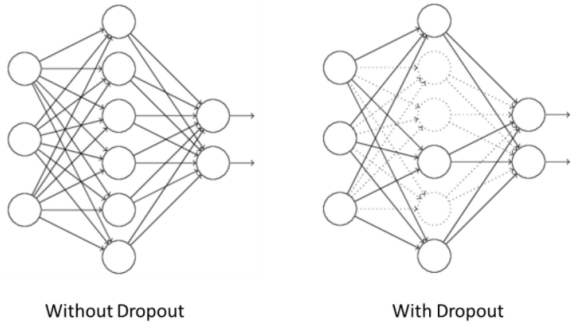

Рис. 1.5: Архитектура нейронной сети без дропаута / с дропаутом [Nielsen, 2018].

Дропаут с вероятностью p – это метод регуляризации, заключающийся в исключении каждого нейрона сети с вероятностью p на каждой итерации обучения (Рис. 1.5). После завершения итерации удаленные нейроны восстанавливаются, новые нейроны исключаются и так далее. Эта процедура происходит на протяжении всего обучения. Важно отметить, что поскольку полученные веса корректировались исходя из наличия $1 - p$ доли нейронов от их общего числа, в конце обучения веса делятся на величину $1 - p$. Вследствие того, что в данном методе веса синапсов не могут полагаться на веса предыдущих синапсов (из-за их периодического отсутствия), нейронная сеть менее склонна к переобучению.

В архитектуре AWD-LSTM авторы предложили 5 различных методов дропаута [Merity, Keskar, Socher, 2017]. Howard и Ruder [Howard, Ruder, 2018], в свою очередь, подобрали оптимальные значения дропаута для каждой из используемых модификаций. В этой связи при использовании модели требуется лишь указать значение коэффициента масштабирования – все используемые в модели значения дропаутов будут умножены на данную величину.

3.6 Тонкая настройка языковой модели на целевом наборе данных

После того, как все необходимые условия были рассмотрены, можно переходить к тонкой настройке. Несмотря на то, что Howard и Ruder [Howard, Ruder, 2018] не были первыми, кто использовал индуктивный перенос знаний с помощью тонкой настройки в NLP, они были одними из первых, кто успешно применил данную технику без последствий переобучения.

В предыдущих попытках процесс тонкой настройки приводил к практически полной потери знаний, полученных при решении исходной задачи (source task). Авторы исправили данные недостатки, введя новые специфичные для NLP методы тонкой настройки. Поскольку во многом благодаря данным техникам ULMFiT стала SOTA моделью в 2018 году, эти методы будут рассмотрены в работе.

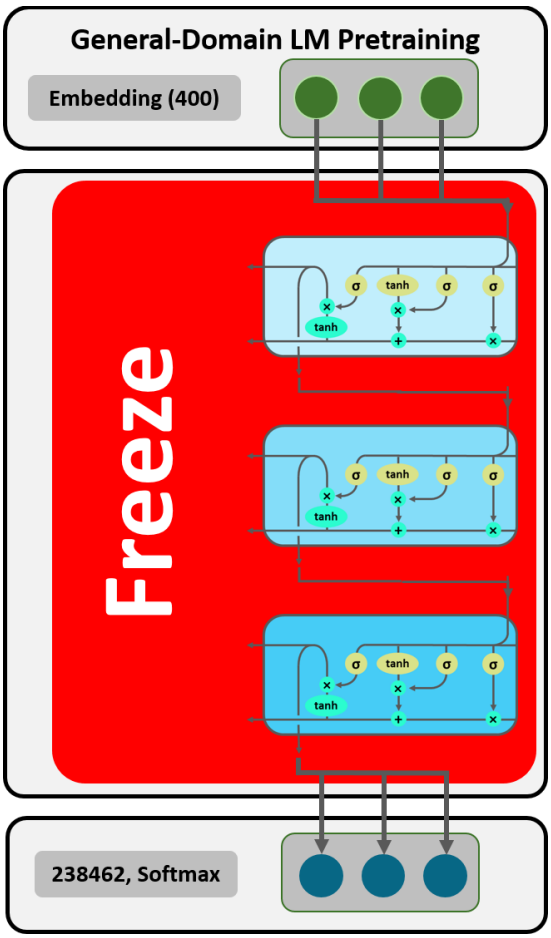

Рис. 1.6: Замораживание (freezing) [Faltl, Schimpke, Hackober, 2019].

3.6.1 Замораживание (freezing)

Вследствие трансформации матрицы эмбеддингов (субсекция 3.5.1), часть эмбеддинговой и декодинговой матриц состоит из случайным образом инициализированных эмбеддингов (для новых токенов).

Если начать обучать модель на целевом наборе данных, существует большой риск серьезной потери накопленных знаний во всех трех клетках LSTM. В этой связи на первом шаге веса всех слоев LSTM «замораживаются» и одну эпоху (один проход по всем объектам тренировочной выборки) обучаются только два других слоя, эмбеддинговая и деконинговая матрицы. Во время этой эпохи корректируются параметры, соответствующих новым словам. Данный метод называется замораживание (freezing) и был введен в работе Felbo et al. [Felbo (и др.), 2017]. На второй эпохе все слои размораживаются и вся архитектура обучается целиком. Процедура замораживания изображена на рисунке 1.6.

3.6.2 Корректировка шага обучения

В ULMFiT используется корректировка шага обучения (Рис. 1.7). Это означает, что величина параметра меняется в процессе обучения. Обучение начинается с маленького значения, которое очень быстро возрастает, а затем, достигнув максимального значения, постепенно уменьшается. Резкий рост величины обеспечивает быструю сходимость к приблизительной области в пространстве весов, а долгое затухание – точную подстройку весов. Данный подход к корректировке шага обучения называется наклонным треугольным шагом обучения (slanted triangular learning rate). Впевые метод был введен в работе Smith et al [Smith, 2017].

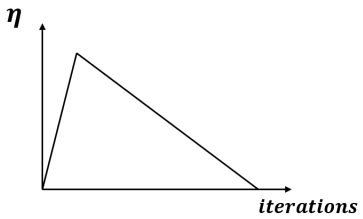

Рис. 1.7: Наклонный треугольный шаг обучения [Faltl, Schimpke, Hackober, 2019].

3.6.3 Дифференциальная тонкая настройка (discriminative fine-tuning)

Как уже отмечалось, AWD-LSTM содержит три связанные LSTM клетки. Yosinski et al. [Yosinski (и др.), 2014] обнаружили, что подобная многоуровневая структура демонстрирует особенно хорошие результаты при работе со сложными данными, такими как текст или изображения, ввиду того, она способна собирать различные категории информации на каждом уровне, начиная с общей информации на первом слое и накапливая все более специфичную информацию на каждом последующем уровне. В контексте текстовых данных первый слой многоуровневой архитектуры может запоминать базовую структуру предложений, а следующие слои – более частную информацию, например, временную зависимость.

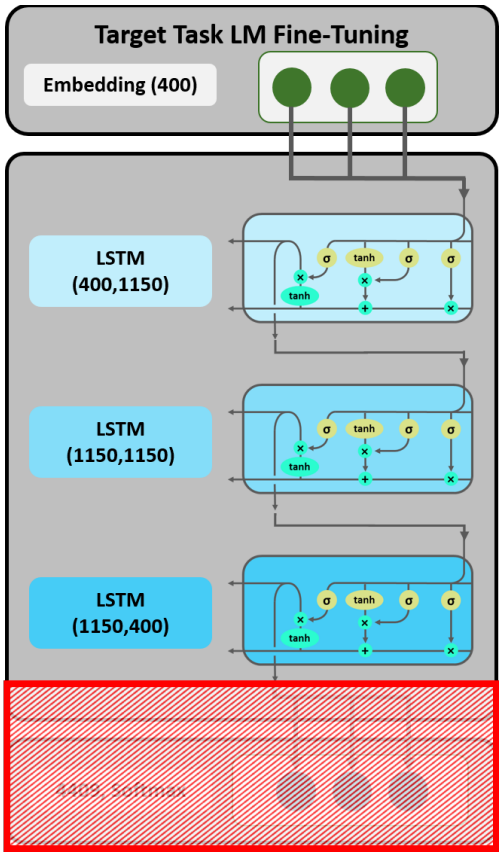

Рис. 1.8: Удаление последнего слоя (cut-off) [Faltl, Schimpke, Hackober, 2019].

Для того, чтобы слои могли запоминать различные типы информации, для обучения весов разных слоев используются разные значения шага обучения. Обычно параметры модели θ обновляются в процессе обучения с фиксированным значением шага обучения η [Ruder, 2017]: $\theta_t = \theta_{t-1} - \eta \nabla_\theta J(\theta)$.

В дифференциальной тонкой настройке параметры каждого слоя име-

ют собственное значение шага обучения η^l [Ruder, 2017]: $\theta_t^l = \theta_{t-1}^l - \eta^l \nabla_{\theta^l} J(\theta^l)$ для каждого слоя l.

Начиная с небольшой величины для первой LSTM, значения шага обучения увеличивается с глубиной модели вследствие растущего объема приобретенной моделью информации и сложности зависимостей. После обучения на протяжении нескольких эпох модель выучивает специфику языка целевого набора данных. Однако для того, чтобы модель в конечном итоге предсказывала тональность текста, ее архитектура должна быть изменена, и в то же время полученные знания должны быть сохранены. Как показано на рисунке 1.8, первые четыре слоя (матрица эмбеддингов и AWD-LSTM) сохраняются, в то время как последний слой, матрица декодирования, удаляется.

3.7 Целевой классификатор

На последней стадии обучения ULMFiT настраивается целевой классификатор. Как уже отмечалось, архитектура классификатора отличается от архитектуры языковой модели в силу решения разных задач и, как следствие, разных выходных данных. Вместо последнего слоя к получившейся на предыдущем шаге модели добавляются два линейных слоя с функциями активации ReLU и Softmax соответственно. Активация Softmax в последнем слое используется для получения вероятностей принадлежности к каждому из классов.

При обучении на мини-выборках языковой модели объекты обрезаются на определенную длину, поэтому большая часть объектов содержит неполные фрагменты текстов. Обучение классификатора подобным образом невозможно, поскольку для каждого текста целевое значение предсказывается отдельно. В этой связи кажда мини-выборка состоит из одного целого твита. Следовательно, каждый твит после прохожде-

ния матрицы эмбеддингов представлен матрицей, содержащей эмбеддинговые векторы для каждого токена. Так, аналогично языковой модели, на выходе из последнего слоя LSTM получается матрица длины, равной длине текста, и ширины, равной размерности эмбеддингов -– 400.

Важно отметить, что поскольку модель обучается при помощи мини-выборок, ко всем текстам, меньшим самого длинного в мини-выборке, применяется паддинг, чтобы все тексты в мини-выборках были одинаковой длины. Соответственно, сортировка документов по их длине снижает как объем паддинга, добавляющего шум, так и время вычисления.

3.7.1 Объединение пулингов (concat pooling)

Как правило, часть слов в тексте содержит больше релевантной информации для классификации, чем другие. В то же время на каждом временном шаге (при обработке каждого слова в AWD-LSTM) скрытое состояние последнего слоя обновляется, и часть важной информации может потеряться, если в следующий слой попадает только последнее скрытое состояние. Объединение пулингов -– метод, введенный Howard и Ruder [Howard, Ruder, 2018], решает данную проблему путем использования информации всей выходной матрицы: вектор последнего скрытого состояния, имеющий размерность 400, объединяется с максимальным и средним значениям пулинга по скрытым

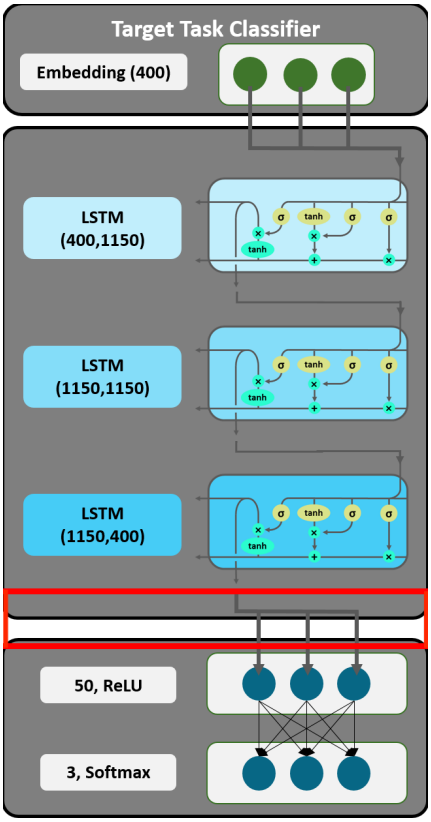

Рис. 1.9: Объединение пулингов (concat pooling) [Faltl, Schimpke, Hackober, 2019].

состояниям последней LSTM клетки (с max-pooling-ом и mean-pooling-ом выходной матрицы). Максимальный пулинг скрытых состояний -– вектор максимальных значений по каждому из 400 столбцов всех скрытых состояний для данного текста. Средний пулинг – вектор средних значений столбцов. Как показано на рисунке 1.9, результатом объединения пулингов является вектор длины 1200. Данный вектор подается на вход в следующий декодирующий слой (decoder).

3.7.2 Линейный декодер (linear decoder)

Линейный декодер является обычной двухслойной нейронной сетью прямого распространения с двумя функциями активации – ReLU и Softmax (Рис. 1.10). Первый слой имеет 50 скрытых нейронов, следовательно, на первом слое матрица весов имеет размерность 1200×50. Число нейронов во втором слое совпадает с числом различных классов, чтобы после применения Softmax i-й элемент получившегося вектора отражал вероятность принадлежности к классу i. На последнем шаге каждой эпохи обучения считается значение функции потерь и кросс-энтропии и применяется оптимизационный метод Adam.

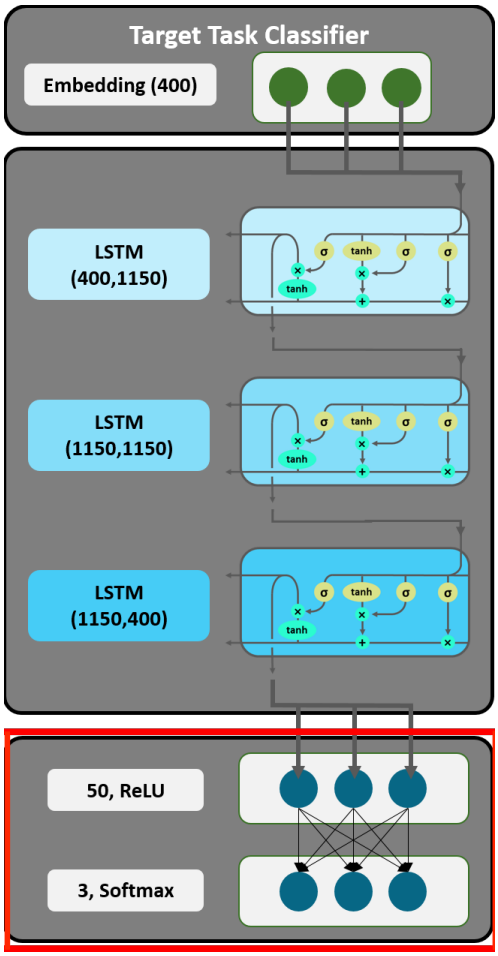

Рис. 1.10: Линейный декодер (linear decoder) [Faltl, Schimpke, Hackober, 2019].

3.7.3 Постепенно размораживание (gradual unfreezing)

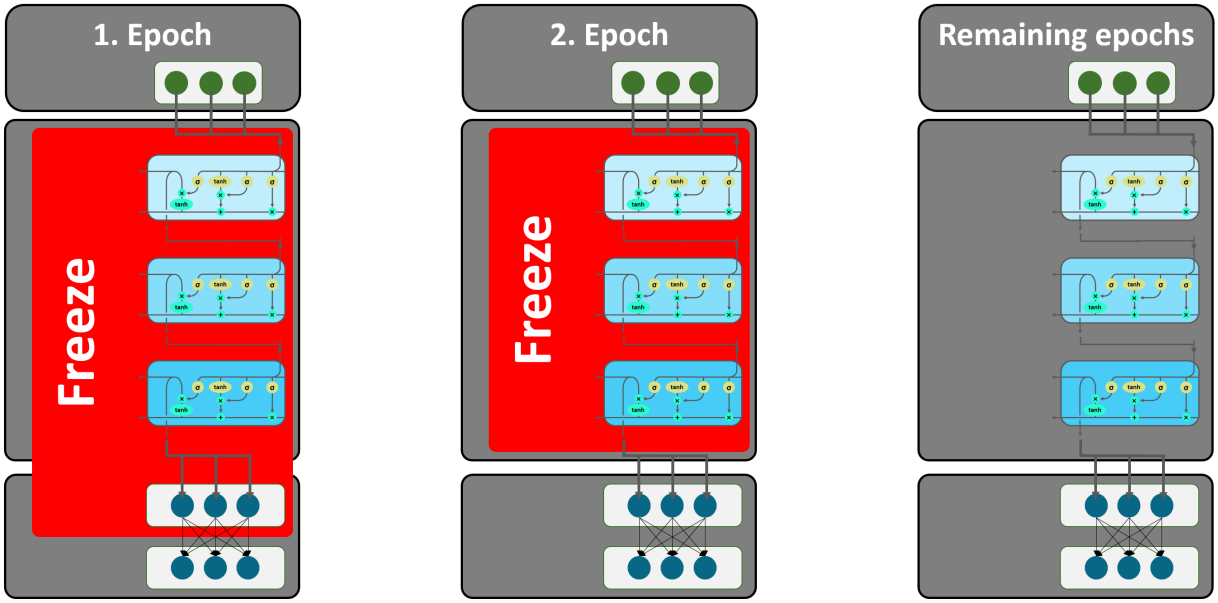

Рис. 1.11: Постепенно размораживание (gradual unfreezing) [Faltl, Schimpke, Hackober, 2019].

