

Санкт-Петербургский Государственный Университет

Фундаментальная информатика и информационные технологии  
Информационные технологии

Зернов Алексей Викторович

Разработка системы автоматического  
анализа новостных публикаций на  
финансовом рынке

Бакалаврская работа

Научный руководитель:  
к. ф.-м. н., доцент кафедры информатики  
Санкт-Петербургского государственного  
университета Григорьев Д. А.

Рецензент:  
д. т. н., профессор, декан факультета  
информационных технологий и управления  
Санкт-Петербургского государственного  
технологического института Мусаев А. А.

Санкт-Петербург  
2017

SAINT-PETERSBURG STATE UNIVERSITY

Fundamental Informatics and Information Technology  
Information Technology

Alexey Zernov

# Development of automatic analysis system of financial market news publications

Bachelor's Thesis

Scientific supervisor:  
PhD., associate professor Dmitry Grigoryev

Reviewer:  
Sc. D., dean Alexander Musaev

Saint-Petersburg  
2017

# Оглавление

|   |           |
|---|-----------|
| <b>Введение</b> . . . . .                               | <b>5</b>  |
| <b>1 Финансовый рынок</b> . . . . .                     | <b>6</b>  |
| 1.1 Определение . . . . .                               | 6         |
| 1.2 Структура . . . . .                                 | 7         |
| 1.3 Участники . . . . .                                 | 8         |
| <b>2 Интеллектуальный анализ текста</b> . . . . .       | <b>9</b>  |
| 2.1 Процесс интеллектуального анализа текста . . . . .  | 10        |
| 2.1.1 Предварительная обработка текста . . . . .        | 11        |
| 2.1.2 Преобразование текста . . . . .                   | 12        |
| 2.1.3 Поиск признаков . . . . .                         | 12        |
| 2.1.4 Методы анализа текста . . . . .                   | 12        |
| 2.1.5 Интерпретация и оценка . . . . .                  | 12        |
| 2.2 Области применения интеллектуального анализа текста | 13        |
| 2.2.1 Извлечение информации . . . . .                   | 13        |
| 2.2.2 Информационный поиск . . . . .                    | 13        |
| 2.2.3 Обработка естественного языка . . . . .           | 13        |
| 2.2.4 Интеллектуальный анализ данных . . . . .          | 14        |
| <b>3 Обзор существующих инструментов</b> . . . . .      | <b>15</b> |
| 3.1 Natural Language Toolkit . . . . .                  | 15        |
| 3.2 Rymorphy2 . . . . .                                 | 15        |
| 3.3 Томита-парсер . . . . .                             | 16        |
| 3.4 Яндекс.Спеллер . . . . .                            | 16        |
| 3.5 OntosMiner . . . . .                                | 16        |
| <b>4 Программная часть</b> . . . . .                    | <b>17</b> |
| 4.1 Постановка задачи . . . . .                         | 17        |
| 4.2 Описание . . . . .                                  | 17        |
| 4.3 Используемые инструменты . . . . .                  | 18        |
| 4.4 Структура программы . . . . .                       | 19        |

|          |  |           |
|----------|--|-----------|
| 4.5      | Работа программы . . . . .   | 19        |
| 4.5.1    | Предварительная обработка . . . . .  | 20        |
| 4.5.2    | Построение модели . . . . .  | 23        |
| 4.5.3    | Обучение, тестирование, предсказание . . . . .                                 | 24        |
| 4.6      | Запуск программы . . . . .   | 25        |
| <b>5</b> | <b>Результаты . . . . .</b>  | <b>26</b> |
| 5.1      | Подбор параметров . . . . .  | 26        |
| 5.2      | Примеры оцененных новостей . . . . .   | 27        |
| 5.3      | Рекомендации по увеличению эффективности . . . . .                             | 28        |
|          | <b>Заключение . . . . .</b>  | <b>30</b> |
|          | <b>Список литературы . . . . .</b>   | <b>31</b> |
|          | <b>Приложение А Исходный код метода downloadNews . . .</b>                     | <b>32</b> |
|          | <b>Приложение В Исходный код метода downloadStocks . .</b>                     | <b>33</b> |
|          | <b>Приложение С Исходный код метода stem . . . . .</b>                         | <b>34</b> |
|          | <b>Приложение D Исходный код метода connect . . . . .</b>                      | <b>35</b> |
|          | <b>Приложение E Исходный код метода fit . . . . .</b>                          | <b>36</b> |
|          | <b>Приложение F Словарь «стоп-слов» . . . . .</b>                              | <b>37</b> |
|          | <b>Приложение G Пример обработки текста новости . . . . .</b>                  | <b>38</b> |
|          | <b>Приложение H Пример просто анализируемых текстов<br/>новостей . . . . .</b> | <b>39</b> |
|          | <b>Приложение I Пример трудно анализируемых текстов<br/>новостей . . . . .</b> | <b>40</b> |

