Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра информатики

Зернов Алексей Викторович

Разработка системы автоматического анализа новостных публикаций на финансовом рынке

Бакалаврская работа

Научный руководитель:
к. ф.-м. н., доцент Григорьев Д. А.

Рецензент:
д. т. н., декан Мусаев А. А.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY
Faculty of Mathematics and Mechanics

Computer Science Department

Zernov Alexey Viktorovich

Development of the automatic analysis system of a financial market's news publications

Bachelor's Thesis

Scientific supervisor:
Sc. C., associate professor Grigoryev D. A.

Reviewer:
Sc. D., dean Musaev A. A.

Saint-Petersburg
2017

Оглавление

Введение

Не смотря на то, что с каждым годом происходит увеличение доли цифровой информации по отношению к бумажной, все равно остается проблема работы с этими данными. Дело в том, что большинство такой информации является неструктурированной, а следовательно на ее обработку требуется достаточно много времени и человеческих ресурсов. Целью данной работы является написание программы, позволяющей уменьшенить объем временных затрат на изучение большого потока новостных публикаций в тех случаях, когда необходимо оценить изменение стоимости акций определенной компании по связанным с ней новостям.

В работе будут рассмотрены основные определения, связанные с финансовым рынком (Раздел 1); базовая теория, касаемая интеллектуального анализа текста (Раздел 2); существующие решения (Раздел 3) и представлен результат работы в виде программы, осуществляющей анализ новостных публикаций с возможностью последующего предсказания изменения стоимости акций (Раздел 4).

1. Финансовый рынок

В данном разделе будет представлен краткий обзор основных терминов, связанных с самим финансовым рынком, его структурой и основными участниками. Более подробная информация может быть получена в книге [7].

1.1. Определение

В более общем виде **финансовый рынок** — совокупность экономических связей его участников, касающихся создания, поддержания и обращения капитала. Финансовый рынок является довольно абстрактным термином, и под ним часто подразумеваются более конкретные: рынок купонных и бескупонных облигаций, рынок акций (или фондовый рынок) или валютный рынок. Не смотря на выделение составляющих, каждая из них является частью единого механизма, в котором финансы перемещаются между каждым из конкретных рынков.

Каждый из финансовых рынков является рынком посредников между начальными владельцами финансов и их конечными пользователями. Если рынок основывается на финансах как на капитале, он называется фондовым рынком, и именно в этой роли выступает как составная часть всего финансового рынка.

В России финансовые рынки имеют следующие критерии, влияющие на их деятельность:

- Инвестиции в экономику страны
- Международные рынки, влиние тенденций глобализации
- Современные компьютерные технологии
- Уровень комьютерной и информационной развитости участников рынков

1.2. Структура

Финансовый рынок может быть:

- Первичным или вторичным
- Организованным или неорганизованным
- Биржевым или внебиржевым
- Традиционным или компьютеризированным
- Кассовым или срочным

Первичный рынок обеспечивает выход ценных бумаг в оборот, это своеобразное «производство» ценных бумаг. На **вторичном рынке** в обороте находятся уже выпущенные ранее ценные бумаги. Вторичный рынок представляет из себя совокупность всех операций с данными ценными бумагами, в результате которых они переходят от одних владельцев к другим.

Организованный рынок отличается от **неорганизованного рынка** тем, что в первом имеются единые для всех участников рынка правила, за соблюдением которых следят организаторы. В неорганизованном рынке соблюдение единых правил для всех участников рынка не гарантируется.

Биржевой рынок — такой рынок, на котором в качестве инструмента торговли используется аукцион. Руководителем же является некоторый специалист, например, NYSE[1] или AMEX[2]. На **внебиржевых рынках** торги организуются при помощи электронных систем.

Срочный рынок чаще всего подразумевает отложенное исполнение сделки, в отличие от **кассового рынка**, когда сделки исполняются сразу. Обычно традиционные ценные бумаги (акции, облигации) идут в оборот на кассовых рынках, а контракты на производные инструменты рынка ценных бумаг — на срочных.

[1]New York Stock Exchange — Нью-Йоркская фондовая биржа

[2]American Stock Exchange - Американская фондовая биржа

1.3. Участники

Участники рынка ценных бумаг — это физические лица или компании, которые продают или приобретают ценные бумаги, обеспечивают их оборот или расчеты по ним.