Поскольку размерность нового декодера отличается от размерности декодера языковой модели, его веса инициализируются случайными значениями. В этой связи при обучении новой архитектуры аналогично субсекции 3.6.1. существует большой риск серьезной потери информации из предыдущих слоев, вследствие чего вновь применяется техника замораживания слоев. Как показано на рисунке 1.11, на первой эпохе обучения все слои кроме последнего (второго слоя декодера) замораживаются, тем самым корректируются только веса выходного слоя. На второй эпохе размораживается предпоследний слой, первый слой декодера, что позволяет всему декодеру настроиться под перенесенные из языковой модели веса. Начиная с третьей эпохи модель размораживается целиком и все параметры корректируются в соответствии с функцией потерь. В результате обучения создается классификатор ULMFiT – модель, способная предсказывать вероятность принадлежности к каждому классу для корпуса текстов.

4 Сбор данных

Глава 4 посвящена сбору данных. В ней содержится описание методов сбора данных для предобучения модели, данных для ее дальнейшего обучения и данных Твиттера, и методов их дальнейшей предобработки, а такжже данных по ценам акций. Описание сбора данных проводится в связи с тем, что стандартные методы извлечения (например, создание sql таблицы для корпуса твитов или использование Twitter API для получения данных Твиттера) требуют большого числа ресурсов или не работают в принципе: так, Twitter API не позволяет бесплатно получить доступ к твитам более, чем недельной давности.

4.1 Котировки акций Сбербанка, Росбанка и ВТБ

Данные по котировкам акций банков взяты с финансового портала *finam.ru* [*Finam*, URL]. В работе используются данные с января 2015 года по март 2020. Это связано с тем, что в 2014 году ввиду нестабильной политической обстановки и введения многочисленных санкций в отношении России и, в частности, российских банков, цены акций были очень волатильными, и оценка влияния данных Твиттера в таких условиях может быть зашумленной.

В работе исследуется зависимость динамики цен акций банков от тональности данных Твиттера – таким образом оценивается способность общественного мнения обуславливать скачки котировок акций. В этой связи корректно предсказывать первую цену акций за день, так как используются данные тональности за предыдущие дни. Для этого данные скачиваются с периодичностью в 1 минуту, чтобы брать самую первую цену акций за день.

4.2 Предобработка текстов

Метод предобработки данных существенно отличается от большинства современных подходов к предобработке текстовых данных. Так, знаки препинания не удаляются, а заменяются специальными токенами (*xxcomma* для запятых, *xxdot* для точки и так далее). Это связано с тем, что знаки препинания обладают определенной смысловой нагрузкой, что позволяет модели улавливать дополнительные зависимости. Все английские слова заменяются на токен «xxeng». Аккаунты, хэштеги и ссылки заменяются на токены «xxaccnt», «xxhshtg», и «xxurl». Числа также заменяются токеном «xxnum». Если в твите содержатся несколько предложений, они разделяются токеном «xxnewsent». Дублирующиеся последовательности для каждого специального токена (начинающегося с «xx») заменяются на название токена и токен «xxmult», указывающий на повтор. Подробнее метод предобработки описан в приложении №1.

4.3 Данные Твиттера

Для импорта данных Твиттера используется библиотека GetOldTweets3 вместо официального Twitter API. Это связано с тем, что Twitter API позволяет скачивать только твиты за последнюю неделю в рамках бесплатного доступа. Библиотека GetOldTweets3 доступна через менеджер пакетов для Python *pip*. Способ получения данных с помощью GetOldTweets3 указан в приложении №2.

В работе используются данные Твиттера, содержащие одно из следующих ключевых слов: «сбербанк» и «сбер» для Сбербанка, «втб» и «внешторгбанк» для ВТБ и «росбанк» для Росбанка; регистр слов не имеет значения. Важно отметить, что наличие зашумленного объекта гораздо хуже, чем удаление корректного, так как число записей за день для каждого банка достаточно большое (больше пятидесяти в 95% слу-

чаев). В этой связи полученные твиты проходят через несколько этапов фильтрации.

Больша́я часть скачанных данных является нерелевантной. Так, около 40% твитов со словом «сбер» используют это слово для обозначения банка карты, на которую требуется оформление перевода (“Я прошу вашей помощи, хоть 5-10 рублей киньте мне на Сбер(89173015817)”). Аналогично для ВТБ 12% твитов относятся к баскетбольной «Единой лиге ВТБ». Поскольку такие записи существенно зашумляют результаты тональности, все записи содержащие список определенных слов или последовательности из минимум четырех цифр (для определения твитов с номерами карт или телефонов) удаляются.

Далее удаляются твиты, длина которых после предобработки меньше 3 токенов. Например, запись «Сбербанк онлайн» будет удалена, а «Сбербанк -– лучшие» оставлена, так как тире заменяется на токен «xxdash». Это связано с тем, что короткие твиты часто не несут смысловой нагрузки и также добавляют шум в результаты.

На следующем шаге удаляются твиты, которые написаны официальными твиттер аккаунтами банков или предназначены им. Тональность записей представителей банка, очевидно, не отражает общественного мнения по отношению к нему. В то же время во время переписок с официальными аккаунтами банков тональность сообщения клиента часто не связана с его мнением насчет банка.

Последним этапом обработки твитов является удаление всех записей, в которых присутствует название более, чем одного банка из списка десяти самых популярных банков в России: Сбербанк, ВТБ, Тинькофф, Газпромбанк, Альфа-Банк, Росбанк, Рокетбанк, Россельхозбанк, Открытие и Райффайзенбанк (со всеми возможными производными названий, например, «альфа» для Альфа-Банка и райфайзен для Райффайзена). Это происходит потому, что твиты, относящиеся к нескольким банкам,

часто содержат положительную информацию об одном и негативную о другом банке: «втб до сбера как до луны».

Все этапы предобработки содержатся в приложении №3.

4.4 Получение данных для предобучения модели

Поскольку конечной целью является получение предсказаний тональности на данных Твиттера, модель необходимо предобучить на корпусе данных, максимально схожем с ними. Благодаря [Ю. Рубцова, 2014] в работе используется набор данных с 17 миллионами записей Твиттера. Поскольку извлечение данных является нетривиальным, оно описано в приложении №4 После предобработки данные насчитывают порядка 15 миллионов записей со средней длиной твита в 10.6 токенов.

4.5 Данные для обучения модели

Для того, чтобы обучить модель распознавать тональность твитов, требуется размеченный набор данных. Поскольку разметка данных -– ресурсозатратная процедура, а размеченного целевого набора данных, не существует, в работе используются три набора данных для обучения модели. В результате обучаются три модели с различными значениями параметров, которые по-разному оценивают тональность твитов. Далее каждый из наборов данных рассматривается более детально.

4.5.1 Общие данные Твиттера

Совместно с набором данных с 17 миллионами записей твиттера Рубцова Ю. [*Rubtsova*, URL] создала выборку с 230 тысячами записей и значением тональности каждой. Специфика данных и их объем являются большим преимуществом данного набора данных, однако способ его разметки является недостатком. Согласно работе [Ю. Рубцова, 2014], раз-

метка проводилась при поммощи свода правил, основанного на словах-триггерах, отражающих позитивные или негативные эмоции. В этой связи даже самая простая модель (а тем более ULMFiT) после обучения предсказывает значения на отложенной выборке с очень высокой точностью ($> 99\%$ даже для tf-idf): модель просто запоминает эти слова-тригеры и ориентируется на них при предсказании. Однако тональность твитов в отношении банков не всегда полностью определяется словами-тригерами: например, многие твиты являются саркастическими («вот это сбербанк молодцы, памятник им за такое надо») или содержат различные эмоции: «это конечно прекрасно, но почему мне до сих пор не вернули деньги??».

4.5.2 Данные Твиттера относительно банков

Следующий используемый набор данных также принадлежит Рубцовой Ю. [Лукашевич, Ю. В. Рубцова, 2016], однако он выпущен двумя годами позже и размечен с помощью более современных методов. Этот набор данных содержит комментарии относительно российских банков и их тональность. Тот факт, что специфика данных совпадает со спецификой целевого набора данных, является бесспорным преимуществом данного набора данных. Однако данный корпус твитов обладает и рядом недостатков. Во-первых, недостатком снова является проблема разметки: тональность комментария не всегда может быть определена словами-триггерами. Например, запись «обалдеть че творится около сбербанка полиция с собаками» не имеет негативного оттенка по отношению к Сбербанку, однако ввиду присутствия слов «обалдеть», «че» и «собаками» данный твит принадлежит к отрицательному классу в наборе данных. Другим существенным недостатком данного набора данных является его маленький объем: он насчитывает чуть более 10 тысяч уникальных записей, из которых около $2/3$ дублируют друг друга («сша ввели санкции

против сбербанка» и «сбербанк попал под санкции сша»). В этой связи число уникальных твитов в данном корпусе чуть больше трех тысяч, что является скромным объемом для обучения такой модели, как ULMFiT.

4.5.3 Новостные данные

Последний из рассматриваемых наборов данных взят с платформы Kaggle [*Kaggle*, 2018]. Он содержит новости, взятые с платформы lenta.ru с размеченной тональностью: положительные, отрицательные или нейтральные. Данный набор данных содержит совсем небольшое число записей (менее 5 тысяч) и существенно отличается по специфике языка от данных Твиттера, однако в отличие от других наборов данных он позволяет анализировать тональность новостей, которые часто встречаются в Твиттере в контексте банков.

5 Создание признаков

В главе представлено предобучение выбранной модели, генерация предсказаний тональности твитов и формирование признаков для моделей предсказания цен акций. Также рассматриваются различные аспекты обучения, такие как метод разбиения выборок или учет инфляции.

5.1 Предобучение модели и предсказания тональности

При помощи сервиса Google Colab [*Google Colab*, URL] модель обучается на протяжении 4 эпох с различными значениями шага обучения для разных слоев приложение №5.1. Итоговая модель способна анализировать и воспроизводить речь разговорного русского языка. Далее происходит обучение языковых моделей на трех используемых наборах данных

приложение №5.2. После того, как каждая из моделей выучила специфику соответствующего корпуса текстов, на каждом из имеющихся наборов данных обучаются итоговые классификаторы. Для оценивания качества моделей используются кросс-энтропия и точность (Accuracy).

На следующем шаге каждая из моделей предсказывает вероятность принадлежности к каждому классу (в модели общих данных Твиттера два класса, «положительный» и «отрицательный», в двух других добавляется класс «нейтральный») и само значение класса на всех полученных данных Твиттера, начиная с 01.01.2015. Классам «отрицательный», «нейтральный» и «позитивный» ставятся в соответствие числа -1, 0 и 1. Поскольку данные предсказания будут использоваться в качестве признаков для прогнозирования динамики цен акций, очевидно, что вероятность принадлежности к одному из имеющихся классов в каждой модели можно удалить, так как она равна $p_i = 1 - \sum_{j \neq i}^{k}$, где последний член является суммой вероятностей принадлежности к остальным классам, что означает ее линейную зависимость с другими вероятностями. Из модели с двумя классами удаляется отрицательный класс, а из двух других – нейтральный, так как он является наименее репрезентативным. Таким образом генерируются 8 признаков для каждого твита.

Поскольку зависимость между динамикой цен акций и тональностью данных Твиттера не обязательно является линейной, к признакам применяются четыре различные нелинейные функции. Ввиду того, что значения имеющихся признаков находятся в диапазоне $[0; 1]$ для признаков вероятности и в диапазоне $[-1; 1]$ для признаков принадлежности классам, результаты классических функций активации в нейронных сетях (сигмоида, гиперболический тангенс, ReLU, знаковая функция), используемых для добавления нелинейности, будут иметь очень большие значения корреляции (больше 0.99) с исходными признаками. В этой связи в работе используются следующие функции нелинейности: логарифмиче-

ская ($softln(x) = ln(1 + x)$), модификация гиперболического тангенса, в которой аргумент дополнительно возводится в куб, чтобы "сместить"распределение ближе к центру ($tanhcb(x) = \frac{e^{2x^3}-1}{e^{2x^3}+1}$), Leaky ReLU с параметром $\alpha = 0.1$ ($lrelu(x) = x \cdot I\{x > 0\} + 0.1x \cdot I\{x < 0\}$) и возведение в квадрат с сохранением исходного знака: $sq(x) = sign(x) \cdot x^2$. Однако если значения аргумента лежат в диапазоне $[0; 1]$, результаты некоторых функций также будут иметь высокую корреляцию с исходными величинами. Вследствие этого перед применением нелинейных функций к признакам вероятности их значения трансформируются в диапазон $[-1; 1]$ умножением на два и вычитанием единицы: $x_{new} = 2x_{old} - 1$.

Выбор нелинейных функций обуславливался несколькими факторами. Прежде всего минимизировалась корреляция исходного признака и результата от применения нелинейной функции. В случае высокой корреляции этих показателей (например, после применения сигмоиды корреляция с исходным признаком превышала 0.99) новый признак ничем не будет отличаться от исходного, ввиду чего использование нелинейной функции становится бессмысленным. Далее, функции должны сохранять монотонность: так как в случае наличия зависимости целевой переменной от признаков тональности эта зависимость должна иметь прямой характер, нарушение монотонности приведет к созданию зашумленных признаков. В частности, необходимость сохранения монотонности не позволяет использовать периодичные функции (например, различные тригонометрические функции). Наконец, четыре используемые функции имеют разные центры распределений, что позволяет использовать различные характеристики признаков.

После применения четырех нелинейных функций число признаков увеличивается в 5 раз. Ввиду того, что прогнозы строятся по первой цене за день, все результаты тональности за день усредняются. Таким образом формируется среднее значение для каждого из признаков за каждый

день в период с 01.01.2015 по 01.03.2020. К имеющимся 40 признакам добавляется еще один – число твитов за день.

5.2 Выбор моделей

В работе используются два класса моделей: линейная регрессия в качестве линейной модели и модификация DART экстремального градиентного бустинга в качестве нелинейной [Rashmi, Gilad-Bachrach, 2015]. Использование градиентного бустинга обусловлено несколькими факторами. Небольшие нейронные сети (1 - 2) слоя практически ничем не отличаются от линейных моделей, в частности, от линейных регрессий, а глубокие модели требуют большого числа наблюдений. Поскольку в данной задаче число наблюдений небольшое (менее 1300), глубокие нейронные сети будут склонны к переобучению даже при использовании дропаута и других техник. Выбор градиентного бустинга перед менее объемными tree-based моделями (одно решающее дерево или случайный лес), в свою очередь, обуславливается более точным способом оценивания признаков в модели градиентного бустинга. Два стандартных метода оценивания важности признаков в моделях, основанных на решающих деревьях – суммарное число вершин, использующих данный признак для разбиения и значение суммарного снижения функции потерь благодаря данному признаку. Первый метод имеет существенный недостаток: признаки, используемые для разбиения в нижних вершинах дерева, являются гораздо менее значимыми, чем признаки, используемые в начальных вершинах, однако при расчете соответствующей величины (суммарное число вершин, использующих для разбиения данный признак) оба случая вносят равносильный вклад. В этой связи в работе используется второй метод оценивания важности признаков. Градиентный бустинг в таком случае является оптимальной tree-based моделью: так как каждое новое

дерево снижает значение функции потерь для ансамбля всех предыдущих деревьев, каждое новое разбиение в градиентном бустинге «жадно» оптимизирует функцию потерь. В то же время при оценивании важности признаков в одном решающем дереве или в модели случайного леса большую роль играет случайность. Для одного дерева самым значимым признаком при используемом методе оценивания в большинстве случаев является признак, используемый в корневой вершине. Следовательно, если предикаты с использованием другим признаков могли бы снизить функцию потерь на схожую величину, немного меньшую получившейся, бóльшая часть этих признаков будет значительно менее важной признака, используемого в корневой вершине, поскольку предикаты с данными признаками в следующих вершинах не обязательно являются оптимальными после разбиения. В случайном лесе либо элемент случайности при построении нового дерева будет небольшим, что приведет к схожести построенных деревьев и проблеме, описанной для одного дерева, либо элемент случайности будет высоким, что приведет к искажению реальной важности признаков. Вследствие этого в работе используется градиентный бустинг в качестве нелинейной модели.