# Введение

Не смотря на то, что с каждым годом происходит увеличение доли цифровой информации по отношению к бумажной, 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 повышение активности работы трейдеров за счет автоматического семантического анализа текста в рамках ограниченных временных ресурсов.

В работе будут рассмотрены основные определения, связанные с финансовым рынком (Раздел 1); базовая теория, касающаяся 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 инструментов и 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 характеристики, влияющие на их деятельность:

- Инвестиции в экономику страны
- Международные рынки, влияние тенденций глобализации
- Современные компьютерные технологии
- Уровень компьютерной 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. Примеры биржевых рынков: NYSE<sup>1</sup> или AMEX<sup>2</sup>. 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**.

---

<sup>1</sup>New York Stock Exchange — Нью-Йоркская фондовая биржа

<sup>2</sup>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. Информация 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 их 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 разметки (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

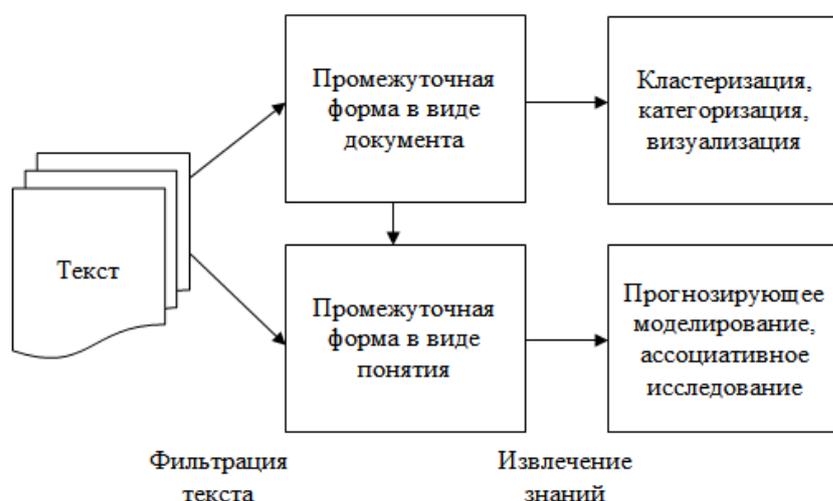


Рис. 1: Общий процесс интеллектуального анализа текста

**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 конкретной 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 predetermined 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[2] является пакетом библиотек и программ для разработки программ на Python, работающих с естественным языком. Сопровождается обширной документацией, а также книгой<sup>3</sup>, 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. Rymorphy2

Rymorphy2[3] написан на языке Python и имеет следующие возможности:

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

Распространяется rymorphy2 под лицензией MIT<sup>4</sup>, если используется в научной работе.

---

<sup>3</sup><http://www.nltk.org/book/>

<sup>4</sup><https://opensource.org/licenses/MIT>

### 3.3. Томита-парсер

Томита-парсер<sup>5</sup> 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<sup>6</sup> (написан на C и C++) находится в свободном доступе.

### 3.4. Яндекс.Спеллер

Яндекс.Спеллер<sup>7</sup> 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<sup>8</sup> является решением компании Eventos<sup>9</sup>, 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.