Основными участниками рынка выступают **эмитенты**, выпускающие акции или облигации, с помощью которых привлекают финансирование, а также размещающие свободные на данный момент денежные средства. Эмитентами могут быть: государство, субъекты государства или коммерческие предприятия. Целью эмитентов на первичном рынке является размещение запланированного транша по максимальной цене.

Инвестор — лицо, заинтересованное во вложении капитала в ценные бумаги. Целью инвесторов является как можно более выгодная покупка ценных бумаг максимально перспективных компаний.

2. Интеллектуальный анализ текста

В настоящее время можно заметить увеличение роли компьютеров в жизни каждого человека. Инорфмация хранится преимущественно в цифровом виде, что значительно упрощает поиск или работу с ней. Но не смотря на это, многие данные все равно остаются довольно трудными для анализа, не смотря на оцифрованный вид, из-за чего можно подразделить из на следующие формы:

- Структурированные данные
- Частично структурированные данные
- Неструктурированные данные

Хорошим примером **структурированных данных** могут являться базы данных. **Частично структурированные данные** — это электронные письма, разнообразные файлы на языках разметок (HTML, XML и другие).

Если работа со структурированными или частично структурированными данными достаточно детерминированная, то **неструктурированные данные** представляют наибольший интерес в этом вопросе. Около 80% корпоративных данных находится именно в неструктурированном формате, в котором сложно проводить поиск или извлекать необходимую информацию. Для этого нужны специфические методы и алгоритмы обработки. И поскольку самая популярная форма хранения информации — это текст, интеллектуальный анализ текста (text mining) является более важным процессом, нежели интеллектуальный анализ данных (data mining).

Интеллектуальный анализ текста стоит на пересечении дисциплин и включает в себя: обработку web-данных, информационный поиск, компьютерную лингвистику и обработку естественного языка.

2.1. Процесс интеллектуального анализа текста

Концепция интеллектуального анализа текста представлена в [6]. В интеллектуальном анализе текста можно выделить два основных этапа (Рис. 1):

- Фильтрация текста
- Извлечение знаний

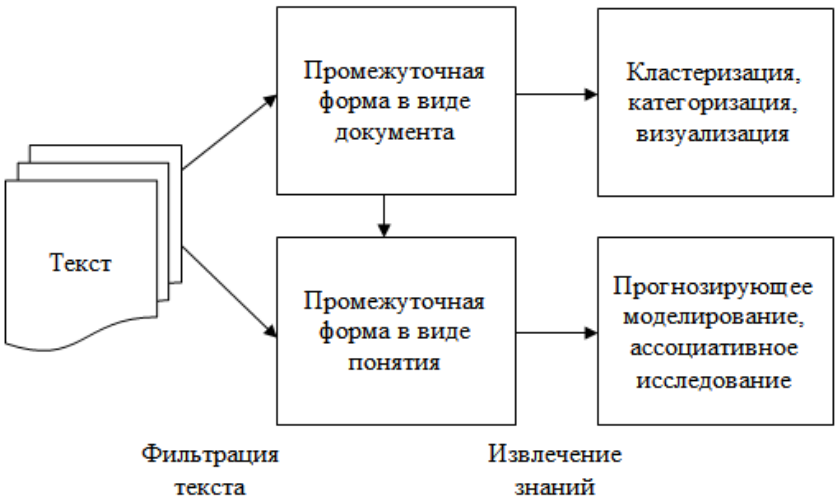

Рис. 1: Общий процесс интеллектуального анализа текста

Фильтрация (или очистка) преобразует исходный текстовый документ в некоторое промежуточное представление. **Извлечение знаний**, в свою очередь, получает полезную информацию (знания) или некоторые шаблоны уже из промежуточного представления. Промежуточное представление может быть как структурированным, так и частично структурированным. Также оно может быть как новым текстовым документом, так понятием, в котором составляющие являются данными или наборами данных из какой-либо предметной области.

Анализ промежуточного представления в виде документов выдает образцы и связи между всеми документами.

Анализ промежуточного представления в виде понятий выдает образцы и связи между объектами или другими понятиями.

Примеры задач анализа промежуточного представления в виде документов: *кластеризация*, *визуализация* и *категоризация документов*; примеры задач анализа промежуточного представления в виде понятий: *прогнозирующее моделирование* и *ассоциативное исследование*.