Модификация DART градиентного бустинга [Rashmi, Gilad-Bachrach, 2015] позволяет существенно снизить переобучение, что является очень важным в данном случае, так как число наблюдений для каждого банка не превышает 1300. Более того, алгоритм обеспечивает масштабирование прогнозов каждого дерева, обеспечивая сглаживание оценивания важности признаков. Так, для проблемы, описанной для одного решающего дерева, признаки, которые могли бы снизить функцию потерь на схожую величину, с большой долей вероятности будут использоваться в качестве признаков для разбиения в корневых вершинах следующих деревьев, а масштабирование означает, что вес этих дереьев и, как следствие, важность признаков будут примерно одинаковыми.

5.3 Создание признаков для моделей предсказания цен акций

Для проведения различных тестов, позволяющих проанализировать наличие статистической взаимосвязь признаков тональности данных Твиттера и динамики акций необходимо обучить две регрессии и модель DART. Первая регрессия использует множество признаков, создание которых будет описано в данной секции, а вторая -– все те же плюс признаки тональности. Данный способ оценивания зависимости используется в связи с тем, что признаки тональности могут содержать часть информации, которая уже содержится в других признаках, и в этом случае он позволит корректно оценить «чистое» влияние новых признаков, а методы, используемые в ряде предыдущих исследований [Kirlić (и др.), 2018; Sasank (и др.), 2016], где сравнивались константная модель и модель, основанная на признаках тональности, оценили бы зависимость некорректно, так как учитывали бы и часть информации, которая в любом случае присутствует при прогнозировании динамики акций.

Модель градиентного бустинга использует те же признаки, что и вторая регрессия, за исключением того, что признаки, получившиеся в результате дамми-кодирования, добавляются в модель в исходном виде. DART не требует дамми-кодирования ввиду того, что в модели все значения каждого признака в любом случае разбиваются на k групп (в большинстве реализаций, в XGBoost, CatBoost и LightGBM по умолчанию $k = 255$). Соответственно, порядковые (упорядоченные категориальные) признаки модель обрабатывает так же, как и обычные вещественные.

Первая группа признаков – временные показатели: квартал, месяц, день недели, день. Важно понимать, что несмотря на то, что эти признаки, на первый взгляд, являются количественные, они относятся к порядковым признакам, а не вещественным. В этой связи для их использова-

ния в регрессионных моделях необходимо применить дамми-кодирование (one-hot кодирование), так как признаки являются категориальными: например, эффект для четвертого дня недели, четверга, не всегда будет в два раза больше эффекта для второго дня недели, вторника. Поскольку в качестве предсказательной модели будет использоваться линейная регрессия, при дамми-кодировании переменной с k различными значениями добавляться будут только $k-1$ признаков, так как значение k-го будет линейно-зависимым с остальными.

С другой стороны, поскольку признаки являются порядковыми, информация об их порядке также может быть важной. Следовательно, исходные признаки также будут использоваться, так как их линейные комбинации могут порождать новые существенные признаки. Например, 30 х номер месяца + число дней приблизительно равно числу дней с начала года.

Вторая группа признаков – предыдущие значения и их производные. Как уже отмечалось, предсказываются первые значения котировок акций за день, поэтому добавляются k предыдущих первых и последних значений за день и их квадраты. В работе используется $k = 2$, данный параметр был получен эмпирическим путем. Далее берутся показатели, оценивающие разброс значений за день: разница между максимальным и минимальным значением и дисперсия минутных значений за каждый из предыдущих $k = 2$ дней. Использование данных признаков обусловлено тем, что сильное дневное колебание цены может означать, что часть инвесторов обладает некой инсайдерской информацией относительно банка, что приводит к резкому скачку его акций.

К используемым признакам в регрессионных моделях также добавляется единичный. Процесс создания признаков описан в приложении №6.

5.4 Стандартизация признаков

Важным аспектом формирования признаков для регрессионных является их масштабирование. Вследствие того, что многие из регрессоров имеют разные масштабы измерений – цены акций измеряются сотнями (в случае Сбербанка и Росбанка) или десятыми долями (ВТБ), а бóльшая часть временных признаков - единицами, признаки, используемые в регрессиях, должны масштабироваться. Две наиболее популярных процедуры масштабирования – стандартизация и нормализация. Стандартизация предполагает приведение значений признака к распределению с нулевым матожиданием и единичной дисперсии. Для этого из значений признака вычитается его математическое ожидание, и полученный результат делится на его стандартное отклонение. Нормализация означает приведение значений признака к фиксированному диапазону – как правило, $[0; 1]$. Это достигается путем вычитания минимального значения признака и деления на разницу между максимальным и минимальным значениями. Таким образом, минимальному значению соответствует нуль, а максимальному – единица. В работе используется стандартизация как для матрицы признаков, используемых в регрессионных моделях, так и для вектора целевой переменной, поскольку она не накладывает ограничений на диапазон значений.

5.5 Разбиение выборок

Длина вектора целевой переменной составляет 1294 наблюдения для каждого банка, что является скромной по размеру выборкой. Недостаточный объем обучающей выборки приводит к недообучению модели и, как следствие, к смещенным значениям оценок коэффициентов и плохому качеству прогнозов. В свою очередь маленький размер отложенной выборки подразумевает смещенные значения метрик качества, так как

выборка не отражает генеральную совокупность. В этой связи для повышения робастности оценок коэффициентов регрессий и значений метрик в работе используется последовательная кросс-валидация, предложенная в работе Mittal et al. [Mittal, A.Goel, 2011]. Использование классической кросс-валидации некорректно ввиду того, что отложенную выборку должны формировать элементы, идущие строго после элементов обучающей выборки, что нарушается в данном методе. Процедура k-фолдовой последовательной кросс-валидации проводится следующим образом. Выборка делится на $k+1$ равных частей (фолдов) по порядку, и формируются k обучающих и отложенных выборок так, что в i-ю тренировочную выборку попадают фолды от первого до $k+1-i$-го включительно, а i-ю тестовую формирует $k+2-i$-й фолд. Поскольку при большом значении k обучающие выборки, полученные на последних итерациях, могут быть слишком маленькими, несколько последних разбиений на обучающую и отложенную выборки удаляются. Далее обучаются k (или меньше, если часть выборок была удалена) моделей (в данном случае линейных регрессий), которые обучаются и тестируются на соответствующих выборках. Для получения итоговых оценок коэффициентов и метрик качества модели эти показатели считаются для каждой из моделей, и результаты усредняются.

В работе используется $k = 8$, то есть каждый фолд насчитывает 143 или 144 наблюдения (чуть меньше пяти месяцев); 6-й, 7-й и 8-й варианты разбиений удаляются, так как обучающая выборка в этих случаях слишком мала (меньше 500 наблюдений). Соответственно, обучаются и тестируются 5 разных моделей. Метриками качества и оценками коэффициентов регрессоров являются усредненные результаты этих моделей.

5.6 Метрики качества моделей

В качестве метрик качества используются средняя квадратичная ошибка (далее MSE – mean squared error) и средняя абсолютная ошибка в процентах (далее - MAPE – mean absolute percentage error). Выбор данных метрик обусловлен двумя факторами. Во-первых, это две наиболее часто используемые метрики в задачах прогнозирования цен акций [Moukalled, El-Hajj, Jaber, 2019; Vanukuru, 2018]. Во-вторых, MSE является функцией потерь классической линейной регрессии. Следовательно, минимизируя функцию потерь, модель оптимизирует целевую метрику, что упрощает задачу обучения. MAPE используется для того, чтобы иметь возможность оценивать не абсолютное, а относительное значение ошибки. Среднее абсолютное отклонение (MAE) не используется в связи с тем, что ее поведение, как правило, очень схоже с поведением MSE.

5.7 Учет инфляции

Целевыми переменными являются котировки акций банков за период, немного больший четырех лет. За этот период индекс потребительских цен, на основе которого рассчитывается инфляция, вырос на целых 32%. В этой связи цены акций должны быть скорректированы с учетом инфляции, так как единица измерения их цен (рубль) постоянно менялась. Месячные данные по инфляции в России взяты с официального сайта [*Таблица инфляции*, URL]. Полагается, что в течение месяца инфляция происходит равномерно: за любые n дней внутри одного месяца инфляция одинаковая. Соответственно, чтобы учесть инфляцию для наблюдения, произошедшего спустя M месяцев после 01.01.2015 в d-й день месяца, его значение x делится на кумулятивное произведение коэффициентов инфляции всех предыдущих месяцев, умноженное на значение инфляции за этот месяц $infl_{M+1}$, умноженное на отношение номера дня

к числу дней в данном месяце $days_{M+1}$: $x_{new} = \frac{x}{(\prod_{m=1}^{M} infl_m)} \cdot \frac{d \cdot infl_{M+1}}{days_{M+1}}$. Учет инфляции позволяет сделать более корректные выводы в исследовании, а также снизить случайную ошибку в модели.

6 Построение моделей и проведение тестов

В главе описаны методы создания моделей для проведения тестов на наличие зависимости, а также рассмотрены сами тесты.

6.1 Создание регрессии без признаков тональности

В предыдущей главе был описан способ формирования признаков для регрессионных моделей на основе временных показателей и предыдущих значений. В результате генерируются 65 признаков, на основании которых первая группа регрессионных моделей прогнозирует первую цену акций на следующий день. Однако многие из полученных признаков имеют зависимость, близкую к линейной: например, значения цены акций за предыдущие дни. Это приводит к тому, что линейная регрессия, обученная на данных признаках, катастрофически переобучается: значения некоторых признаков превышают 10^9. В этой связи перед обучением модели требуется произвести отбор признаков, который осуществляется с помощью лассо регрессий следующим образом. Методом золотого сечения [*The Publications and Writings of Jack Kiefer*, 1984] находится оптимальный с точки зрения MSE параметр регуляризации λ для лассо регрессии. В качестве новых признаков берутся те, вес которых в оптимальной лассо регрессии больше нуля, так как зануление признаков в лассо регрессии означает их линейную зависимость с другими.

Стоит отметить, что в работе используется именно лассо регрессия, а не гребневая (Ridge), так как в на данном этапе требуется именно проце-

дура отбора признаков, а не получения оптимального качества. Преимущества лассо регрессии перед гребневой для выявления наиболее важных признаков описаны в [Muthukrishnan, Rohini, 2016]. В свою очередь, при должном отборе признаков классическая линейная регрессия всегда превосходит в качестве как лассо, так и гребневую, так как не накладывает дополнительных ограничений на минимизацию функции потерь. Сравнение этих трех типов моделей проведено в [Rahman, Thevaraja, Gabriel, 2019], где авторы приходят к аналогичному выводу. В этой связи итоговой моделью без признаков является классическая линейная регрессия, обученная на новых признаках. Данный шаг описан в приложении №7.

6.2 Создание регрессии с признаками тональности

Вторая группа моделей использует полученные на предыдущем шаге регрессоры вместе с показателями тональности, сформированными в 5.1. Однако многие признаки тональности также будут иметь зависимость, близкую к линейной. Во-первых, все три предсказательные модели (ULMFiT) используют один и тот же набор данных для формирования предсказаний. Вследствие того, что модели предсказывают похожие показатели (тональность с разных сторон), результаты их предсказаний будут коррелировать друг с другом. Далее, добавленный признак «значение предсказанного класса» будет иметь зависимость с вероятностями, на основании которых он получен. В-третьих, для двух моделей с тремя классами даже после удаления вероятности принадлежности одному из классов два других будут связаны зависимостью: $p_1 + p_2 < 1$. Подобная зависимость, несмотря на нелинейность, также может привести к переобучению. Наконец, все используемые нелинейные функции сохраняют монотонность. Следовательно, их результаты будут также сильно коррелированы. Более того, вероятность того, что как минимум несколь-

ко признаков из 41 показателя тональности будут случайным образом коррелировать с целевой переменной на отложенных выборках, крайне высока. В этой связи использование всех признаков тональности в итоговых моделях (регрессиях и DART) также приведет к переобучению, вследствие чего на данном шаге снова необходимо провести отбор признаков.

Процедура отбора признаков тональности также является нетривиальной. Запускается цикл, в рамках которого к имеющимся признакам добавляется каждый из признаков тональности по отдельности и оценивается качество линейной регрессии как на отложенной, так и на тренировочной выборках. Далее отбираются все модели, которые имеют меньшее значение MSE как на отложенной, так и на тренировочной выборках – для того, чтобы избежать случайной зависимости признаков с целевой переменной на тестовых выборках. Среди них выбирается модель с наименьшим значением MSE на отложенной выборке, и используемый в данной регрессии признак тональности добавляется к имеющимся. Таким образом для каждого банка формируется свой набор признаков тональности, которые используются в итоговой линейной регрессии.

6.3 Создание модели DART

Наличие линейной зависимости между признаками не является серьезной проблемой в модели градиентного бустинга. Более того, используется модификация DART, позволяющая существенно снизить последствия переобучения алгоритма. Ввиду того, что в предыдущей секции было выделено умеренное количество признаков - 12 (четыре, основанных на временных показателях, и восемь, основанных на предыдущих значениях) - они используются в моделях для всех банков. Кроме того, в модель добавляются отобранные на предыдущем шаге признаки

тональности для банка.

В качестве деления на тренировочную и валидационную выборки для градиентного бустинга используется простое разбиение для временных рядов, где $\frac{3}{4}$ первых наблюдений формируют обучающую выборку, а остальные – отложенную. Это связано с тем, что построение нескольких моделей DART (как предполагает последовательная кросс-валидация) является очень времязатратным процессом.

Помимо этого, большую роль в моделях градиентного бустинга играют гиперпараметры. Правильный выбор гиперпараметров позволяет балансировать между переобучением и недообучением модели. Эмпирическим путем были получены следующие:

- тип градиентного бустинга (boosting): boosting,
- величина шага обучения (learning_rate): 0.01,
- максимальная глубина дерева (max_depth): 8,
- максимальное число листовых вершин в дереве (num_leaves): 100,
- максимальное число групп, на которые делятся значения признака (max_bin): 100,
- минимальное число объектов в листовой вершине (min_data_in_leaf): 15,
- доля признаков, случайно отбираемых для каждого дерева (feature_fraction): 0.5,
- доля объектов тренировочной выборки, случайно отбираемых для каждого дерева (bagging_fraction): 0.4,

- минимальная доля, на которую должна снизиться текущая ошибка для проведения разбиения по данному предикату (min_split_gain): 0.01
- коэффициент λ L1-регуляризации (lambda_1): 1,
- коэффициент λ L2-регуляризации (lambda_2): 1,
- минимальная сумма гессиана в листовой вершине (min_sum_hessian_in_leaf): 0.01

Для остальных гиперпараметров используются значения по умолчанию из библиотеки LighGBM. Ознакомиться с ними можно на сайте библиотеки. Новые деревья строятся до тех пор, пока не будет выполнен хотя бы один из критериев останова: либо для последних 10-ти деревьев качество алгоритма на валидационной выборке без них лучше, чем с ними, либо число деревьев превышает 2000. В качестве итоговой модели берется та, для которой значение функции потерь на отложенной выборке минимальное.

6.4 Тесты на значимость признаков тональности

Статистическая взаимосвязь признаков тональности с целевой переменной оценивается при помощи трех критериев: p-value коэффициентов на уровнях значимости 5% и 1%, F-теста и теста Диболда-Мариано, применяемых к регрессионным моделям, а также сравнением значений суммарного снижения функции потерь в модели градиентного бустинга для используемых признаков с аналогичными показателями для двух наиболее значимых признаков и двух случайных. Рассмотрим каждый из методов более подробно.

6.4.1 P-значение коэффициентов

P-value – вероятность для данной случайной величины ξ принять такое же или более экстремальное значение статистики при справедливой нулевой гипотезе: $p(\xi) = \mathbb{P}\{\xi|H_0\}$. Обозначим за $T(\xi)$ статистику, используемую при проверке нулевой гипотезы H_0, а ее функцию распределения за $F(t) = \mathbb{P}\{T < t\}$. Так как вероятность попасть в заданную точку в непрерывном распределении равна нулю ($\mathbb{P}\{\xi = t\} = 0$), при проверке правосторонней альтернативной гипотезы p-value равно $p(t) = \mathbb{P}\{T > t\} = 1 - \mathbb{P}\{T < t\} = 1 - F(t)$; при проверке левосторонней альтернативы $p(t) = \mathbb{P}\{T < t\} = F(t)$. Соответственно, в случае двусторонней альтернативной гипотезы $p(t) = 2min\{F(t); 1 - F(t)\}$.