---

<sup>5</sup><https://tech.yandex.ru/tomita/>

<sup>6</sup><https://github.com/yandex/tomita-parser/>

<sup>7</sup><https://tech.yandex.ru/speller/>

<sup>8</sup><http://my-eventos.com/solution/ontosminer/>

<sup>9</sup><http://my-eventos.com/solution/ontosminer/>

## 4. Программная часть

### 4.1. Постановка задачи

Основной задачей работы было выявление взаимосвязей текста новости, связанной с компанией, с последующим изменением курса акций данной компании. Оценкой успешности работы программы можно считать процент верно предсказанных изменений без информации о том, что действительно произошло.

### 4.2. Описание

В результате работы была написана программа<sup>10</sup>, позволяющая автоматически анализировать новостные публикации сайта `mfd.ru`. Данная 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 рекуррентную 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

---

<sup>10</sup><https://github.com/Zernov/diploma/tree/master/src>

- `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)

### 4.3. 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 рекуррентной нейронной сети, для чего необходимы соответствующие инструменты

В связи с подзадачами был выбран язык программирования Python версии 3.6.0 и библиотеки `urllib`<sup>11</sup> (работа с web-запросами) версии 1.21.1, `bs4`<sup>12</sup> (обработка html-файлов) версии 4.6.0, `nltk`<sup>13</sup>[2] (преобразование текстовых документов) версии 3.2.2 и `keras`<sup>14</sup>[1] (работа с рекуррентными нейронными сетями) версии 2.0.3. Возможность написания

<sup>11</sup><https://docs.python.org/3/library/urllib.html>

<sup>12</sup><https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

<sup>13</sup><http://www.nltk.org/>

<sup>14</sup><https://keras.io>

всех программных модулей на одном языке упрощает разработку и поддержку, что было еще одним преимуществом.

## 4.4. 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
- `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

## 4.5. Работа программы

Работу программы (Рис. 3) 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» (Приложение F)). Во время построения модели выделяются и строятся требуемые слои рекуррентной нейронной сети.