Промежуточное представление в виде документа может быть преобразовано в промежуточное представление в виде понятия путем выделения релевантной информации, которая относится к необходимым объектам из какой-либо предметной области. Отсюда вытекает то, что промежуточное представление чаще не зависит от конкретное предметной области. К примеру, новостные потоки при фильтрации текста преобразуются в промежуточные представления в виде документов, соответствующим определенным статьям. Затем, в зависимости от поставленных задач визуализации или навигации, каждый документ (статья) проходит обработку знаний. Для извлечения же знаний в определенной предметной области промежуточное представление в виде документа может быть преобразовано в промежуточное представление в виде понятия в соответствии с необходимыми требованиями. К примеру, можно извлечь информацию, касающуюся определенного товара или услуги из промежуточного представления в виде документа и сформировать базу данных товаров или услуг для предоставления знаний о них.

2.1.1. Предварительная обработка текста

Предварительная обработка включает в себя:

1. Токенизацию
2. Удаление «стоп-слов»
3. Определение происхождения слов

Токенизация Сначала текст разделяется на отдельные слова, освобождаясь от пробелов и знаков препинания.

Удаление «стоп-слов» На этом этапе происходит избавление от «ненужных» конструкций текста. Это могут быть HTML или XML теги, предлоги, артикли и прочее.

Происхождения слов Представляет из себя выявление корней определенных слов. Порой эта обработка бывает более грубой и выделяются, например, только своеобразные основы (обрубаются окончания или приставки).

2.1.2. Преобразование текста

Текстовый документ состоит из слов и информации об их происхождении. Два основных подхода представления документа: «мешок слов» («bag-of-words») и векторные пространства слов.

2.1.3. Поиск признаков

Под признаками можно понимать переменные. То есть в результате этого шага отбирается подмножество наиболее значимых признаков для их дальнейшего применения при построении моделей. Убираются, например, признаки, которые избыточны или не несут никакой информации.

2.1.4. Методы анализа текста

На данном шаге начинается построение модели с использованием разных методов, таких как кластеризация, классификация, информационный поиск и других. Данные методы распознавания данных также подходят и для интеллектуального анализа текста.

2.1.5. Интерпретация и оценка

На последнем шаге (в зависимости от того, что требуется) проводится анализ результатов.

2.2. Области применения интеллектуального анализа текста

Как уже упоминалось выше, интеллектуальный анализ текста стоит на пересечении разных дисциплин и включает в себя: извлечение информации, информационный поиск, обработку естественного языка и интеллектуальный анализ данных.

2.2.1. Извлечение информации

В процессе извлечения информации автоматически извлекается структурированная информация из неструктурированных данных. С помощью распознавания образов данная система определяет, например, где имена людей, где названия компаний, а где местоположение. То есть в документах происходит поиск предопределенных последовательностей. Подобное решение позволяет получить элементы, подходящие для использования в базах данных для дальнейшего хранения, анализа или обработки.

2.2.2. Информационный поиск

В данной задаче используются методы, используемые для хранения, представления и доступа к информации, которая преимущественно представлена в виде текстовых документов (а также новостных лент или книг), которые могут быть получены по запросу пользователя. Это своего рода расширение поиска по документам, позволяющее сужать набор документов, имеющих отношение к запросу пользователя. Эти системы значительно сокращают время, необходимое для поиска необходимой информации. Наиболее известными системами информационного поиска являются поисковые системы Google.

2.2.3. Обработка естественного языка

Данная задача представляет из себя самую активную проблему в области искусственного интеллекта. Цель: исследовать естественный

язык так, чтобы у компьютеров была возможность понимать языки, подобные тем, что используют для общения люди. Обработка естественного языка включает в себя распознавание и генерацию, которые отвечают за такие способности компьютера как «читать» и «говорить» на естественном языке соответственно. Подобные системы включают в себя проверку грамматики, лексические, синтаксические и семантические анализаторы.

2.2.4. Интеллектуальный анализ данных

Данные задачи относятся к поиску знаний или релевантной информации в большом объеме данных. Система пытается обнаружить правила (статистически) и образцы (автоматически) от данных. Подобные системы имеют возможность предсказания, основываясь на «опыте», полученном в результате исследования.

3. Обзор существующих инструментов

В данном разделе будут рассмотрены основные инструменты, представленные в виде библиотек или отдельных сервисов. Внимание уделено в основном инструментам, работающим с русским языком.

3.1. Natural Language Toolkit

NLTK[5] является пакетом библиотек и программ для разработки программ на Python, работающих с естественным языком. Сопровождается обширной документацией, а также книгой[3], объясняющей основные концепции проблем, для решения которых предназначен данный пакет.