Полученное p-значение сравнивается с заданным уровнем значимости α – вероятностью отклонить нулевую гипотезу в случае, когда она верна (допустимой вероятностью ошибки первого рода). Если p-value меньше α, нулевая гипотеза отвергается в пользу альтернативной. В противном случае нулевая гипотеза не отклоняется, а альтернативная – отвергается.

В данном случае P-значение используется для проверки нулевой гипотезы H_0 о незначимости коэффициентов. Следовательно, альтернативной гипотезой H_1 будет являться значимость коэффициентов на данном уровне значимости α. В качестве статистики используется t-распределение Стьюдента со степенями свободы $n - k$ (n - число наблюдений, k - число регрессоров в модели). Используемые значения $alpha$ – 0.01 и 0.05. Значение $\alpha = 0.1$ не рассматривается по той причине, что вероятность хотя бы для одного из сорока признаков быть значимым при отсутствии реальной зависимости с целевой переменной на данном уровне значимости со средним размером обучающей выборки в 862 наблюдений очень высока, что приведет к ошибке первого рода. Значение $\alpha = 0.05$ используется для того, чтобы отобрать те признаки тональности, для которых

взаимосвязь с целевой переменной вероятна, но не однозначна: использование значимости признаков на уровне $\alpha = 0.05$ часто подвергается критике за ошибки первого рода [Woolston, 2015]. Соответственно, значение $\alpha = 0.01$ взято для того, чтобы определять признаки, которые имеют выраженное влияние на целевую переменную. Использование P-значения в Python описано в приложении №8.

6.4.2 F-тест

Классический F-тест является корректным в случае, когда случайные ошибки модели имеют нормальное распределение. Поскольку в работе данная предпосылка не выполняется, используется его асимптотическая модификация, вычисляемая с помощью статистики асимптотического теста Вальда. Соответствующая F-статистика равна $F = \frac{n-k_{ur}}{q} \cdot \frac{W}{n}$, где n – число наблюдений (1294), k_{ur} – количество регрессоров в модели без ограничений (в модели с признаками тональности в данном случае), q - число ограничений (используемых признаков тональности), а W – значение статистики Вальда. Статистика Вальда рассчитывается как $\frac{MSE_r - MSE_{ur}}{MSE_{ur}/n}$, где MSE_r – среднее квадратичное отклонение в ограниченной модели (без признаков тональности), а MSE_{ur} – в модели с учетом признаков тональности. Полученное значение F-статистики имеет распределение Фишера (F-распределение) со степенями свободы q и $n - k_{ur}$. Оно сравнивается с соответствующим критическим значением на данном уровне значимости α. Если значение статистики превышает критическое значение, нулевая гипотеза о незначимости коэффициентов отвергается. В противном случае H_0 не отклоняется.

Аналогично P-значению, для F-теста в работе используются $\alpha = 0.01$ и $\alpha = 0.05$. Важно отметить, что в отличие от двух других критериев, F-тест проводится на всей выборке, а не только на обучающей. Проведение F-теста показано в приложении №9.

6.4.3 Тест Диболда-Мариано

Критерий Диболда-Мариано позволяет сравнивать качество прогнозов двух моделей в задаче предсказаний временных рядов. Тест проводится следующим образом. Обозначим за n число наблюдений, за $\{y_t\}_{t=1}^n$ значения временного ряда, за $\{y_{it}\}_{t=1}^n, i = 1, 2$ – прогнозные значения обеих моделей, за $g(x)$ – функцию потерь. Тогда остатки прогнозов моделей равны $\{y_{it} - y_t\}_{t=1}^n = \{e_{it}\}_{t=1}^n, i = 1, 2$. Следовательно, разница между качеством прогнозов равна $\{d_t\}_{t=1}^n = \{g(e_{1t}) - g(e_{2t})\}_{t=1}^n$. Нулевая гипотеза заключается в равенстве качества прогнозов моделей: $H_0 : \mathbb{E}(d) = 0$; альтернативная – в их отличии: $H_1 : \mathbb{E}(d) \neq 0$.

Если $\{d_t\}_{t=1}^n$ является слабостационарным временным рядом, то $\sqrt{n}(\overline{d} - \mu) \xrightarrow{d} N(0, \sigma^2)$, где $\overline{d} = \frac{1}{n}\sum_1^n d_t$ - матожидание разности между качеством прогнозов, μ - неизвестное матожидание процесса, σ^2 - его дисперсия. Вычисляемая статистика рассчитывается как $S = \frac{\overline{d}}{\sqrt{\overline{\sigma^2}/T}}$, где $\overline{\sigma^2} = \sum_{t=-\infty}^{t=+\infty} \gamma_d(\tau)$, где $\gamma_d(\tau)$ - автоковариация d порядка τ. Нулевой гипотезе соответствует $S \sim N(0, 1)$. Таким образом, полученное значение сравнивается с соответствующим критическим значением для данного уровня значимости α.

Как и в случае двух предыдущих тестов, в работе используются $\alpha = 0.05$ и $\alpha = 0.01$. Поскольку данный критерий не накладывает ограничений на функцию потерь, в работе используются средняя квадратичная ошибка MSE и среднее абсолютное отклонение, выраженное в процентах MAPE. Использование теста в Python описано в приложении №10.

6.5 Важность признака в модели DART

Модификация DART экстремального градиентного бустинга использует четыре временных признака, восемь показателей, основанных на предыдущих значениях, отобранные для данного банка признаки тональности и два случайных признака: имеющий стандартное нормальное распределение и стандартное равномерное. Важность признака в модели оценивается следующим образом. В каждом решающем дереве ансамбля для всех вершин, в которых предикат использует данный признак для разбиения, считается значение, на которое снизилась функция потерь (MSE в данном случае) после разбиения согласно предикату, после чего результаты суммируются для каждого дерева. Далее значение для каждого дерева умножается на его (дерева) вес, после чего все результаты снова суммируются. Полученная величина характеризует важность признака в модели: чем больше значение, тем выше важность признака. Величина сравнивается с аналогичными показателями для двух наиболее важных признаков (для всех банков это первое и последнее значение цен акций за день) и для двух случайных, что позволяет сделать выводы о важности признака в модели.

Процесс создания модели градиентного бустинга и получения важности принаков описан в приложении №11.

7 Результаты

В главе анализируются результаты трех выбранных тестов для рассматриваемых банков.

7.1 Сбербанк

В качестве регрессоров для модели предсказания акций Сбербанка без признаков тональности использовались следующие: первое и последнее значения за предыдущий день и их квадраты и разница между максимальным и минимальным значением за два предыдущих дня из группы предыдущих значений, дамми переменные, соответствующие вторнику, четвергу, августу, а также дням в месяце под номером 2, 14, 23, 28, 29 и 30 из временных показателей и единичный признак. Из признаков тональности были отобраны вероятность принадлежности к положительному классу в модели общих данных Твиттера и Leaky ReLU от нее ($proba\_tw, lrelu\_proba\_tw$), вероятность принадлежности к отрицательному классу в модели данных данных Твиттера касательно банков ($proba\_bank\_neg$) и модифицированный гиперболический тангенс от вероятности принадлежности к положительному классу в новостной модели ($tanh\_proba\_news\_pos$). Корреляционная матрица признаков тональности представлена на рисунке 1.12.

Итого для прогнозирования цены акций используются 16 регрессоров в модели без признаков тональности и 20 в модели с ними. Далее будут рассмотрены результаты каждого из используемых критериев.

7.1.1 P-значение

Результаты P-значения для Сбербанка представлены в таблице 1.1. На уровне значимости 5% значимыми являются сразу два признака – вероятность принадлежности к положительному классу для модели, обученной на общих данных Твиттера и вероятность принадлежности к отрицательному классу модели, обученной на данных Твиттера относительно банков. Вероятность принадлежности к положительному классу для модели общих данных Твиттера является также значимым на уровне

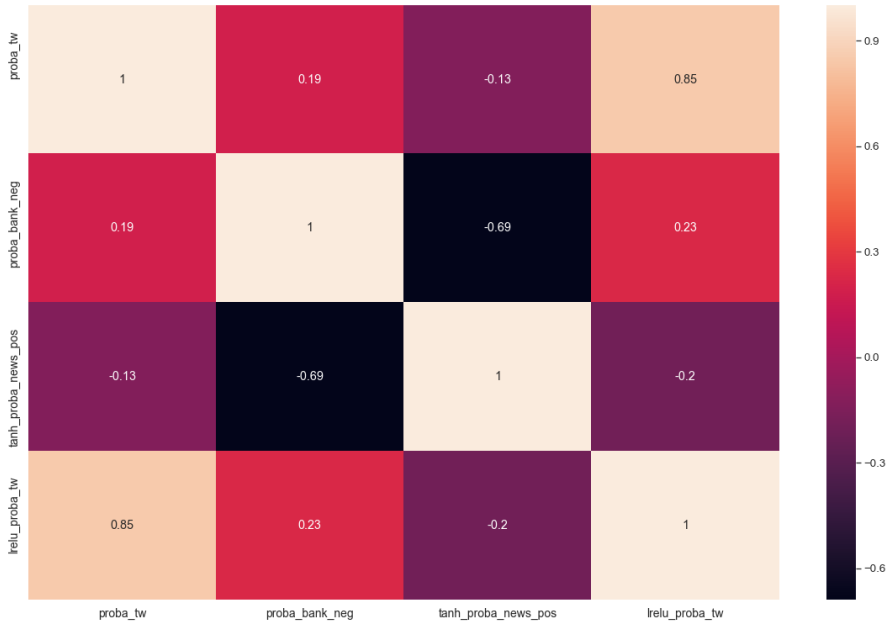

Рис. 1.12: Корреляционная матрица признаков тональности для Сбербанка.

значимости 1%, что означает, что данный признак определенно влияет на динамику целевой переменной. Скорее всего, большое p-значение для признака *lrelu_proba_tw* связано с тем, что данный признак очень сильно коррелирует с признаком *proba_tw*, в результате чего он оказался незначимым.

Признак	Коэф. $\hat{\beta}$	Ст. ошибка $\hat{se}(\beta)$	t-стат.	p-value
proba_tw**	0.012	0.0045	2.71	0.007
lrelu_proba_tw	0.0023	0.0013	1.73	0.084
proba_bank_neg*	-0.0085	0.0041	-2.04	0.042
tanh_proba_news_pos	0.0013	0.0017	0.74	0.462

Таблица 1.1: Результаты P-значения для Сбербанка. * – значимость на уровне $\alpha = 0.05$; ** – значимость на уровне $\alpha = 0.01$.

7.1.2 F-тест

Результаты F-теста для Сбербанка представлены в таблице 1.2. Тест проводится для моделей, использующих различные комбинации признаков. Поскольку в случае одного добавленного в неограниченную модель признака результаты F-теста идентичны P-значение, рассматривались модели с двумя или более добавленными признаками. Гипотеза о равенстве используемых ограничений нулю отвергается при $\alpha = 0.01$ для двух моделей, причем в обеих использовались признаки $proba\_tw$, $proba\_bank\_neg$. Это означает, что как минимум два используемых признака ($proba\_tw$, $proba\_bank\_neg$) имеют существенное влияние на целевую переменную – первое значение цены акций Сбербанка за день. Гипотеза о равенстве ограничений нулю для модели с признаками без $tanh\_proba\_news\_pos$ отвергается при $\alpha = 0.01$, в то время как для модели, использующей все четыре признака, не отвергается на данном уровне значимости. Помимо этого, модель, использующая признаки $lrelu\_proba\_tw$ и $tanh\_proba\_news\_pos$, не отвергается при $\alpha = 0.05$, Следовательно, признак $tanh\_proba\_news\_pos$ с большой долей вероятности не влияет на цену акций Сбербанка.

№	Используемые признаки	F-stat	P-значение
1**	proba_tw, proba_bank_neg	5.63	0.004
2*	proba_bank_neg, lrelu_proba_tw	3.21	0.041
3	lrelu_proba_tw, tanh_proba_news_pos	2.74	0.065
4*	proba_bank_neg, tanh_proba_news_pos	3.04	0.048
5**	proba_tw, proba_bank_neg, lrelu_proba_tw	4.35	0.005
6*	proba_tw, proba_bank_neg lrelu_proba_tw tanh_proba_news_pos	3.18	0.013

Таблица 1.2: Результаты F-теста для Сбербанка. * – значимость на уровне $\alpha = 0.05$; ** – значимость на уровне $\alpha = 0.01$.

7.1.3 Тест Диболда-Мариано

Результаты теста Диболда-Мариано для Сбербанка представлены в таблице 1.3. Поскольку предыдущие два теста показали очевидное влияние признаков *proba_tw* и *proba_bank_neg* на целевую переменную (о чем также свидетельствует результат данного критерия для модели, учитывающей все 4 признака), помимо модели, содержащей все 4 признака тональности, с моделью без признаков также сравнивались модель, содержащая только признак *lrelu_proba_tw* из признаков тональности, только признак *tanh_proba_news_pos*, признаки *proba_tw* и *proba_bank_neg* и признаки *proba_tw*, *proba_bank_neg* и *lrelu_proba_tw*. Последние две рассматривались для того, чтобы вы-

явить, позволяет ли признак $lrelu\_proba\_tw$ снизить ошибку прогнозов при использовании двух других и требуется ли его использование в оптимальных с точки зрения MSE и MAPE моделях.

С функцией потерь MSE при $\alpha = 0.01$ различие между моделью без признаками и рассматриваемыми оказалось значимо для всех моделей кроме той, где использовался только признак $tanh\_proba\_news\_pos$. Гипотеза о равенстве прогнозов данной модели и модели без показателей тональности не отвергается при любом разумном уровне значимости. С точки зрения MAPE различие между моделями также оказалось незначимым. В этой связи можно сделать вывод, что признак $tanh\_proba\_news\_pos$ не влияет на целевую переменную.

Различие между моделью, использующей только признак $lrelu\_proba\_tw$ и моделью без признаков тональности с точки зрения MAPE также значимо при $\alpha = 0.01$. Поскольку значение dm-статистики для обеих функций потерь в случае использования признака $lrelu\_proba\_tw$ меньше, чем без него, можно сделать вывод, что использование функции Leaky ReLU от признака $proba\_tw$ позволяет немного снизить значение ошибки прогнозов, вследствие чего данный признак также имеет влияние на котировки акций Сбербанка.

№	Используемые признаки	dm-стат. MSE	dm-стат. MAPE
1**	lrelu_proba_tw	0.0089	0.0092
2	tanh_proba_news_pos	0.228	0.1964
3**	proba_tw, proba_bank_neg,	2.9e-8	1.2e-7
4**	tanh_proba_news_pos, lrelu_proba_tw	0.0098	0.0136
5**	proba_tw, proba_bank_neg, lrelu_proba_tw	4.7e-12	3.5e-10
6**	proba_tw, proba_bank_neg, tanh_proba_news_pos, lrelu_proba_tw,	1.3e-9	2.7e-9

Таблица 1.3: Результаты теста Диболда-Мариано для Сбербанка. ** – значимость на уровне $\alpha = 0.01$ с точки зрения MSE.

7.1.4 Важность признаков в модели DART

Результаты сравнения важности признаков в модели DART представлены в таблице 1.4. Признак proba_tw позволяет снизить ошибку на 2.4 % – в 10 раз меньше, чем второй по важности признак. Более того, он является восьмым среди всех признаков по данному показателю, что означает наличие его статистической взаимосвязи с целевой переменной. Случайные признаки в модели DART практически не используются для разбиения. Следовательно, так как важность в процентах у признаков lrelu_proba_tw и proba_bank_neg значительно больше, чем у случайных (более чем в сто раз больше, чем у распределенного нормально), и есть как минимум четыре других признака (без учета tanh_proba_news_pos и случайных), имеющих меньшую важность, чем

оба признака, можно сделать вывод о наличии зависимости между данными показателями и целевой переменной (о том, что зависимость не случайная). Важность признака tanh_proba_news_pos превышает максимальный показатель среди случайных в менее чем 3 раза, и данный принак является наименее значимым среди всех признаков без учета случайных – из этого следует, что он имеет совсем небольшую зависимость с целевой переменной, которая могла проявиться в силу маленького размера выборки.

Признак	Важность (%)	Номер
Первое значение за пред. день	20.2	2
Последнее значение за пред. день	52.6	1
proba_tw	2.4	8
lrelu_proba_tw	0.3	10
proba_bank_neg	0.6	11
tanh_proba_news_pos	0.008	16
Случайный, станд. нормальный	0.003	17
Случайный, станд. равномерный	0.001	18

Таблица 1.4: Результаты сравнения важности признаков в DART для Сбербанка.