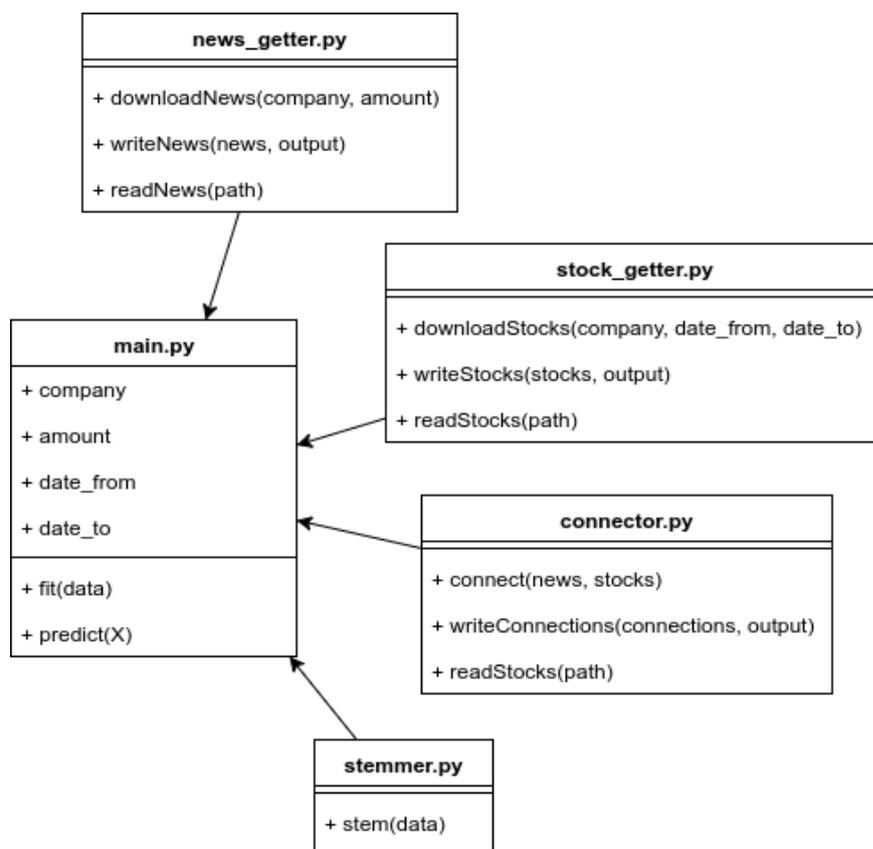


Рис. 2: Модули программы

#### 4.5.1. 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,

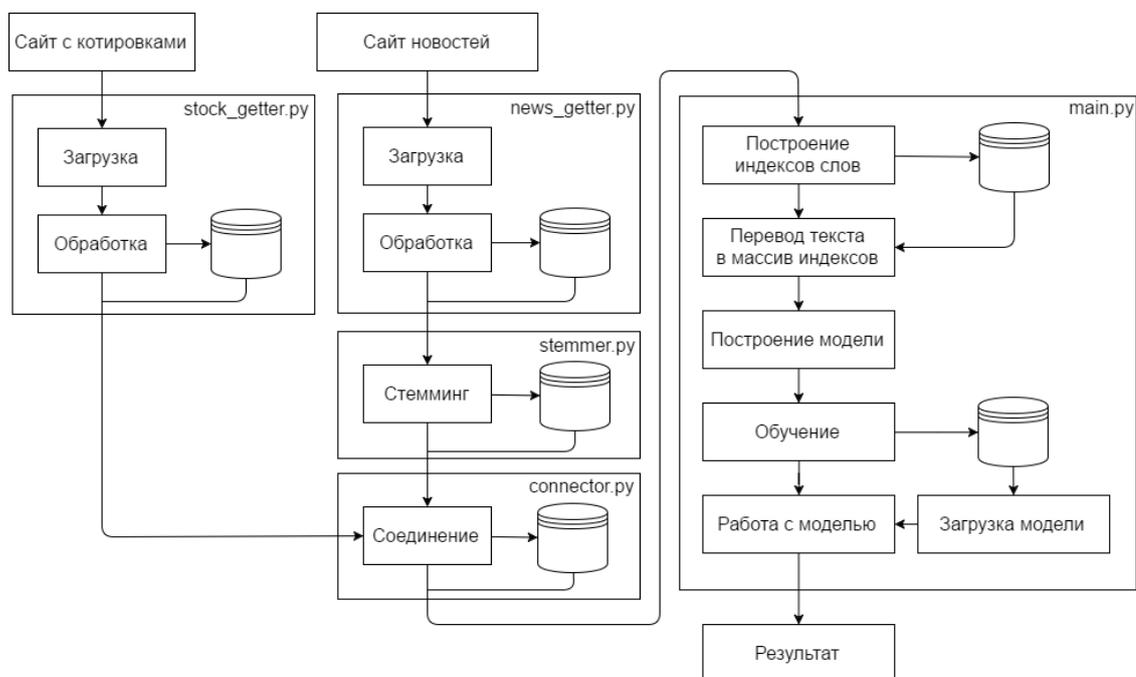


Рис. 3: Схема работы программы

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 В) 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 изме-

рения интервала между стоимостью 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 С) 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 основы 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 ин-

формации 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 (Приложение G), каждое из которых указывает на соответствующее слово в словаре.

#### 4.5.2. Построение модели

Как уже было сказано ранее, на основе полученных данных программа обучает рекуррентную нейронную сеть (RNN). Рекуррентная 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 [4], имеющий более объемную память и более высокую скорость обучения по сравнению с другими моделями рекуррентных нейронных сетей. Как видно из кода (Приложение E), в модели присутствуют слои: `Embedding`, `LSTM`, `Dropout`, `Dense` и `Activation` (Рис. 4). Рассмотрим подробнее некоторые из них.