Данный пакет подходит для таких областей как компьютерная лингвистика, эмпирическая лингвистика, когнитивистика, искусственный интеллект, информационный поиск и машинное обучение. NLTK используется преимущественно в качестве учебного пособия, индивидуального обучения или прототипирования и создания систем, ориентированных на научно-исследовательскую деятельность.

NLTK — свободное программное обеспечение, то есть доступное бесплатно.

3.2. Pymorphy2

Pymorphy2[4] написан на языке Python и имеет следующие возможности:

- Приведение слова к нормальной форме
- Ставить слово в нужную форму
- Возвращать грамматическую информацию о слове

Распространяется pymorphy2 под лицензией MIT[4], если используется в научной работе.

[3] `http://www.nltk.org/book/`

[4] `https://opensource.org/licenses/MIT`

3.3. Томита-парсер

Томита-парсер[5] способен извлекать структурированные данные из текстов на естественном языке. Как и почти во всех инструментах, рассматриваемых в данном разделе, Томита-парсер ориентирован преимущественно на русскоязычные тексты. В нем используются контекстно-свободные грамматики и словари ключевых слов. Код проекта[6] (написан на C и C++) находится в свободном доступе.

3.4. Яндекс.Спеллер

Яндекс.Спеллер[7] выполняет задачу проверки орфографии в текстах на английском, русском и украинском языках. Для этого используется орфографический словарь. К тому же, предоставлен набор API методов (для JavaScript) для реализации данной проверки разработчиками сайтов или приложений.

3.5. OntosMiner

OntosMiner[8] является решением компании Eventos[9], занимающейся в большей степени разработкой продуктов в области лингвистического анализа текстовой информации, кластеризацией и классификацией информации. Конкретно OntosMiner является целой комплексной системой, дающей возможность распознавания связей между сущностями в текстах на естественной языке. Также, она позволяет определять общую тональность текста.

[5] https://tech.yandex.ru/tomita/
[6] https://github.com/yandex/tomita-parser/
[7] https://tech.yandex.ru/speller/
[8] http://my-eventos.com/solution/ontosminer/
[9] http://my-eventos.com/solution/ontosminer/

4. Программная часть

В результате работы была написана программа[10], позволяющая автоматически анализировать новостные публикации сайта `mfd.ru`.

4.1. Описание

Программа способна выполнять следующие функции:

- Загружать заданное количество последних новостных публикаций определенной компании
- Загружать данные о котировках определенной компании за заданный промежуток времени
- Формировать и обучать рекурентную нейронную сеть по заданным данным
- Предсказывать изменение цены по заданной новостной публикации

На вход программы подается название компании, выступающей в роли эмитента, количество новостей, начальная и конечные даты, в течение которых необходимо получить изменение изменения цен. В результате работы программы получаются следующие файлы:

- `news/company.csv` — скаченные новости в формате csv с двумя колонками: дата и текст
- `stocks/company.csv` — скаченные котировки в формате csv с двумя колонками: дата и стоимость акций
- `stems/company.csv` — обработанные новости в формате, аналогичном `news/company.csv`
- `connections/company.csv` — соединенные новости и котировки в формате csv с тремя колонками: дата, обработанный текст и изменение акции (положительное или отрицательное)

[10] `https://github.com/Zernov/diploma/tree/master/src`

4.2. Используемые инструменты

Выбор инструментов основывался на тех задачах, которые нужно было решать в процессе написания программы. Исходя из поставленной задачи можно выделить следующие подзадачи:

- Загрузка данных с интернет-ресурсов, для чего необходима работа с web-запросами
- Преобразование содержимого web-страниц, для чего нужны инструменты преобразования содержимого HTML-файлов
- Преобразование текстовых документов в более пригодный для обучения вид
- Обучение рекурентной нейронной сети, для чего необходимы соответствующие инструменты

В связи с подзадачами был выбран язык программирования Python версии 3.6.0 и библиотеки `urllib`[11] (работа с web-запросами) версии 1.21.1, `bs4`[12] (обработка html-файлов) версии 4.6.0, `nltk`[13][5] (преобразование текстовых документов) версии 3.2.2 и `keras`[14][1] (работа с рекурентными нейронными сетями) версии 2.0.3. Возможность написания всех программных модулей на одном языке упрощает разработку и поддержку, что было еще одним преимуществом.

4.3. Структура программы

Всего в программе присутствует 6 основных файлов (модулей), каждый из которых отвечает за свою часть работы (Рис. 2).