7.1.5 Вывод

Таким образом, на первое значение цен акций Сбербанка за день оказывают влияние два признака тональности твитов за предыдущий день: усредненное значение вероятностей принадлежности к положительному классу в модели общих данных Твиттера ($proba\_tw$) и усредненное значение вероятностей принадлежности к отрицательному классу в моде-

ли данных данных Твиттера касательно банков ($proba\_bank\_neg$). Использование преобразования функции Leaky ReLU от первого признака ($lrelu\_proba\_tw$) также позволяет снизить значение функции потерь.

График значений первых цен акций Сбербанка за день за рассматриваемый представлен на рисунке 1.13. Цвет отражает значение наиболее влиятельного признака, proba_tw, для записей за предыдущий день.

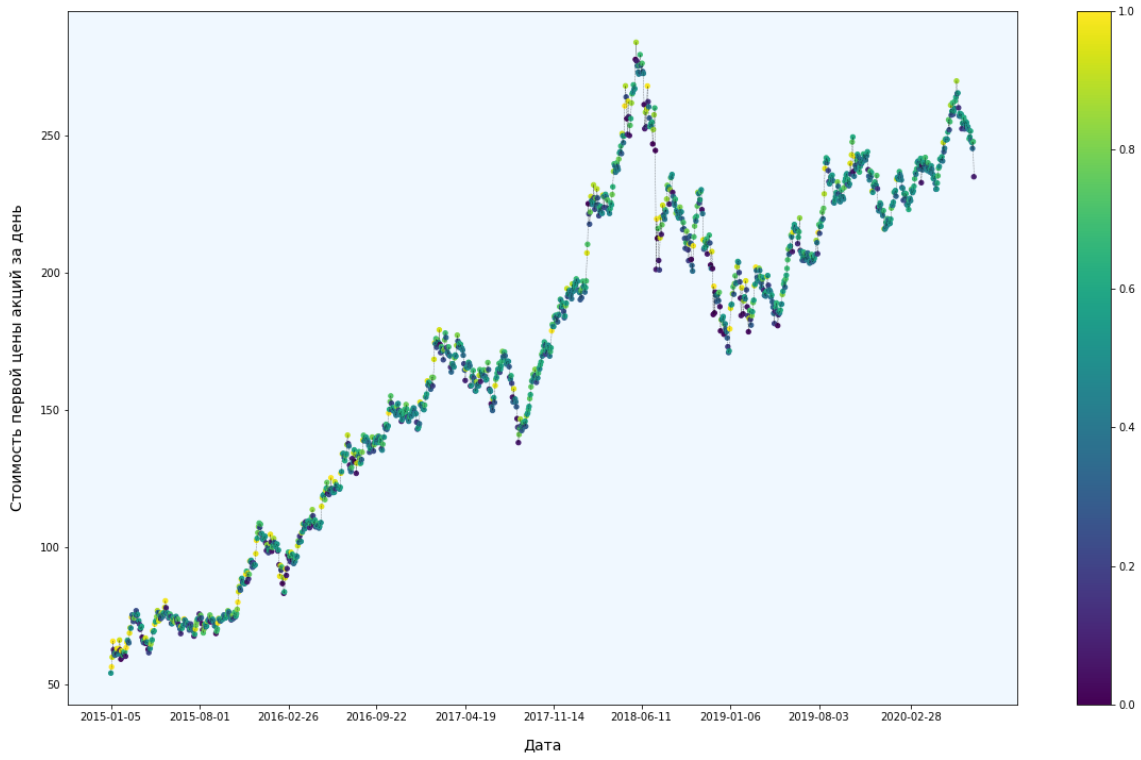

Рис. 1.13: График значений первых цен акций Сбербанка за день с подсветкой значений признака proba_tw.

7.2 ВТБ

В качестве регрессоров для модели предсказания акций ВТБ без признаков тональности использовались следующие: первое и последнее значения за предыдущий день, квадрат последнего значения и разница между максимальным и минимальным значением за два предыдущих дня из группы предыдущих значений, дамми переменные, соответствующие по-

недельнку, вторнику, январю, августу, а также дням в месяце под номером 4, 7, 29 и 30 из временных показателей и единичный признак. Из признаков тональности были отобраны вероятность принадлежности к положительному классу для модели общих данных Твиттера ($proba\_tw$), логарифмическая функция от вероятности принадлежности к положительному классу для модели данных данных Твиттера касательно банков ($log\_proba\_bank\_pos$) и квадрат от среднего значения предсказаний модели общих данных твиттера (sq_pred_tw) с сохранением исходного знака. Корреляционная матрица признаков тональности представлена на рисунке 1.14.

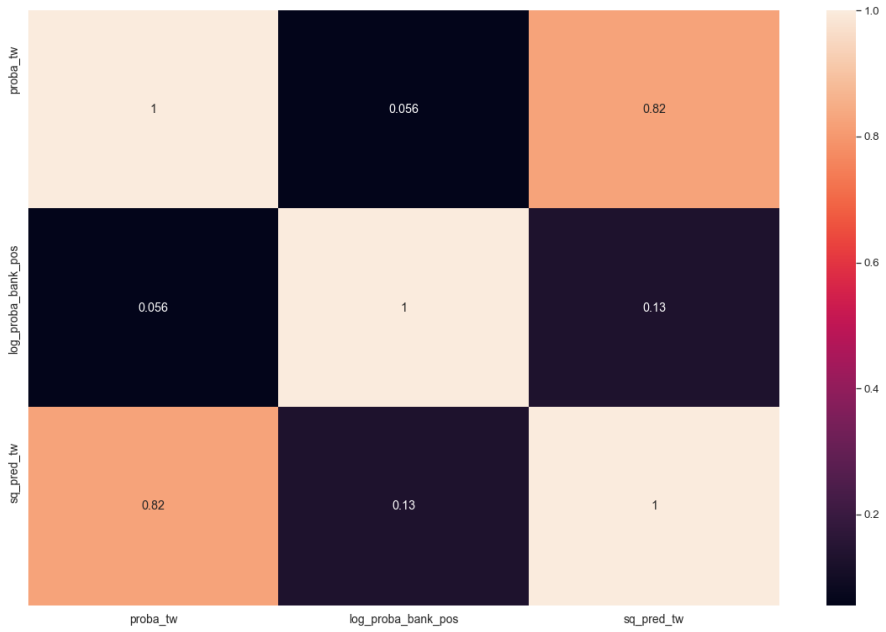

Рис. 1.14: Корреляционная матрица признаков тональности для ВТБ.

Итого для прогнозирования цены акций используются 14 регрессоров в модели без признаков тональности и 17 в модели с ними. Далее будут рассмотрены результаты каждого из используемых критериев.

7.2.1 P-значение

Результаты P-значения для ВТБ представлены в таблице 1.5. На уровне значимости 5% значимыми являются все три признака. Признак *log_proba_bank_pos*, результат применения логарифмической функции к вероятности принадлежности к положительному классу в модели, обученной на данных Твиттера касательно банков, является также значимым на уровне $\alpha = 0.01$, что свидетельствует о его выраженной зависимости с котировкой акций ВТБ.

Признак	Коэф. β	Ст. ошибка $\hat{se}(\beta)$	t-стат.	p-value
log_proba_bank_pos**	0.0092	0.0033	2.7878	0.0053
sq_pred_tw*	0.0021	0.00093	2.2826	0.0226
proba_tw*	0.0089	0.0041	2.1732	0.03

Таблица 1.5: Результаты P-значение для ВТБ. * – значимость на уровне $\alpha = 0.05$; ** – значимость на уровне $\alpha = 0.01$.

7.2.2 F-тест

Результаты F-теста для ВТБ представлены в таблице 1.6. В качестве признаков тональности для моделей использовались их всевозможные комбинации за исключением случаев с одним признаком. Гипотеза о незначимости коэффициентов отверглась во всех случаях на уровне значимости 5%. Более того, только в модели, использующей признаки *sq_pred_tw* и *proba_tw*, нулевая гипотеза не отверглась на уровне $\alpha = 0.01$. Скорее всего, это вызвано коррелированостью данных признаков. Незначимость нулевой гипотезы в остальных моделях при $\alpha = 0.01$ означает наличие взаимосвязи целевой переменной с каждым из трех признаков.

№	Используемые признаки	F-stat	P-значение
1**	proba_tw, log_proba_bank_pos, sq_pred_tw	4.2	0.006
2**	log_proba_bank_pos, sq_pred_tw	6.29	0.002
3*	sq_pred_tw, proba_tw	4.31	0.014
4**	proba_tw, log_proba_bank_pos	4.73	0.009

Таблица 1.6: Результаты F-теста для ВТБ. * – значимость на уровне $\alpha = 0.05$; ** – значимость на уровне $\alpha = 0.01$.

7.2.3 Тест Диболда-Мариано

Результаты теста Диболда-Мариано для ВТБ представлены в таблице 1.7. Помимо модели со всеми используемыми признаками, с моделью без признаков тональности сравнивались следующие: модель, использующая только признак *proba_tw*, только признак *sq_pred_tw* и использующая оба признака. Для всех моделей разница между ними и моделью без признаков тональности оказалась значимой при $\alpha = 0.01$ для обеих метрик. Следовательно, использование признаков *sq_pred_tw* и *proba_tw* позволяет улучшить качество прогнозов, что говорит о наличии его влияния на значение первой цены акций ВТБ за день.

Близкое к нулю значение dm-статистики для обеих метрик для модели со всеми тремя признаками означает, что целевая переменная имеет выраженную зависимость от данных признаков тональности.

№	Используемые признаки	dm-стат. MSE	dm-стат. MAPE
1**	proba_tw, sq_pred_tw, log_proba_bank_pos	4.4e-9	8.5e-10
2**	proba_tw	7.6e-3	9.2e-3
3**	sq_pred_tw	3.2e-4	2e-3
4**	sq_pred_tw, proba_tw	4.5e-5	1.1e-5

Таблица 1.7: Результаты теста Диболда-Мариано для ВТБ. * – значимость на уровне $\alpha = 0.05$ с точки зрения MSE; ** – значимость на уровне $\alpha = 0.01$ с точки зрения MSE.

7.2.4 Важность признаков в модели DART

Результаты сравнения важности признаков в модели DART представлены в таблице 1.8. Два самых важных признака снижают в сумме ошибку на 73.7% – следовательно, оставшиеся 13 признаков (без учета случайных) снижают ошибку на 26.3%. Поскольку каждый из трех признаков тональности снижает ошибку не менее, чем на 0.9%, относительно остальных признаков они снижают ошибку не менее, чем на 3.5%. Все признаки тональности также имеют большую значимость, чем как минимум четыре других показателя. Кроме того, случайные признаки в сумме снижают ошибку на 0.01% (меньше наименее важного из признаков тональности в 150 раз), что означает, что зависимость между каждым из используемых признаков тональности и целевой переменной не является случайной.

Признак	Важность (%)	Номер
Первое значение за пред. день	35.1	2
Последнее значение за пред. день	38.6	1
log_proba_bank_pos	1.5	9
sq_pred_tw	0.9	11
proba_tw	1.3	10
Случайный, станд. нормальный	0.002	17
Случайный, станд. равномерный	0.008	16

Таблица 1.8: Результаты сравнения важности признаков в DART для ВТБ.

7.2.5 Вывод

Проведенные тесты демонстрируют, что первое значение цен акций ВТБ за день имеет устойчивую зависимость от трех признаков тональности твитов за предыдущий день: логарифма среднего значения вероятностей принадлежности к положительному классу в модели данных Твиттера относительно банков *proba_bank_pos*, среднего значения вероятностей принадлежности к положительному классу в модели общих данных Твиттера (*proba_tw*) и квадрата с сохранением исходного знака среднего значения предсказанного класса в модели общих данных Твиттера (*sq_pred_tw*), где положительному классу соответствует 1, а отрицательному – -1.

График значений первых цен акций ВТБ за день за рассматриваемый представлен на рисунке 1.15. Цвет отражает значение наиболее влиятельного признака, proba_bank_pos, для записей за предыдущий день.

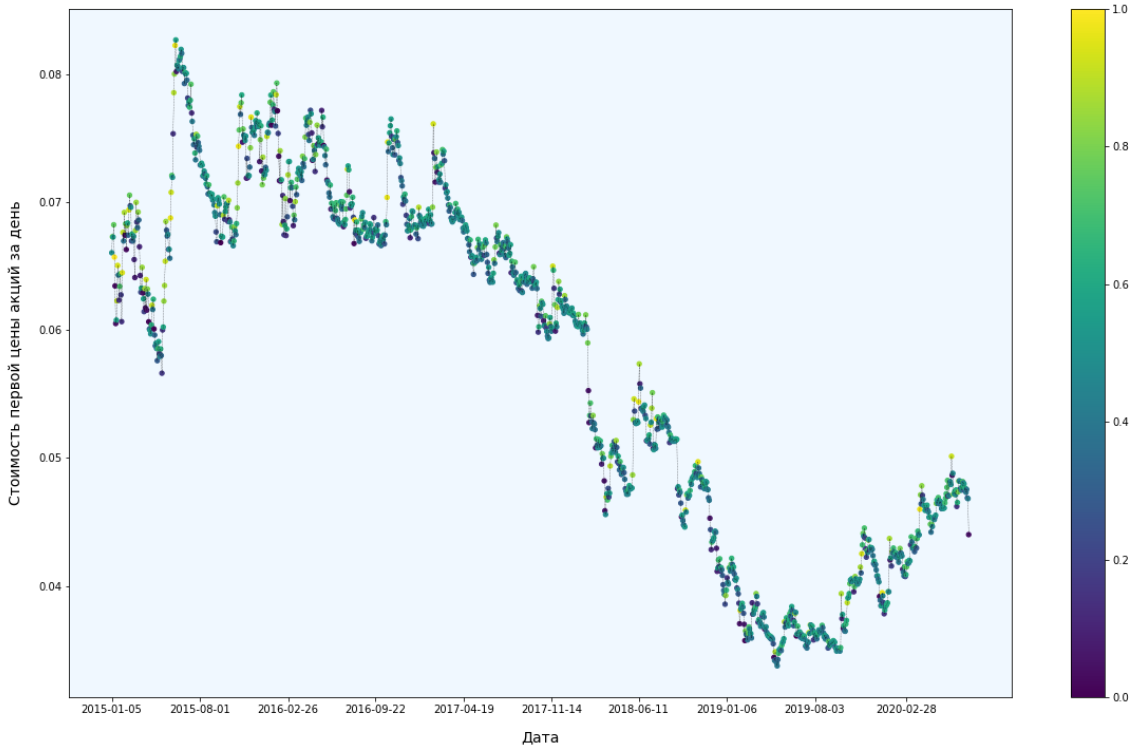

Рис. 1.15: График значений первых цен акций ВТБ за день с подсветкой значений признака proba_bank_pos.

7.3 Росбанк

В качестве регрессоров для модели предсказания акций Росбанка без признаков тональности использовались следующие: первое и последнее значения за предыдущий день, первое значение два дня назад и квадрат последнего значения за предыдущий день из группы предыдущих значений, дамми переменные, соответствующие январю, декабрю и дням в месяце под номером 1, 3, 25 и 29 из временных показателей и единичный признак. Из признаков тональности были отобраны модифицированный гиперболический тангенс вероятности принадлежности к положительному классу для новостной модели ($tanhcb\_proba\_news\_pos$), вероятность принадлежности к положительному классу для модели общих данных Твиттера ($proba\_tw$) и Leaky ReLU от среднего значения предсказаний модели данных Твиттера относительно банков ($lrelu\_pred\_bank$).

Корреляционная матрица признаков тональности представлена на рисунке 1.16.

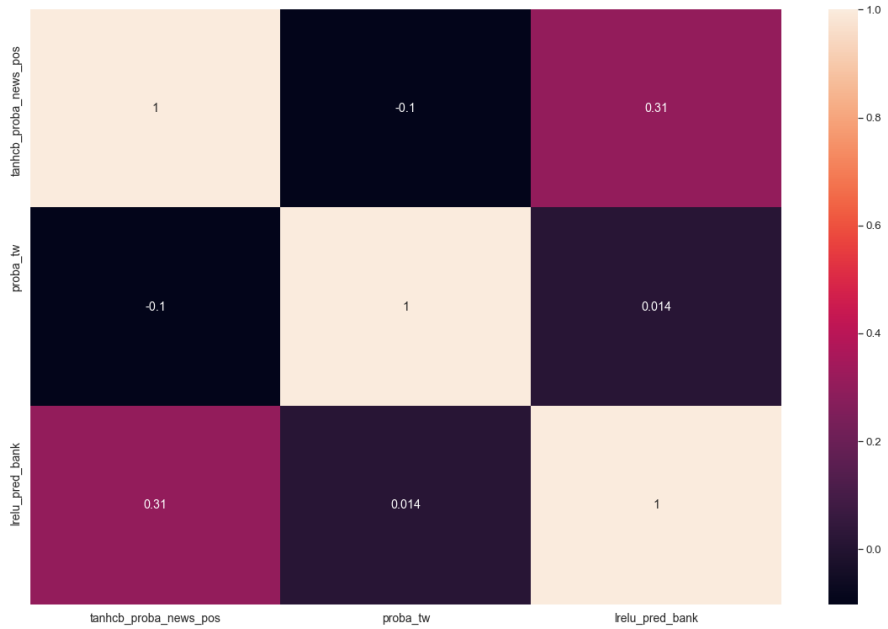

Рис. 1.16: Корреляционная матрица признаков тональности для Росбанка.