**Embedding** 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 [4].

**Dropout** Схема работы Dropout подробно описана в работе [5]. 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.

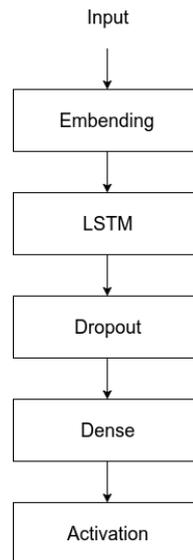


Рис. 4: Слои модели рекуррентной нейронной сети

### 4.5.3. Обучение, тестирование, предсказание

После построения имеющейся модели используется метод `fit` для непосредственного обучения модели. Данный метод имеет следующие параметры:

- `x` — набор входных данных в формате Numpy array
- `y` — значения в формате Numpy array
- `batch_size` — количество данных, которые берутся одновременно за одно обновление градиента
- `epochs` — количество поколений для обучения модели

- `validation_split` — число из интервала  $[0; 1)$ , часть набора входных данных, которая будет использоваться для валидации

Для тестирования используется метод `evaluate`, которому достаточно после окончания обучения передать всего два параметра: набор входных данных и значения. Метод `evaluate` возвращает вектор с потерями соответствующих данных.

Для предсказаний (в том случае, когда неизвестны реальные значения) используется метод `predict`, которому достаточно передать только набор входных данных. Результатом работы метода `predict` является набор предсказаний.

## 4.6. Запуск программы

Для запуска программы необходимо скачать файлы из репозитория<sup>15</sup>, находящиеся в каталоге `src` и установить используемые библиотеки. После чего есть два возможных способа запуска программы:

- `python main.py -f {company} {amount} {datef} {datet}`
- `python main.py {company} {predict_path}`

В первом случае происходит загрузка новостей заданной компании `company` в количестве, равном числу `amount`, загрузка котировок за период от `datef` до `datet`, построение и обучение модели. При запуске программы по второму шаблону происходит загрузка построенной модели для `company` с последующим предсказанием новостей, расположенных в файле по адресу `predict_path`.

Одним из недостатков данной программы является отсутствие возможности «дообучения» системы при подаче дополнительных данных к уже обученной модели. Необходимо по-новой ее заполнять с уже расширенным данным.

---

<sup>15</sup><https://github.com/Zernov/diploma>

## 5. Результаты

В качестве примера были взяты данные компании «Сбербанк» (10000 новостей). Построенная модель имела предсказывающую точность около 65%.

### 5.1. Подбор параметров

Во время подбора параметров были получены следующие зависимости:

- Оптимальное значение параметра  $l1$  (Рис. 5) находится около значения 0.1.
- Параметр  $l2$  (Рис. 6) позволяет получить наибольшую точность при значении 0.1.
- Оптимальное значение параметра  $lr$  (Рис. 7) является 0.01, последующее увеличение вызывает резкое падение точности.
- Параметр  $epoch$  (Рис. 8) при увеличении дает прирост точности, однако требуется значительно увеличивать количество эпох, чтобы достичь больших изменений.