- `news_getter.py` отвечает за скачивание новостей с сайта `mfd.ru`, за запись новостей в файл и за чтение новостей из файла

[11] `https://docs.python.org/3/library/urllib.html`
[12] `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`
[13] `http://www.nltk.org/`
[14] `https://keras.io`

- `stock_getter.py` отвечает за загрузку котировок с сайта `finam.ru`, за запись котировок в файл и за чтение котировок из файла
- `connector.py` является вспомогательным модулем, ответственным за объединение новостей и подсчет изменения котировок за соответствующие даты
- `stemmer.py` выполняет небольшую задачу по выделению основ слов, чтобы избежать излишнего увеличения числа переменных при обучении
- И наконец, все перечисленные выше файлы подключатся в основной (`main.py`), который выполняет последовательно необходимые действия и имеет два метода: обучение нейронной сети по данным и предсказание изменений по заданному набору новостей

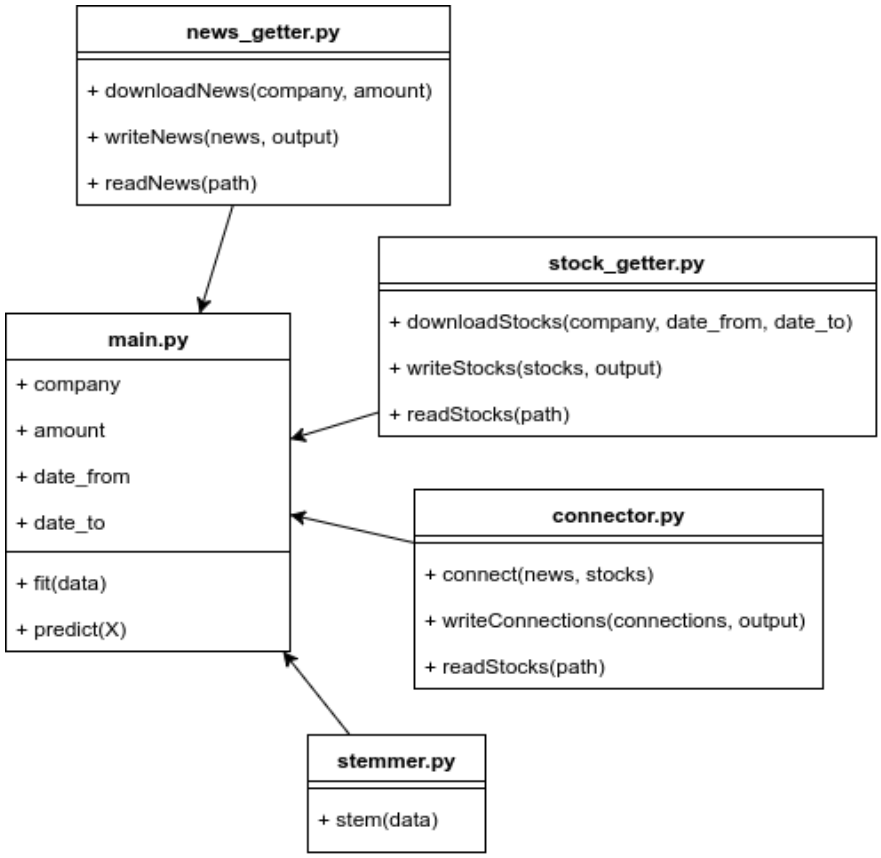

Рис. 2: Структура программы

4.4. Работа программы

Работу программы можно разбить на два основных этапа: предварительная обработка и построение модели. Во время предварительной обработки происходит загрузка и преобразование данных (включая стемминг и удаление «стоп-слов»). Во время построения модели выделяются и строятся требуемые слои рекурентной нейронной сети.

4.4.1. Предварительная обработка

Изначально необходимо получить требуемые данные: тексты новостей и котировок. В случае добавления и/или изменения новостных источников или сайтов, позволяющих загрузить данные о котировках, затрагивается только единственный метод в соответствующем модуле.

Экспорт новостей В случае экспорта новостей информационным источником выступал сайт `mfd.ru`. В методе `downloadNews` (Приложение А), который находится в модуле `news_getter.py`, имеются два входных параметра: название компании и количество требуемых новостей. Название компании преобразуется в идентификатор эмитента соответствующей компании на сайте `mfd.ru`, после чего строятся адреса последних новостей в требуемом количестве, и начинается загрузка. Подобное решение было принято в связи с тем, что новостная лента может обновляться во время загрузки большого количества данных, требуемых для обучения, и в результате загрузки мы получим дублирование некоторых новостей. Факт долгой загрузки большого объема данных так же создает проблему возможных сбоев при загрузке. Она была решена отловом различных HTTP-ошибок с остановкой запросов на некоторое время и последующим возобновлением загрузки. После загрузки новости к результатам добавлялась очередная пара, состоящая из даты и текста новости. Результат экспорта возвращался в основную программу для дальнейших действий с ним (записи в файл или непосредственной обработки).