Итого для прогнозирования цены акций используются 11 регрессоров в модели без признаков тональности и 14 в модели с ними. Далее будут рассмотрены результаты каждого из используемых критериев.

7.3.1 P-значение

Результаты P-значения для Росбанка представлены в таблице 1.9. На уровне значимости 5% значимым оказался только один признак, $proba\_tw$. Двое других оказались значимыми на уровне $\alpha = 0.1$, но, как уже отмечалось, данный уровень значимости не является показательным в силу большого числа используемых признаков и недостаточного объема выборки. На уровне значимости 1% значимых признаков тональности нет.

Признак	Коэф. β	Ст. ош. $\hat{se}(\beta)$	t-стат.	p-value
proba_tw*	0.0086	0.004	2.15	0.032
tanhcb_proba_news_pos	0.0019	0.0011	1.81	0.071
lrelu_pred_bank	0.0019	0.0012	1.652	0.099

Таблица 1.9: Результаты P-значения для Росбанка. * – значимость на уровне $\alpha = 0.05$.

7.3.2 F-тест

Результаты F-теста для Росбанка представлены в таблице 1.10. Аналогично процедуре F-теста для ВТБ в качестве признаков тональности для сравниваемой модели использовались их всевозможные комбинации за исключением случаев с одним признаком. Гипотеза о незначимости ограничений отверглась в трех случаях на уровне значимости 5%. Гипотеза о равенстве коэффициентов нулю не отверглась для модели с признаками $tanhcb\_proba\_news\_pos$ и $lrelu\_pred\_bank$. Также при $\alpha = 0.01$ нулевая гипотеза не отверглась ни в одном из случаев, что указывает на слабую зависимость этих признаков с динамикой первой цены акций за день Росбанка.

№	Используемые признаки	F-stat	P-значение
1*	proba_tw, lrelu_pred_bank	3.03	0.049
2*	tanhcb_proba_news_pos, proba_tw, lrelu_pred_bank	2.65	0.048
3*	tanhcb_proba_news_pos, proba_tw	3.39	0.034
4	tanhcb_proba_news_pos, lrelu_pred_bank	2.11	0.122

Таблица 1.10: Результаты F-теста для Росбанка. * – значимость на уровне $\alpha = 0.05$.

7.3.3 Тест Диболда-Мариано

Результаты теста Диболда-Мариано для Росбанка представлены в таблице 1.11. С моделью без признаков тональности сравнивались модели со всевозможными комбинациями признаков тональности, поскольку предыдущие два теста не выявили выраженного влияния каких-либо признаков тональности на целевую переменную. Для всех моделей, в которых использовался признак $proba\_tw$, разница с моделью без признаков тональности оказалось существенной на уровне значимости 0.01. Следовательно, данный признак имеет зависимость с целевой переменной.

Модель, использующая только признак $tanhcb\_proba\_news\_pos$, оказалась значимой при $alpha = 0.05$. Кроме того, модели, использующие данный признак, имеют меньшее p-value (то есть меньшую вероятность отвергнуть нулевую гипотезу при ее справедливости), чем аналогичные модели без него. Таким образом, сделать однозначный вывод о влиянии признака на цену акций Росбанка невозможно. Скорее всего, признак

№	Используемые признаки	dm-стат. MSE	dm-стат. MAPE
1*	tanhcb_proba_news_pos	0.02	0.046
2	lrelu_pred_bank	0.052	0.084
3**	proba_tw	8.4e-4	9.5e-4
4**	proba_tw, lrelu_pred_bank	2.6e-4	4.5e-4
5**	tanhcb_proba_news_pos, proba_tw, lrelu_pred_bank	1.6e-6	2.8e-5
6**	tanhcb_proba_news_pos, proba_tw	2e-4	4.7e-4
7*	tanhcb_proba_news_pos, lrelu_pred_bank	0.012	0.027

Таблица 1.11: Результаты теста Диболда-Мариано для Росбанка. * – значимость на уровне $\alpha = 0.05$ с точки зрения MSE; ** – значимость на уровне $\alpha = 0.01$ с точки зрения MSE.

tanhcb_proba_news_pos имеет зависимость с целевой переменной, однако эта зависимость является очень слабой.

Что касается признака *lrelu_pred_bank*, то для модели, использующей только данный признак, разница прогнозов с моделью без признаков тональности оказалась незначимой на уровне $\alpha = 0.05$. В этой связи гипотеза о незначимости коэффициента для прогнозирования первой цены акций Росбанка за день не отвергается.

7.3.4 Важность признаков в модели DART

Результаты сравнения важности признаков в модели DART представлены в таблице 1.12. Небольшие значения показателей для признаков тональности обусловлены тем, что два самых значимых признака в сумме

снижают ошибку более, чем на 82% – в этой связи веса признаков существенно ниже, чем в моделях для других банков, так как веса должны удовлетворять условию нормировки. Тот факт, что признак proba_tw является девятым из пятнадцати (без учета случайных признаков), означает, что он выше двух временных признаков или показателей, основанных на предыдущих значениях. Это подтверждает наличие его статистической взаимосвязи с целевой переменной. В то же время на основании данного сравнения сделать однозначный вывод о наличии зависимости между двумя другими признаками тональности и значением первой цены акций Росбанка за день затруднительно.

Признак	Важность (%)	Номер
Первое значение за пред. день	18.8	2
Последнее значение за пред. день	63.4	1
proba_tw	0.9	9
tanhcb_proba_news_pos	0.4	13
lrelu_pred_bank	0.6	12
Случайный, станд. нормальный	0.004	16
Случайный, станд. равномерный	0.0	17

Таблица 1.12: Результаты сравнения важности признаков в DART для Росбанка.

7.3.5 Вывод

Три проведенных теста и результат важности признаков в модели градиентного бустинга демонстрируют зависимость первого значения цены акций Росбанка за день от одного признака тональности твитов предыдущего дня –средней вероятностью принадлежности к положительно-

му классу в модели, обученной на общих данных Твиттера ($proba\_tw$). На целевую переменную также влияет модифицированный гиперболический тангенс среднего значения вероятностей принадлежности записей за предыдущий день к положительному классу в новостной модели ($tanhcb\_proba\_news\_pos$), однако ее зависимость с данным признаком является неоднозначной, так как по результатам P-значения он оказался незначимым при $\alpha = 0.05$.

График значений первых цен акций Росбанка за день за рассматриваемый представлен на рисунке 1.17. Цвет отражает значение наиболее влиятельного признака, proba_tw, для записей за предыдущий день.

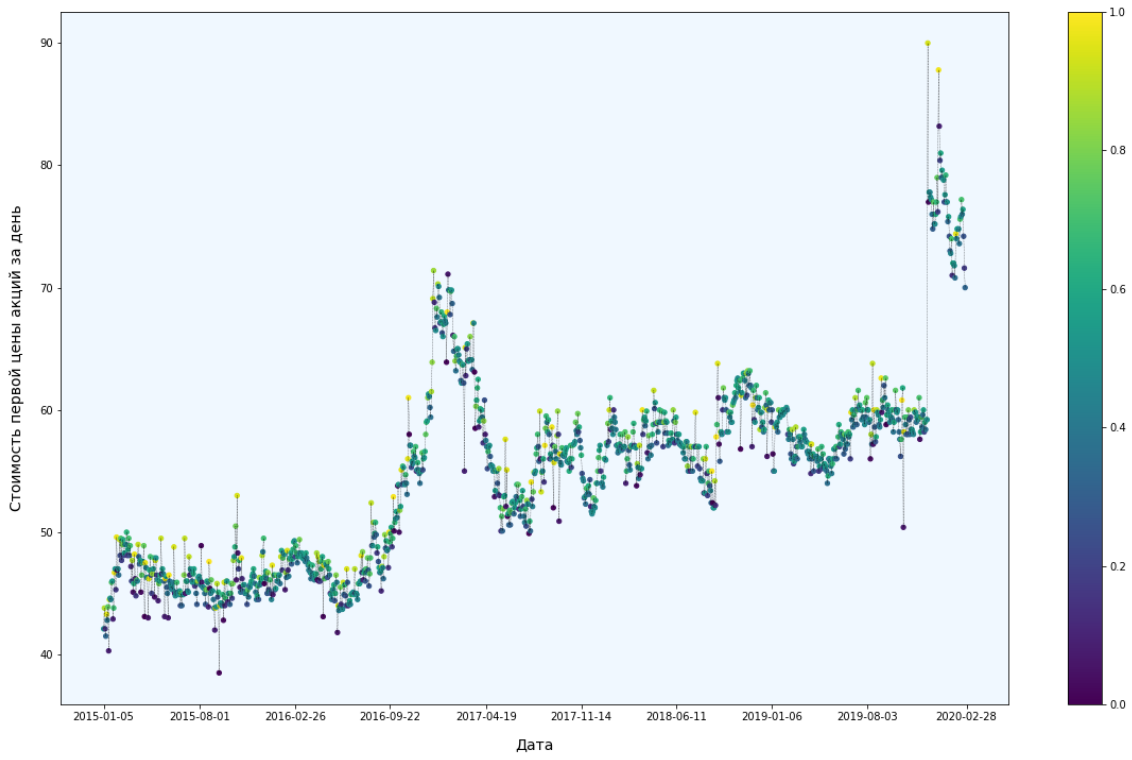

Рис. 1.17: График значений первых цен акций Росбанка за день с подсветкой значений признака proba_tw.

7.4 Итоги

Для выявления зависимости цен акций Сбербанка, ВТБ и Росбанка от различных признаков тональности записей в Твиттере за предыдущий день были проведены три теста: P-значение, F-тест, тест Диболда-Мариано и сравнение значений суммарного снижения функции потерь в модели градиентного бустинга для используемых признаков с аналогичными показателями для двух наиболее значимых признаков и двух случайных. Результаты представлены в таблице 1.13. На цены акций Сбербанка существенное влияние оказывают такие признаки тональности данных Твиттера за предыдущий день, как усредненное значение вероятностей принадлежности к положительному классу в модели, обученной на общих данных Твиттера и результат применения к нему Leaky ReLU, а также усредненное значение вероятностей принадлежности к отрицательному классу в модели, обученной на данных данных Твиттера отоносительно банков.

Котировки акций ВТБ имеют выраженную зависимость со следующими признаками тональности твитов за предыдущий день: логарифм среднего значения вероятностей принадлежности к положительному классу в модели, обученной на данных Твиттера относительно банков, среднее значение вероятностей принадлежности к положительному классу в модели, обученной на общих данных Твиттера и результат от применения квадрата с сохранением исходного знака к среднему значению предсказанного класса в модели, обученной на общих данных Твиттера, в которой положительному классу соответствует 1, а отрицательному – -1.

Котировки цен акций Росбанка имеют зависимость со средним значением вероятностей принадлежности к положительному классу в модели, обученной на общих данных Твиттера, для записей Твиттера за

Банк	Значимые при $\alpha = 0.01$ для всех критериев	Значимые при при $\alpha = 0.05$ для всех критериев	Значимые при $\alpha = 0.05$ для части критериев
Сбербанк	proba_tw	proba_bank_neg	lrelu_proba_tw
ВТБ	log_proba_bank_pos	proba_tw, sq_pred_tw	-
Росбанк	-	proba_tw	tanhcb_proba_news_pos

Таблица 1.13: Результаты теста Диболда-Мариано для Росбанка. * – значимость на уровне $\alpha = 0.05$ с точки зрения MSE; ** – значимость на уровне $\alpha = 0.01$ с точки зрения MSE.

предыдущий день. Тот факт, что на цену акций Росбанка оказывает значимое влияние только один признак, а на цены акций Сбербанка и ВТБ по три, скорее всего, вызван слишком маленьким числом твитов, содержащих упоминание Росбанка. Вследствие непопулярности банка, число записей в день с упоминанием Росбанка в среднем в 4 раза меньше, чем с упоминанием ВТБ, и в 5 раз меньше, чем с упоминанием Сбербанка, в результате чего выборка наблюдений за день могла часто получаться нерепрезентативной в силу маленького размера.

Признак, характеризующий вероятность принадлежности к положительному классу в модели, обученной на общих данных Твиттера, оказался значимым для всех трех исследуемых банков. В этой связи его можно считать «ключевым» признаком тональности для предсказания цен акций российских банков, так как он наиболее ярко демонстрирует зависимость котировок акций банков от тональности данных Твиттера.

8 Заключение

В работе представлены результаты исследования зависимости цен акций трех российских банков, Сбербанка, ВТБ и Росбанка, от тональности записей в Твиттере, связанных с ними, за предыдущий день. В ходе исследования были выполнены поставленные в начале работы задачи и достигнута цель исследования. Далее представлены выводы по каждой из задач.

В исследовании проведена процедура предобучения модели обработки языка ULMFiT на корпусе русскоязычных данных Твиттера. Набор данных взят из работы [Ю. Рубцова, 2014]. Модель может быть использована в различных целях, например, для генерации текста или получения эмбеддингов слов.

Для обучения предобученной модели были использованы три размеченных набора данных: набор данных Твиттера общего характера, набор данных Твиттера касательно банков и набор новостных данных. Далее для каждой выборки обучалась языковая модель, параметры которой настраивались под конкретный набор данных. После этого происходило обучение итоговых классификаторов.

Данные Твиттера с упоминанием исследуемых банков скачивались при помощи специальной библиотеки, приспособленной под данную задачу. Обученные классификаторы предсказывали вероятность принадлежности к каждому классу и итоговый класс для каждого твита, после чего результаты за день усреднялись и удалялись линейно-зависимые признаки. К полученным признакам тональности также применялись четыре нелинейные функции: логарифм значения плюс единица, модифицированный гиперболический тангенс, Leaky ReLU с параметром $\alpha = 0.1$ и возведение в квадрат с сохранением исходного знака.

На следующем шаге для предсказания цен акций каждого из банков

были созданы оптимальные с точки зрения среднего квадратичного отклонения линейные регрессии, использующие временные показатели и признаки, основанные на предыдущих значениях. Далее для каждой регрессии отбирались признаки тональности, снижающие значение среднего квадратичного отклонения на всей выборке. В результате для каждого банка были обучены две линейные регрессии, с признаками тональности и без них.

На заключительном этапе влияние признаков тональности на целевую переменную оценивалось при помощи трех критериев: P-значения, F-теста и теста Диболда-Мариано. Помимо этого, суммарное значение снижения функции потерь в модели градиентного бустинга для каждого признака сравнивалось с аналогичным показателем для первого и последнего значений акций за предыдущий день (два наиболее важных признака во всех моделях), а также с двумя случайными признаками - распределенными нормально и равномерно. По результатам критериев для каждого банка были выделены признаки тональности, имеющие выраженное влияние на котировки его акций. В частности, признак, характеризующий среднее значение вероятностей принадлежности твитов за предыдущий день к положительному классу в модели, обученной на общих данных Твиттера, оказался значимым для всех банков. Данный признак наиболее ярко демонстрирует влияние тональности записей в Твиттере на цены акций российских банков на следующий день. Таким образом, цель исследования – выявление наличия зависимости котировок акций российских банков от тональности данных Твиттера – была достигнута.

Весь код, использовавшийся в работе, представлен в репозитории на платформе github.com.

Дальнейшие исследования

Исследование обладает рядом ограничений, нивелирование которых может стать основой для дальнейших исследований в данной области. Во-первых, это отсутствие размеченного под конкретную задачу набора данных. Используемые для обучения наборы данных имеют некоторые различия с целевой выборкой, что приводит к зашумленности признаков тональности. Использование размеченного набора данных позволит модели более точно предсказывать тональность данных Твиттера.

Далее, в работе не рассматривается зависимость котировок акций банка от тональности данных относительно других банков. Между тем наличие подобной взаимосвязи является очень вероятным. Использование соответствующих признаков позволит снизить ошибку прогнозов итоговой регрессии.

В-третьих, ограничением является отсутствие графичесского процессора (GPU). При его наличии вместо ULMFiT можно обучить более современные архитектуры, такие как T5 или BERT. Использование данных NLP моделей также позволит снизить ошибку предсказания тональности твитов.

Помимо этого, в рамках дальнейших исследований могут быть отобраны другие российские компании и изучено влияние тональности данных Твиттера на цены их акций.