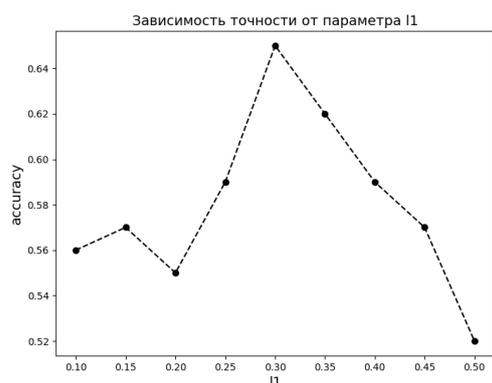


Рис. 5: Зависимость точности от параметра  $l1$

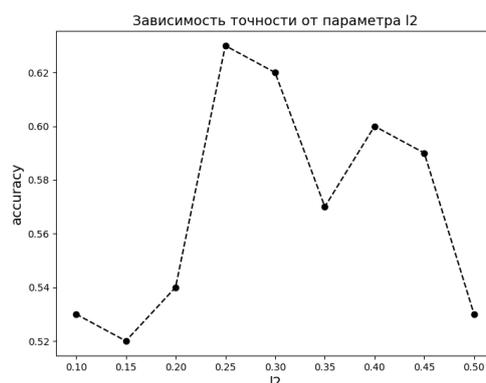


Рис. 6: Зависимость точности от параметра  $l2$

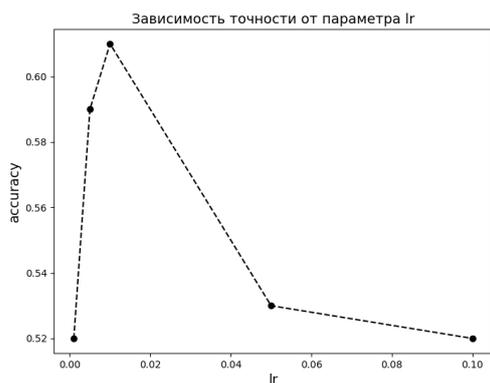


Рис. 7: Зависимость точности от параметра lr

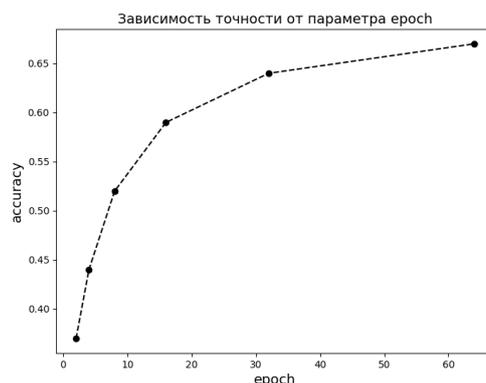


Рис. 8: Зависимость точности от параметра epoch

## 5.2. Примеры оцененных новостей

Если рассмотреть, на каких новостях программа выдает успешные результаты (Приложение Н), то можно сделать вывод, что в случае положительного изменения стоимости акций чаще присутствуют такие слова как «кредит», «владеют», «доля», а в случае отрицательного — «позволят», «выплатить», «ослабление», что вполне естественно. Кроме того, в текстах верно оцененных новостей чаще всего не присутствует неоднозначно интерпретируемых слов или цитат.

Если же подробнее посмотреть на неудачные результаты (Приложение I), чья вероятность успеха очень близка к 0.5, то одновременное присутствие положительной стороны в виде слова «кредит» и противоречащих ему негативно оцененных слов уменьшают общую вероятность успеха, создавая неопределенность. Ровно такая же ситуация и в случае, если присутствуют другие противоречащие слова: и имеющие положительную оценку, и отрицательную. В одной из ситуаций это может быть речевым оборотом, означающим ровно противоположное, а в другом — чьей-нибудь цитатой, не соответствующей действительности, однако которую программа восприняла серьезно.

В негативных случаях необходимо больше данных для обучения или специфические способы обработки отдельных часто встречающихся случаев. Однако не смотря на погрешности, программа выдает достаточно близкий к реальности результат (Рис. 9).