Экспорт котировок В случае экспорта котировок данные получались с сайта `finam.ru`, на котором имеется возможность с помощью HTTP-запроса получить информацию по котировкам определенной компании. Метод, отвечающий за это, называется `downloadStocks` (Приложение B) и находится в модуле `stock_getter.py`. На вход он принимает три параметра: название компании и границы дат, между которыми необходимо получить информацию. Название компании позволяет определить идентификатор эмитента соответствующей компании и ее код — параметры в адресе запроса. В данной работе единицей измерения интервала между стоимостями котировок являлся один день. Из нескольких цен, предоставленных в результате экспорта (цена на момент открытия торгов, цена на момент закрытия торгов, максимальная цена за время торгов и минимальная цена за время торгов) бралась единственная — цена на момент открытия торгов. Далее именно разница между ценами на момент открытия торгов в два разных дня станет оценкой новостей, опубликованных за этот промежуток времени. Результатом экспорта является набор пар, состоящих из даты и цены на момент открытия торгов в этот день, и он возвращается в основную программу для дальнейших действий (записи в файл или непосредственной обработки).

Преобразование данных Преобразование данных тоже можно разбить на две части: обработка текста и соединение новостей с соответствующими котировками по датам. Первую часть выполняет метод `stem` (Приложение C) модуля `stemmer.py`, принимающий на вход необработанные новости. При обработке текста новости в первую очередь убираются цифры, знаки пунктуации и латинские буквы (в связи с их небольшим количеством). Затем каждое слово в тексте проходит операцию стемминга, то есть выделения основы слова для избавления от чрезмерного дублирования похожих слов в словаре. В этом же методе происходит «склейка» новостей одного дня в единую новость этого же дня. Результатом обработки текста является набор, содержащий даты с соответствующими «склеенными» новостями, содержащими лишь ос-

новы слов без знаков пунктуации, цифр и латинских букв. После этого этапа происходит создание подходящего набора данных для обучения, содержащего новости и соответствующие им оценки (в простейшем случае 0, если последовали отрицательные изменения и 1, если последовали положительные изменения). За эту задачу отвечает метод `connect` (Приложение D) в соответствующем модуле `connector.py`, принимающий на вход новости и котировки. Изначально выделяется пересечение множеств дат из обоих наборов данных (количество этих дат и определяет размер набора данных для обучения). В случае отсутствия информации о котировках в день, в который была опубликована новость, она «склеивается» с предыдущими (как в обработке текста). Затем для каждой новости вычисляется ее оценка: 0, если цена акций к следующей новости упала, и 1 в противном случае. Результатом соединения является набор троек: дата, новость, оценка. После отработки метода, его результат возвращается в основную программу, где текст проходит предварительную обработку с помощью `Tokenizer` — класса, позволяющего индексировать все слова данного множества текстов, превратив их тем самым в наборы чисел, каждое из которых указывает на соответствующее слово в словаре.

4.4.2. Построение модели

Как уже было сказано ранее, на основе полученных данных программа обучает рекурентную нейронную сеть (RNN). Рекурентая нейронная сеть отличается от обычной наличием памяти. Однако в первоначальной ее модели память имеет небольшой объем — несколько элементов. В связи с этим было принято решение использовать метод LSTM [3], имеющий более объемную память и более высокую скорость обучения по сравнению с другими моделями рекурентных нейронных сетей. Как видно из кода (Приложение E), в модели присутствуют слои: Embending, LSTM, Dropout, Dense и Activation (Рис. 3). Рассмотрим подробнее некоторые из них.