Список литературы

А.В.Дубко, К.Буассье. Влияние социальных медиа на волатильность цен на акции: состояние проблемы. — 2015.

Андрианова Е. Г., Новикова О. А. Роль методов интеллектуального анализа текста в автоматизации прогнозирования рынка ценных бумаг // Cloud of Science. — 2018. — т. 5, № 1.

Лукашевич Н. В., Рубцова Ю. В. SentiRuEval-2016: преодоление временных различий и разреженности данных для задачи анализа репутации по сообщениям Твиттера // Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialogue 2016". — 06.2016. — URL: http://www.dialog-21.ru/media/3410/loukachevitchnvrubtsovayv.pdf.

Лысенко В. Д. Анализ тональности текста для прогнозирования цен на фондовом рынке // Молодой ученый. — 2018. — 22 (208). — с. 420—423.

Рубцова Ю. Построение корпуса текстов для настройки тонового классификатора // Программные Продукты и Системы. — 2014.

A Gentle Introduction to Backpropagation Through Time //. — Jason Brownlee, 2017.

Banea C., Mihalcea R., Wiebe J., Hassan S. Multilingual Subjectivity Analysis Using Machine Translation // Conference on Empirical Methods in Natural Language Processing. — 2008.

Bekaert G., *Harvey C.* Time-Varying World Market Integration // The Journal of Finance. — 1995. — Vol. 50, no. 2. — P. 403–444.

Bengio Y., *Ducharme R.*, *Vincent P.* A Neural Probabilistic Language Model //. Advances in Neural Information Processing Systems 13 (NIPS'00). — MIT Press, 2001.

Bengio Y., *Ducharme R.*, *Vincent P.*, *Jauvin C.* A Neural Probabilistic Language Model // Journal of Machine Learning Research. — 2003. — Vol. 3. — P. 1137–1155.

Bollen J., *Mao H.*, *Zeng X.* Twitter mood predicts the stock market // Journal of Computational Science. — 2011. — Mar. — Vol. 2, no. 1. — P. 1–8. — eprint: 1010.3003 (cs.CE). — URL: https://arxiv.org/pdf/1010.3003.pdf.

Butler K. C., *Malaikah S. J.* Efficiency and inefficiency in thinly traded stock markets: Kuwait and Saudi Arabia // Journal of Banking and Finance. — 1992. — Feb. — Vol. 16, no. 1. — P. 197–210.

Cootner P. H. The random character of stock market prices. — MIT Press, 1964.

Ederington L. H., *Lee J. H.* How Markets Process Information: News Releases and Volatility // The Journal of Finance. — 1993. — Sept. — Vol. 48, no. 4. — P. 1161–1191.

Faltl S., *Schimpke M.*, *Hackober C.* Universal Language Model Fine-Tuning (ULMFiT): State-of-the-Art in Text Analysis // Humboldt-University Berlin: Information Systems (WS18/19). — 2019.

Fama E. F. Efficient Capital Markets: II // The Journal of Finance. — 1991. — Dec. — Vol. 46, no. 5. — P. 1575–1617.

FastAI. — URL: https://forums.fast.ai/t/ulmfit-russian/36312.

Felbo B., *Mislove A.*, *Søgaard A.*, *Rahwan I.*, *Lehmann S.* Using millions of emoji occurrences to learn any-domain representations for detecting

sentiment, emotion and sarcasm // arXiv. — 2017. — Oct. — arXiv: 1708.00524v2 [stat.ML].

Go A., *Bhayani R.*, *Huang L.* Twitter Sentiment Classification using Distant Supervision. — 2009. — Stanford Computer Science.

He B., *Macdonald C.*, *He J.*, *Ounis I.* An Effective Statistical Approach to Blog Post Opinion Retrieval // 17th ACM Conference on Information and Knowledge Management. — 2008.

Heusel M., *Ramsauer H.*, *Unterthiner T.*, *Nessler B.*, *Hochreiter S.* GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium // arXiv. — 2018. — Jan. — arXiv: 1706.08500v6 [cs.LG].

Howard J., *Ruder S.* Universal Language Model Fine-tuning for Text Classification // arXiv. — 2018. — May. — arXiv: 1801.06146v5 [cs.CL].

Jensen M. C. Some Anomalous Evidence Regarding Market Efficiency // Journal of Financial Economics. — 1978. — Vol. 6. — P. 95–101.

Kim O., *E.Verrecchia R.* Market Liquidity and Volume Around Earnings Announcements // Journal of Accounting and Economics. — 1994. — Jan. — Vol. 17. — P. 41–67.

Kirlić A., *Orhan Z.*, *Hasovic A.*, *Kevser-Gokgol M.* Stock Market Prediction Using Twitter Sentiment Analysis // Invention Journal of Research Technology in Engineering and Management (IJRTEM). — 2018. — Jan. — Vol. 2. — P. 01–04.

Merity S., *Keskar N. S.*, *Socher R.* Regularizing and Optimizing LSTM Language Models // arXiv. — 2017. — arXiv: 1708.02182 [cs.CL].

Mikolov T., *Chen K.*, *Corrado G.*, *Dean J.* Efficient Estimation of Word Representations in Vector Space // arXiv. — 2013. — arXiv: 1301.3781v3 [cs.CL].

Mitchell M., *Mulherin J.* The Impact of Public Information on the Stock Market // The Journal of Finance. — 1994. — July. — Vol. 49, no. 3. — P. 923–950. — URL: https://www.jstor.org/stable/2329211.

Mittal A., *A.Goel.* Stock Prediction Using Twitter Sentiment Analysis : tech. rep. / Stanford University, Computer Science Department. — 2011. — URL: http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf.

Moukalled M., *El-Hajj W.*, *Jaber M.* Automated Stock Price Prediction Using Machine Learning : tech. rep. / American University of Beirut, Computer Science Department. — 2019.

Muthukrishnan R., *Rohini R.* LASSO: A feature selection technique in predictive modeling for machine learning // 2016 IEEE International Conference on Advances in Computer Applications (ICACA). — 10/2016. — P. 18–20. — DOI: 10.1109/ICACA.2016.7887916.

Neurohive. — URL: https://neurohive.io/ru/papers.

Niederhoffer V. The Analysis of World Events and Stock Prices // Journal of Business. — 1971. — Vol. 44. — P. 193–219.

Nielsen M. Improving the way neural networks learn, Neural Networks and Deep Learning. — 10/2018.

Olah C. Colah's blog. — (Visited on).

Paper With Code. — URL: https://paperswithcode.com/area/natural-language-processing.

R.French K., *Roll R.* Stock return variances: The arrival of information and the reaction of traders. — 1986.

Rahman A., *Thevaraja 0.*, *Gabriel M.* Recent Developments in Data Science: Comparing Linear, Ridge and Lasso Regressions Techniques Using Wine Data. — 04/2019.

Rashmi K. V., *Gilad-Bachrach R.* DART: Dropouts meet Multiple Additive Regression Trees // arXiv. — 2015. — May. — arXiv: 1505.01866v1 [cs.LG].

Ruder S. An overview of gradient descent optimization algorithms // arXiv. — 2017. — June. — arXiv: 1609.04747v2 [cs.LG].

Sasank V. P., *Challa K. N. R.*, *Panda G.*, *Majhi B.* Sentiment Analysis of Twitter Data for Predicting Stock Market Movements // International conference on Signal Processing, Communication, Power and Embedded System (SCOPES)-2016. — 2016.

Schapire R. E., *Singer Y.* BoosTexter: A Boosting-based System for Text Categorization // Machine Learning. — 2000. — Vol. 39. — P. 135–168.

Schwert G. W. The Adjustment of Stock Prices to Information About Inflation // Journal of Finance. — 1981. — Vol. 36. — P. 15–29.

Smith L. N. Cyclical Learning Rates for Training Neural Networks // arXiv. — 2017. — Apr. — arXiv: 1506.01186v6 [cs.CV].

The Publications and Writings of Jack Kiefer // Ann. Statist. — 1984. — June. — Vol. 12, no. 2. — P. 424–430. — DOI: 10.1214/aos/1176346497. — URL: https://doi.org/10.1214/aos/1176346497.

Thompson R. B., *Olsen C.*, *Dietrich J. R.* Attributes of News About Firms: an Analysis of Firm-Specific News Reported in the Wall Street Journal Index // Journal of Accounting Research. — 1987. — Vol. 25, no. 2.

Vanukuru K. Stock Market Prediction Using Machine Learning. — 11/2018. — DOI: 10.13140/RG.2.2.12300.77448.

Vu T. T., *Chang S.*, *Ha Q. T.*, *Collier N.* An Experiment in Integrating Sentiment Features for Tech Stock Prediction in Twitter. — 2013.

Wilson T., *Hoffmann P.*, *Somasundaran S.*, *Kessler† J.*, *Wiebe†‡ J.*, *Choi Y.*, *Cardie C.*, *Riloff E.*, *Patwardhan S.* OpinionFinder: A system for subjectivity analysis. — 2005.

Woolston C. Psychology journal bans P values // International Weekly Journal of Science. — 2015.

Yosinski J., Clune J., Bengio Y., Lipson H. How transferable are features in deep neural networks? // arXiv. — 2014. — Nov. — arXiv: 1411.1792v1 [cs.LG].

BERT. — URL: http://docs.deeppavlov.ai/en/master/features/pretrained%5C_vectors.html.

Coronavirus: Oil price collapses to lowest level for 18 years. — 03.2020. — URL: https://www.bbc.com/news/business-52089127.

ELMO. — URL: https://github.com/HIT-SCIR/ELMoForManyLangs.

Facebook. — URL: https://www.facebook.com/robots.txt.

Finam. — URL: finam.ru.

Google Colab. — URL: colab.research.google.com.

Howard J. Lesson 10: Deep Learning Part 2, NLP Classification and Translation. — 05.2018. — (дата обр.).

Kaggle. — 2018. — URL: https://www.kaggle.com/c/sentiment-analysis-in-russian.

Papers With Code. — URL: https://paperswithcode.com/task/sentiment-analysis.

Rubtsova. — URL: https://study.mokoron.com/.

The MPQA Opinion Corpus. — URL: https://mpqa.cs.pitt.edu/.

ULMFiT. — URL: https://github.com/ademyanchuk/ulmfit-multilingual.

Yahoo. — URL: https://finance.yahoo.com/quote/%5C%5EDJI/history/.

Акции российских компаний обвалились на открытии Московской биржи // Forbes. — 2020. — март. — URL: https://www.forbes.ru/newsroom/biznes/394641-akcii-rossiyskih-kompaniy-obvalilis-na-otkrytii-moskovskoy-birzhi.

Мосбиржа подготовилась к обвалу на российских торгах. — 03.2020. — URL: https://www.rbc.ru/economics/09/03/2020/5e6699b.

Московская биржа начала расти впервые за неделю. Почему? — 03.2020. — URL: https://www.bbc.com/russian/news-51867794.

Сайт Москвоской Биржи. — URL: https://www.moex.com/ru/ir/interactive-analysis.aspx.

Таблица инфляции. — URL: https://www.macrotrends.net/countries/RUS/russia/inflation-rate-cpi.

Цены на нефть обрушились на 30%. — 03.2020. — URL: https://www.rbc.ru/economics/09/03/2020/5e656d349a79474203e30da2.

Приложения

Приложение 1. Предобработка текстовых данных.

```
def preprocess(text):


    text = (
        re.sub('\s+', ' ',
        re.sub('ххюрл', ' xxurl ',
        re.sub('ххxштг', ' xxhshtg ',
        re.sub('ххакнт', ' xxaccnt ',
        re.sub('xxnewsent xxdash', ' xxnewsent ',
        re.sub('xxexclam(xxexclam[\s])+', ' xxexclam xxmult ',
        re.sub('xxelips(xxelips[\s])+', ' xxelips xxmult ',
        re.sub('xxquest(xxquest[\s])+', ' xxquest xxmult ',
        re.sub('xxnum(xxnum[\s])+', 'xxnum xxmult ',
        re.sub('xxeng(xxeng[\s])+', 'xxeng xxmult ',
```

```
        re.sub('xxdash(xxdash[\s])+', 'xxdash xxmult ',
        re.sub('xxdot xxdot xxdot', 'xxelips',
        re.sub('\d+', 'xxnum',
        re.sub('[\W_]+',' ',
        re.sub('\(', ' xxlbrckt ',
        re.sub('\)', ' xxrbrckt ',
        re.sub('»', ' xxrquotes ',
        re.sub('«', ' xxlquotes ',
        re.sub('\"', ' xxquotes ',
        re.sub('\?', ' xxquest ',
        re.sub('\!', ' xxexclam ',
        re.sub('[\--\-\xa0]', ' xxdash ',
        re.sub('\:', ' xxcolon ',
        re.sub(r'[,;]', ' xxcomma ',
        re.sub(r'\.', ' xxdot ',
        re.sub('\.\.+', ' xxelips ',
        re.sub("\w+[\'\--\xa0]\w+",
            (lambda x: ''.join(re.findall('\w', x.group()))),
        re.sub('[\.\?\!]+[ \n+]+[A-Я\d+-\-\xa0"«(]',
            (lambda x: x.group()[:-2] \
             + ' xxnewsent ' + x.group()[-1]),
        re.sub('[A-Za-z]+', 'xxeng',
        re.sub('\[[a-zA-Z\d]+\]', '',
        re.sub('\<[a-zA-Z\d\_\+\-\@\!\?\.]+\>', '',
        re.sub('https?://[^\"\s>]+', ' xxЮРЛ ',
        re.sub('RT', '',
        re.sub('#[A-Za-z_\d]+[ \:,]?', ' xxХШТГ ',
        re.sub('@[A-Za-z_\d]+[ \:,]?', ' xxАКНТ ',
```

```
                text))))))))).lower()
        ))))))))))))))))))))))).strip()

    return text
```

Приложение 2. Получение данных Твиттера.

```
import pandas as pd
import numpy as np
from tqdm import tqdm_notebook as tqdm
import json
import time
import GetOldTweets3 as got

queries = ['сбербанк', 'сбер', 'втб',
           'внешторгбанк', 'росбанк']
cols = ['date', 'text']
data_dir = 'DATA_DIR'

for query in tqdm(queries):

    start_year = 2020
    years = 5
    months = 12
    for year in tqdm(range(years)):
        for i in tqdm(range(months)):
            for parts in [[1, 15], [15, 1]]:
                since = (f"{start_year-year} - \
```

```
                {months-i\%12}-{parts[0]}",
                until = f"{start_year-year+(months - i)//12 \
                - 1 * (parts[1] == 15 and i == 0)} - \
                {((months-i)\%12)+1*(parts[1]==1) + \
                12 * (i == 0 and parts[1] == 15)}-{parts[1]}")
                tweetCriteria = got.manager.TweetCriteria().\
                    setQuerySearch(query).setSince(since). \
                    setUntil(until).setLang('ru')
                time.sleep(180)
                tweets = got.manager.TweetManager.getTweets(
                    tweetCriteria)
                df = pd.DataFrame(np.zeros((len(tweets), 2)),
                                  columns=cols)
                for j in df:
                    df[j] = list(map(lambda x: getattr(x, j),
                                     tweets))
                name = f'{data_dir}/{query}_{start_year-year}\
                _{months - i}_part_{int(parts[0]!=1)}.csv'
                df.sort_values('date').to_csv(name,
                                              index=False)
                print('Done', query, year, month, parts[0])
```