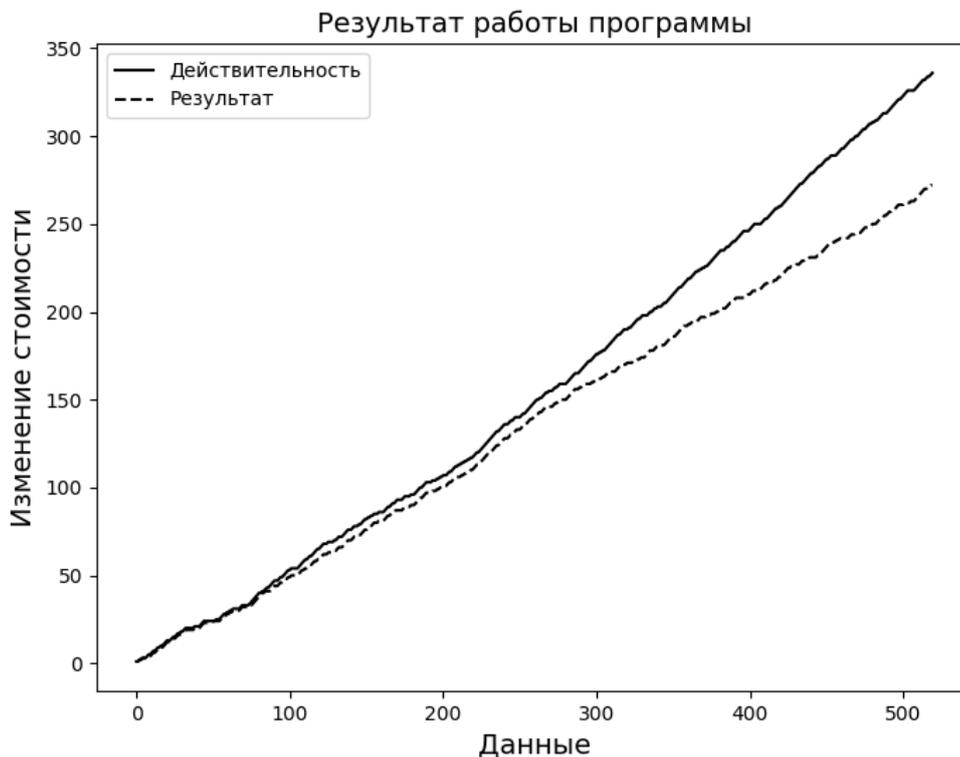


Рис. 9: Результат работы программы

### 5.3. Рекомендации по увеличению эффективности

В рамках имевшихся ресурсов (как вычислительных, так и временных) имело место ограничение на объем данных для обучения. Например, из 6300 изначально скаченных новостей получился набор данных размером около 300 элементов, так как минимальной временной единицей являлся один день. В таком случае имеет место одно (или несколько) из следующих решений:

**Отсутствие привязки новостей к определенным компаниям** В данном случае принадлежность новости к компании можно устанавливать какими-либо специальными метками, а само обучение проводить на данных, не зависящих от компании. В таком случае набор данных будет расширен в разы за счет получения информации о различных эмитентах одновременно. Но в данном случае возможно снижение эффективности за счет сложности разнообразных зависимостей акций

компании друг от друга. В связи с чем возникает идея брать «кластеры» компаний, имеющих более-менее похожий вектор изменения, отслеживая их группами. Но для реализации подобного необходим первоначальный анализ данных, который можно произвести с помощью программы, написанной в результате этой работы.

**Увеличение количества источников** В этом случае вместо единственного новостного сайта предлагается использовать несколько, в связи с чем возможна проблема дублирования новостей, но есть вероятность, что точность при этом возрастет.

**Загрузка более старых новостей** Последним из предлагаемых решения является увеличение временного промежутка с целью загрузки более ранних новостей. С одной стороны предполагается увеличение точность за счет расширения данных для обучения, но с другой стороны слишком старая информация может оказаться неактуальной в данный момент.

В каждом из трех предложенных решений подразумевается расширение объема данных для обучения, а следовательно требуется увеличение вычислительной мощности и дополнительные временные ресурсы. Однако результаты текущей работы могут стать основой для более серьезных разработок в данной области.

## Заключение

В данной работе представлена программа, позволяющая автоматически анализировать новостные публикации компаний в соответствии с ценами их акций в соответствующие временные промежутки. Кроме того, программа имеет хорошую точность в предсказании изменения стоимости акций после публикации определенной группы новостей. Полученный результат может быть расширен (за счет модульной архитектуры) на любое число компаний и новостных источников. Также результат данной работы может быть использован в качестве основы для разработки более крупных систем финансового анализа.