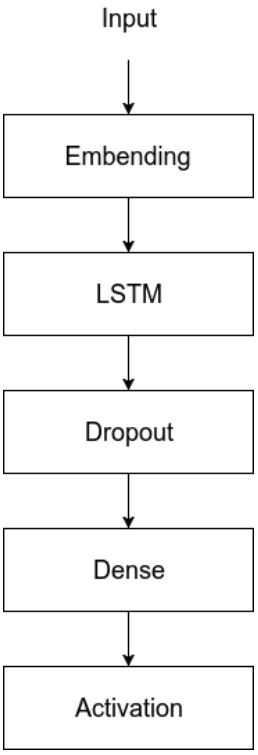

Рис. 3: Слои модели рекурентной нейронной сети

Embending Этот слой преобразует индексы слов в вектора заданной размерности. Задача этого слоя — придать семантическое значение индексам, чтобы похожие слова имели близкие векторы.

LSTM Схема работы LSTM подробно описана в работе [3].

Dropout Схема работы Dropout подробно описана в работе [2]. Задачей этого метода является предотвращение переобучения: на каждом шаге обнуляется pn компонент входного вектора, где p — параметр Dropout, а n — длина вектора.

Dense В данном слое задаются параметры регуляризации, позволяющие уменьшить риск переобучения.

Activation В конце вычисляется активационная сигмоидальная функция, принимающая значение из полуинтервала $[0; 1)$, интерпретируемая как вероятность изменения акций в положительную сторону.

Заключение

Будет добавлено после финальных экспериментов

Список литературы

[1] Chollet François et al. Keras. — `https://github.com/fchollet/keras`. — 2015.

[2] Dropout: a simple way to prevent neural networks from overfitting. / Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky et al. // Journal of Machine Learning Research. — 2014. — Vol. 15, no. 1. — P. 1929–1958.

[3] Hochreiter Sepp, Schmidhuber Jürgen. Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.

[4] Korobov Mikhail. Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts / Ed. by Mikhail Yu. Khachay, Natalia Konstantinova, Alexander Panchenko et al. — Springer International Publishing, 2015. — Vol. 542 of Communications in Computer and Information Science. — P. 320–332.

[5] Loper Edward, Bird Steven. NLTK: The Natural Language Toolkit // Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1. — ETMTNLP '02. — Stroudsburg, PA, USA : Association for Computational Linguistics, 2002. — P. 63–70.

[6] Sumathy K. L., Chidambaram M. Article: Text Mining: Concepts, Applications, Tools and Issues — An Overview // International Journal of Computer Applications. — 2013. — October. — Vol. 80, no. 4. — P. 29–32. — Full text available.

[7] V.P. Romanov. Information technology modeling of financial markets - (Applied Information Technology) / Informatsionnye tekhnologii modelirovaniya finansovykh rynkov - ("Prikladnye informatsionnye tekhnologii"). — Finansy i statistika, 2010. — ISBN: 5279034444.

A. Исходный код метода downloadNews

```
def downloadNews(company, amount):
 domain = 'http://mfd.ru'
 news_dates = []
 news = []
 news_count = 0
 if company == 'sberbank':
  company = '1'
 elif company == 'gazprom':
  company = '3'
 amount = int(amount)
 trs = getTrs(company, amount)
 total = len(trs)
 current = 0
 while current < total:
  try:
   td = trs[current].findAll('td')
   temp_date = td[0].getText().split(',')[0].strip()
   if temp_date == 'сегодня':
    today = datetime.date.today()
    item_date = today.strftime('%d/%m/%y')
   elif temp_date == 'вчера':
    yesterday = datetime.date.today() - datetime.timedelta(1)
    item_date = yesterday.strftime('%d/%m/%y')
   else:
    temp_date_split = temp_date.split('.')
    item_date = '{}/{}/{}'.format(str(temp_date_split[0]),
     str(temp_date_split[1]), str(temp_date_split[2][2:]))
    item_url = domain + td[1].find('a').get('href')
    item_bs = BeautifulSoup(urlopen(item_url), 'html.parser')
    item_content = item_bs.find('div', { 'class' : 'm-content' })
    item_data = item_content.findAll('p')
    item_string = ''
    for j in range(1, len(item_data) - 2):
     item_string += item_data[j].getText() + '␣'
     item_string = item_string.strip()
     if item_string != '':
      news_dates.append(item_date)
      news.append(item_string)
      news_count += 1
     current += 1
     time.sleep(delay)
   except:
    time.sleep(delay_except)
 return news_dates[::-1], news[::-1], news_count
```