Приложение 3. Предобработка твитов.

```
import pandas as pd
import numpy as np


from tqdm import tqdm_notebook as tqdm
```

```
import re

import os
from glob import glob

from functions import preprocess

BANK_NAMES = ['Сбербанк', 'ВТБ', 'Росбанк']
BANK_ACCOUNTS = ['sberbank', 'vtb', 'rosbank_info']
ABBRS = ['сбер', 'втб', 'росб']
columns = ['date', 'text']

DATA_DIR = 'Data/'
OUT_DIR = 'Preprocessed/'

banks_data = pd.DataFrame(columns=columns)

for e, bank in tqdm(enumerate(BANK_NAMES)):

    df_bank = pd.DataFrame(columns=columns)

    abbr = ABBRS[e]
    for file in glob(DATA_DIR+abbr+'*'):

        'File name: ' + file.split('/')[1].split('.')[0]
        df = pd.read_csv(file)
        df.sample(2)
```

```
        'Length: ', len(df)

        # 0) Apply preprocess function
        %time df.text = df.text.apply(preprocess)
        df.to = df.to.fillna('').apply(lambda x: x.lower())

        # 1) Check for tweets with bad keywords
        not_relevant_keywords = [
            'помощь', 'помощи', 'помоги', 'помогите',
            'помочь', 'кидайте', 'оплата', 'кидай',
            'донатьте', 'донать', 'переводи',
            'переводите', 'лига', 'лиги', 'переведу',
            'переведи', 'донат', 'донаты', 'дона
            'сбросьте', 'сбрось', 'сброс
            'сбросила', 'молю', 'прошу', 'номер
            'карта', 'карты', 'карту', 'карте']

        not_relevant = df.text.apply(lambda x: np.any(
            [bad_word in x.split() for bad_word \
             in not_relevant_keywords]))
        'Not relevant n: ', not_relevant.sum()

        # 2) Check for short tweets
        is_short = df.text.apply(lambda x: len(x.split()) < 3)
        'Is short n: ', is_short.sum()

        # 3) Check for consequences of digits
        #    in tweet (4 minimum)
```

```
        has_digits_cons = df.text.apply(lambda x: len(
            re.findall('\d\d\d', x)) > 0)
        'Has digits cons n: ', has_digits_cons.sum()

        # 4) Check for the presence of the bank name
        #    (its first 3-4  letters) and for tweets
        #    directed for official banks accounts
        bank_acc = BANK_ACCOUNTS[e]
        without_bank_name = (df.to + ' ' + df.text).apply(
            lambda x: (abbr not in x) and (bank_acc not in x))
        'Without bank name n: ', without_bank_name.sum()

        # 5) Check for tweets with several banks names
        other_banks = np.delete(ABBRS, e)
        has_other_banks = df.text.apply(lambda x: np.any(
            [other_bank in x for other_bank in other_banks]))
        'Has other banks n: ', has_other_banks.sum()

        # 6) Check for tweets directed to other banks
        other_banks_accounts = np.delete(BANK_ACCOUNTS, e)
        to_other_banks = df.to.apply(lambda x: x \
                    in other_banks_accounts)
        'To other banks: ', to_other_banks.sum()

        # 6) Delete all the 'suspicious' tweets
        df = df[~((has_digits_cons) | (is_short) | \
            (not_relevant) | (without_bank_name) | \
            (has_other_banks) | (to_other_banks))]
```

```
        'Total new length: ', len(df)

        df_bank = df_bank.append(df)

    'Length with duplicates: ', len(df_bank)
    df_bank.drop_duplicates('text', inplace=True)
    'Length without duplicates: ', len(df_bank)

    df_bank['bank'] = np.repeat(bank, len(df_bank))
    df_bank = df_bank.loc[:, ['bank']+columns]

    print(df_bank.to.value_counts()[:3])
    df_bank.to_csv(bank+'.csv')

    banks_data = banks_data.append(
        df_bank.reset_index(drop=True))

len(banks_data)
banks_data.drop_duplicates('text', keep=False, inplace=True)
len(banks_data)

banks_data.to_csv(OUT_DIR+'all_data.csv', index=False)
```

Приложение 4. Извлечение текстовых данных из корпуса Рубцовой Ю. для предобучения ULMFiT.

```
import pandas as pd
import numpy as np
from tqdm import tqdm_notebook as tqdm

def sub_null(line):
    if line.count('NULL') > 0:
        line = re.sub('NULL', '"Добрый день"', line)
    return line

with open('/Users/akimtsvigun/Downloads/db.sql', 'r') as f:
    lines = list(f.readlines())

lines = list(map(lambda x: x[31:-2], lines[47:-13]))

lines = [eval(sub_null(i)) for i in tqdm(lines)]
df = pd.concat(([pd.DataFrame(i) for i in tqdm(lines)]),
               axis=0)

df[[3]].rename(columns={3: 'text'}).to_csv('twits_general.gz',
    compression='gzip', index=False)
```

Приложение 5. Предобучение и обучение дальнейший моделей.

Предобучение ULMFiT

```
import numpy as np
import pandas as pd
```

```
import re
import os

from fastai import *
from fastai.text import *
import torch
from tqdm import tqdm_notebook as tqdm
from fastai.callbacks import SaveModelCallback, CSVLogger
from fastai.train import ShowGraph

path = Config.data_path()/'/home/akim/raid_akim/data'
data = load_data(path, 'data_tw.pkl')

learn = language_model_learner(data, AWD_LSTM,
    pretrained=False, drop_mult=1,
    callback_fns=[CSVLogger]).to_fp16()

lr  = 1e-2
lrm = 2.6
lrs = np.array([lr/(lrm**3), lr/(lrm**2), lr/lrm, lr])

learn.fit_one_cycle(1, lrs * 3, moms=(0.8,0.7))

learn.fit_one_cycle(1, lrs, moms=(0.8,0.7))

learn.fit_one_cycle(1, lrs / 2, moms=(0.8,0.7))

learn.fit_one_cycle(1, lrs / 5, moms=(0.8,0.7))
```

```
learn = learn.to_fp32()

learn.save(path/f'm', with_opt=False)
learn.data.vocab.save(path/(f'm' + '.pkl'))
```

Обучение трех моделей ULMFiT

```
df_names = ['df_twitter', 'df_sentiment_news',
    'df_rusenteval']

lr = {0: 3*lrs, 1: lrs, 2: lrs, 3: lrs/2, 4: lrs/3, 5: lrs/5}
lrs_clf = np.array([lr/(lrm**4), lr/(lrm**3),
                    lr/(lrm**2), lr/lrm, lr])

for i in df_names:

    min_freq = 5 if i == 'df_twitter' else 3

    data = TextLMDataBunch.from_csv(path, f'{i}.csv',
        valid_pct=0.1, min_freq=min_freq, text_cols='text')
    data.save(path/f'data_{i[3:]}.pkl')
    data = load_data(path, f'data_{i[3:]}.pkl')
    data.show_batch()

    learn = language_model_learner(data, AWD_LSTM,
            pretrained_fnames=[path/f'm', path/f'm'],
        drop_mult=1)
```

```
learn.freeze_to(-1)

for j in range(15):



    if j in lr.keys():
        lr_use = lr[j]
    else:
        lr_use = lrs / 10

    learn.to_fp16().fit_one_cycle(1, lr_use,
                                  moms=(0.8,0.7))

    learn.unfreeze()

    learn = learn.to_fp32()
    learn.save(path/f'm_{i[3:]}', with_opt=False)
    learn.data.vocab.save(path/f'm_{i[3:]}.pkl')
    learn.save_encoder(path/f'm_{i[3:]}_enc')



data_clf = TextClasDataBunch.from_csv(path, f'{i}.csv',
    valid_pct=0.1, min_freq=min_freq, text_cols='text',
    label_cols='label', vocab=data.vocab)
data_clf.save(path/f'data_clf_{prefix}.pkl')
data_clf.show_batch(2)
```

```
    classifier = text_classifier_learner(data, AWD_LSTM,
        pretrained=False)
    classifier.load_encoder(path/f'm_{i[3:]}_enc')

    classifier.freeze_to(-1)
    classifier.fit_one_cycle(1, lrs_clf * 3, moms=(0.8,0.7))

    classifier.save(path/f'clf_{i[3:]}', with_opt=False)
    classifier.data.vocab.save(path/(f'clf_{i[3:]}' + '.pkl'))

    classifier.freeze_to(-2)
    classifier.fit_one_cycle(1, lrs_clf * 3, moms=(0.8,0.7))

    classifier.save(path/f'clf_{i[3:]}', with_opt=False)
    classifier.data.vocab.save(path/(f'clf_{i[3:]}' + '.pkl'))

    classifier.unfreeze()
    classifier.fit_one_cycle(3, lrs_clf, moms=(0.8,0.7))

    classifier.save(path/f'clf_{i[3:]}', with_opt=False)
    classifier.data.vocab.save(path/(f'clf_{i[3:]}' + '.pkl'))

    classifier.unfreeze()
    classifier.fit_one_cycle(3, lrs_clf / 5, moms=(0.8,0.7))

    classifier.save(path/f'clf_{i[3:]}', with_opt=False)
    classifier.data.vocab.save(path/(f'clf_{i[3:]}' + '.pkl'))
```

Приложение 6. Создание признаков.

```
import pandas as pd
import numpy as np

for bank in ['Сбербанк', 'ВТБ', 'Росбанк']:
        df = pd.read_csv(f'stocks/{bank}_min.txt')
        df_last = df.groupby('date').agg('last').reset_index()
        df.groupby('date').agg('first').reset_index(
            inplace=True)

        df['datetime'] = df.date.astype(str).apply(lambda x: \
            x[:4]+'/'+x[4:6]+'/'+x[6:]).apply(pd.to_datetime)
        df.drop(columns=df.columns[:4], inplace=True)

    dt_cols = ['year', 'month', 'day', 'dayofweek']
        for i in dt_cols:
            df[i] = df.datetime.apply(lambda x: getattr(x, i))

        df['unique_day'] = df.datetime.apply(lambda x: \
            x.date()).astype('str')
        df = df.drop(columns='datetime')

        df.set_index('unique_day', inplace=True)
        df[dt_cols] = df[dt_cols].astype(int)

    df_last['unique_day'] = df['unique_day']
        df = df.merge(df_last[['close', 'unique_day']].rename(
```

```
                columns={'close': 'last'}), on='unique_day')

    variance = pd.read_csv(f'{bank}_variance.csv')
    df = df.merge(variance, on='unique_day')

    inflation = pd.read_csv('inflation.csv')
    df = df.merge(inflation, how='left', on='unique_day')
    df.close /= df.cum_inflation
    df.drop(columns='cum_inflation', inplace=True)

    diff = pd.read_csv(f'{bank}_diff.csv')
    df = df.merge(diff, on='unique_day', how='left')
    df.dropna(axis=0, inplace=True)

    n_prev = 3
    for i in range(1, n_prev+1):

        df[f'prev_first_{i}'] = np.roll(df.close, i)
        df[f'prev_last_{i}'] = np.roll(df.last, i)
        df[f'prev_first_{i}_sq'] = np.roll(df.close, i)**2
        df[f'prev_last_{i}_sq'] = np.roll(df.last, i)**2
        df[f'prev_{i}_var'] = np.roll(df['diff'], i)

    df.to_csv(f'{bank}_dataframe.csv', index=0)
```

Приложение 7. Отбор признаков для модели без признаков тональности.

```
def golden_section_search(point, vector, f, n_gold_sect_iter):

        if np.all(vector == 0):
            return point, 0

        phi = (1 + 5**(1/2)) / 2
        betas_interval_point = n_gold_sect_iter // 3
        points = [point - betas_interval_point * vector,
                  point + betas_interval_point * vector]
        values = [f(*point) for point in points]
        a, b = points if values[0] < values[1] \
               else points[::-1]

        for i in range(n_gold_sect_iter):

            part = (b - a) / phi
            x1 = b - part
            x2 = a + part

            if f(*x1) >= f(*x2):
                a = x1
            else:
                b = x2

        new_point = (a + b) / 2
```

```
        beta = ((new_point - point) / vector)[np.argwhere(
            vector).ravel()[0]]
        return (a + b) / 2, beta

def mse(y_true, y_pred): return np.mean((y_true - y_pred)**2)

def loss(LR, param, X_train, X_test, y_train, y_test):

    loss_value = 0

    for board in [1150, 1006, 862, 718, 574]:

        lasso = LR(param)
        lasso.fit(X_train, y_train)
        loss_value += mse(y_test, lasso.predict(X_test))

    return loss_value / 5
```

Приложение 8. P-значение.

```
import numpy as np
from scipy import stats

def p_value(beta, se, n):

    value = abs(beta / se)
    p_val = (1 - stats.norm.cdf(value)) * 2
    return p_val
```

Приложение 9. F-тест.

```
def mse(x, y): return ((x - y)**2).mean()
def betas(x, y): return np.linalg.inv(x.T @ x) @ x.T @ y

def f_test(x, x_add, y):

    x_add = x_add.reshape(len(x_add), -1)
    x_stack = np.c_[x, x_add, np.ones(len(x))]
    x = np.c_[x, np.ones(len(x))]

    mse_r = mse(y, x @ betas(x, y))
    mse_ur = mse(y, x_stack @ betas(x_stack, y))

    n = len(x)
    q = x_add.shape[1]
    k_ur = x_stack.shape[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)

    print(mse_r, mse_ur, stat)

    return p_val
```

Приложение 10. Тест Диболда-Мариано.

За основу реализации теста Диболда-Мариано в Python взят код, принадлежащий John Tsang URL.

```
def dm_test(actual_lst, pred1_lst,        pred2_lst,
            h = 1, crit="MSE", power = 2):
    # Routine for checking errors
    def error_check():
        rt = 0
        msg = ""
        # Check if h is an integer
        if (not isinstance(h, int)):
            rt = -1
            msg = "h must be an integer."
            return (rt, msg)
        # Check the range of h
        if (h < 1):
            rt = -1
            msg = "h is too small."
            return (rt, msg)
        len_act = len(actual_lst)
        len_p1  = len(pred1_lst)
        len_p2  = len(pred2_lst)
        # Check if lengths of actual values and
        # predicted values are equal
        if (len_act != len_p1 or len_p1 != len_p2 \
                    or len_act != len_p2):
            rt = -1
```

```
            msg = "Lengths must match."
            return (rt, msg)
        # Check range of h
        if (h >= len_act):
            rt = -1
            msg = "h is too large."
            return (rt, msg)
        # Check if criterion supported
        if (crit != "MSE" and crit != "MAPE" and \
                        crit != "MAD" and crit != "poly"):
            rt = -1
            msg = "The criterion is not supported."
            return (rt, msg)
        # Check if every value of the input
        # lists are numerical values
        from re import compile as re_compile
        comp = re_compile("^\d+?\.\d+?$")
        def compiled_regex(s):
            """ Returns True is string is a number. """
            if comp.match(s) is None:
                return s.isdigit()
            return True
        for actual, pred1, pred2 in zip(actual_lst,
            pred1_lst, pred2_lst):
            actual_ok = compiled_regex(str(abs(actual)))
            pred1_ok = compiled_regex(str(abs(pred1)))
            pred2_ok = compiled_regex(str(abs(pred2)))
            if (not np.all([actual_ok, pred1_ok, pred2_ok])):
```

```
                msg = "Found a non-numeric element."
                rt = -1
                return (rt, msg)
        return (rt, msg)

# Error check
error_code = error_check()
# Raise error if cannot pass error check
if (error_code[0] == -1):
    raise SyntaxError(error_code[1])
    return
# Import libraries
from scipy.stats import t
import collections
import pandas as pd
import numpy as np

# Initialise lists
e1_lst = []
e2_lst = []
d_lst  = []

# convert every value of the lists into real values
actual_lst = list(map(lambda x: float, actual_lst))
pred1_lst = list(map(lambda x: float, pred1_lst))
pred2_lst = list(map(lambda x: float, pred2_lst))

# Length of lists (as real numbers)
```

```
    T = float(len(actual_lst))
                zipped = zip(actual_lst,pred1_lst,pred2_lst)

    # construct d according to crit
    if (crit == "MSE"):
        for actual, p1, p2 in zipped:
            e1_lst.append((actual - p1)**2)
            e2_lst.append((actual - p2)**2)
        for e1, e2 in zip(e1_lst, e2_lst):
            d_lst.append(e1 - e2)
    elif (crit == "MAD"):
        for actual,p1,p2 in zipped:
            e1_lst.append(abs(actual - p1))
            e2_lst.append(abs(actual - p2))
        for e1, e2 in zip(e1_lst, e2_lst):
            d_lst.append(e1 - e2)
    elif (crit == "MAPE"):
        for actual,p1,p2 in :
            e1_lst.append(abs((actual - p1)/actual))
            e2_lst.append(abs((actual - p2)/actual))
        for e1, e2 in zip(e1_lst, e2_lst):
            d_lst.append(e1 - e2)
    elif (crit == "poly"):
        for actual,p1,p2 in zipped:
            e1_lst.append(((actual - p1))**(power))
            e2_lst.append(((actual - p2))**(power))
        for e1, e2 in zip(e1_lst, e2_lst):
            d_lst.append(e1 - e2)
```

```
    # Mean of d
    mean_d = pd.Series(d_lst).mean()

    # Find autocovariance and construct DM test statistics
    def autocovariance(Xi, N, k, Xs):
        autoCov = 0
        T = float(N)
        for i in np.arange(0, N-k):
              autoCov += ((Xi[i+k])-Xs)*(Xi[i]-Xs)
        return (1/(T-1))*autoCov
    gamma = []
    for lag in range(0,h):
        gamma.append(autocovariance(d_lst,
                       len(d_lst),lag,mean_d))
    V_d = (gamma[0] + 2*sum(gamma[1:]))/T + 1e-23
    DM_stat=V_d**(-0.5)*mean_d
    harvey_adj=((T+1-2*h+h*(h-1)/T)/T)**(0.5)
    DM_stat = harvey_adj*DM_stat
    # Find p-value
    p_value = 2*t.cdf(-abs(DM_stat), df = T-1)

    # Construct named tuple for return
    dm_return = collections.namedtuple(
                                'dm_return', 'DM p_value')

    rt = dm_return(DM = DM_stat, p_value = p_value)

    return rt
```