B. Исходный код метода downloadStocks

```
def downloadStock(company, date_from, date_to):
 company = str(company)
 if company == 'sberbank':
  code = 'SBER'
  em = '3'
 elif company == 'gazprom':
  code = 'GAZP'
  em = '16842'
 dfs = date_from.split('/')
 df = dfs[0].lstrip('0')
 mf = str(int(dfs[1].lstrip('0')) - 1)
 yf = dfs[2]
 datef = dfs[0] + '.' + dfs[1] + '.' + dfs[2]
 dts = date_to.split('/')
 dt = dts[0].lstrip('0')
 mt = str(int(dts[1].lstrip('0')) - 1)
 yt = dts[2]
 datet = dts[0] + '.' + dts[1] + '.' + dts[2]
 cn = company
 url = 'http://export.finam.ru/stock.txt?market=1&em={}&code={}' +
     '&apply=0&df={}&mf={}&yf={}&from={}&dt={}&mt={}&yt={}&to={}' +
     '&p=8&f=stock_1&e=.txt&cn={}&dtf=4&tmf=3&MSOR=1&mstime=on' +
     '&mstimever=1&sep=1&sep2=1&datf=5&at=1'.format(em, code,
        df, mf, yf, datef, dt, mt, yt, datet, cn)
 stocks_dates = []
 stocks = []
 stocks_count = 0
 data = urlopen(url).read().decode("utf-8").split('\r\n')
 for i in range(1, len(data) - 1):
  item_split = data[i].split(',')
  stocks_dates.append(item_split[0])
  stocks.append(item_split[2])
  stocks_count += 1
 return stocks_dates, stocks, stocks_count
```

C. Исходный код метода stem

```
def stem(news_dates, news, news_count):
 stems_dates = []
 [stems_dates.append(date) for date in news_dates if date not in stems_dates]
 stems = []
 stems_count = len(stems_dates)
 i = 0
 j = 0
 while i < stems_count:
  stem = []
  while j < news_count and stems_dates[i] == news_dates[j]:
   words = text_to_word_sequence(news[j], filters = ''.join(punctuation) +
     '—01234567890abcdefghijklmnopqrstuvwxyz')
    for word in words:
     if word not in stemmer.stopwords and word != '␣':
      stem.append(stemmer.stem(word))
   j += 1
  i += 1
  stems.append('␣'.join(stem))
 return stems_dates, stems, stems_count
```

D. Исходный код метода connect

```
def connect(news_dates, news, news_count, stocks_dates, stocks, stocks_count):
 connections_dates = []
 for i in range(news_count):
  for j in range(stocks_count):
   if news_dates[i] == stocks_dates[j] and
     news_dates[i] not in connections_dates:
    connections_dates.append(news_dates[i])
 connections_news = []
 connections_stocks = []
 connections_count = len(connections_dates)
 i = 0
 j = 0
 k = 0
 while connections_dates[i] != news_dates[j]:
  j += 1
 while connections_dates[i] != stocks_dates[k]:
  k += 1
 while i < connections_count - 1:
  connection_news = []
  while j < news_count and connections_dates[i + 1] != news_dates[j]:
   connection_news.append(news[j])
   j += 1
  connections_news.append('␣'.join(connection_news))
  stocks_start = float(stocks[k])
  while k < stocks_count and connections_dates[i + 1] != stocks_dates[k]:
   k += 1
  stocks_end = float(stocks[k])
  connection_stocks = 1 if stocks_end > stocks_start else 0
  connections_stocks.append(connection_stocks)
  i += 1
 return connections_dates[:-1], connections_news, connections_stocks,
  connections_count - 1
```

Е. Исходный код метода fit

```
def fit(name):
 model = Sequential()
 model.add(Embedding(input_dim=num_words, output_dim=dimension))
 model.add(LSTM(units=dimension))
 model.add(Dropout(rate=dropout_rate))
 model.add(Dense(units=1, kernel_regularizer=l1_l2(l1=l1_rate, l2=l2_rate)))
 model.add(Activation(activation='sigmoid'))
 model.compile(optimizer=Adam(lr=l_rate), loss=binary_crossentropy,
   metrics=[binary_accuracy])
 hist = model.fit(training_X, training_y, batch_size=batch_size,
   epochs=epochs, validation_split=validation_split)
 model.save(path + 'models/{}_model-{}.h5'.format(company, name))
 with open(path + 'models/{}_history-{}.txt'.format(company, name),
   'w+', encoding='utf8') as temp:
  temp.write(str(hist.history))
 score = model.evaluate(testing_X, testing_y, batch_size=batch_size)
 with open(path + 'models/{}_score-{}.txt'.format(company, name),
   'w+', encoding='utf8') as temp:
  temp.write(str(score))
```

F. Результаты работы

Будут добавлены после финальных экспериментов