## Список литературы

- [1] Chollet François et al. Keras. — <https://github.com/fchollet/keras>. — 2015.
- [2] Loper E. Bird S. 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.
- [3] M. Korobov. Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts. — Springer International Publishing, 2015. — Vol. 542 of Communications in Computer and Information Science. — P. 320–332.
- [4] S. Hochreiter, J. Schmidhuber. Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.
- [5] Srivastava N. Hinton G. Krizhevsky A., I. Sutskever. Dropout: a simple way to prevent neural networks from overfitting. // Journal of Machine Learning Research. — 2014. — Vol. 15, no. 1. — P. 1929–1958.
- [6] Sumathy K. L. Chidambaram M. Text Mining: Concepts, Applications, Tools and Issues — An Overview // International Journal of Computer Applications. — 2013. — October. — Vol. 80, no. 4. — P. 29–32.
- [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() + '\n'
            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
```

## V. 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
```

## С. 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))
```

## Г. Словарь «стоп-слов»

|            |            |            |              |              |
|------------|------------|------------|--------------|--------------|
| 1: и       | 32: вот    | 63: ничего | 94: этого    | 125: нас     |
| 2: в       | 33: от     | 64: ей     | 95: какой    | 126: про     |
| 3: во      | 34: меня   | 65: может  | 96: совсем   | 127: всего   |
| 4: не      | 35: еще    | 66: они    | 97: ним      | 128: них     |
| 5: что     | 36: нет    | 67: тут    | 98: здесь    | 129: какая   |
| 6: он      | 37: о      | 68: где    | 99: этом     | 130: много   |
| 7: на      | 38: из     | 69: есть   | 100: один    | 131: разве   |
| 8: я       | 39: ему    | 70: надо   | 101: почти   | 132: три     |
| 9: с       | 40: теперь | 71: ней    | 102: мой     | 133: эту     |
| 10: со     | 41: когда  | 72: для    | 103: тем     | 134: моя     |
| 11: как    | 42: даже   | 73: мы     | 104: чтобы   | 135: впрочем |
| 12: а      | 43: ну     | 74: тебя   | 105: нее     | 136: хорошо  |
| 13: то     | 44: вдруг  | 75: их     | 106: сейчас  | 137: свою    |
| 14: все    | 45: ли     | 76: чем    | 107: были    | 138: этой    |
| 15: она    | 46: если   | 77: была   | 108: куда    | 139: перед   |
| 16: так    | 47: уже    | 78: сам    | 109: зачем   | 140: иногда  |
| 17: его    | 48: или    | 79: чтоб   | 110: всех    | 141: лучше   |
| 18: но     | 49: ни     | 80: без    | 111: никогда | 142: чуть    |
| 19: да     | 50: быть   | 81: будто  | 112: можно   | 143: том     |
| 20: ты     | 51: был    | 82: чего   | 113: при     | 144: нельзя  |
| 21: к      | 52: него   | 83: раз    | 114: наконец | 145: такой   |
| 22: у      | 53: до     | 84: тоже   | 115: два     | 146: им      |
| 23: же     | 54: вас    | 85: себе   | 116: об      | 147: более   |
| 24: вы     | 55:нибудь  | 86: под    | 117: другой  | 148: всегда  |
| 25: за     | 56: опять  | 87: будет  | 118: хоть    | 149: конечно |
| 26: бы     | 57: уж     | 88: ж      | 119: после   | 150: всю     |
| 27: по     | 58: вам    | 89: тогда  | 120: над     | 151: между   |
| 28: только | 59: ведь   | 90: кто    | 121: больше  |              |
| 29: ее     | 60: там    | 91: этот   | 122: тот     |              |
| 30: мне    | 61: потом  | 92: того   | 123: через   |              |
| 31: было   | 62: себя   | 93: потому | 124: эти     |              |

## Г. Пример обработки текста новости

**Исходная новость** ”Доходность по инвестициям пенсионных накоплений в 2014 году положительная. Порядка 2%, но мы еще не распределяли. Доходность положительная, несмотря на обвал ОФЗ, рынка облигаций и т.д.”, — сказала она. Структура портфеля пенсионных накоплений фонда на данный момент распределена следующим образом: 40% на банковских депозитах, 40% — в корпоративных облигациях, около 2–3% — в акциях. При этом около 20% накоплений инвестированы в краткосрочные финансовые инструменты, остальные — в долгосрочные.

**Новость после стемминга** доходн инвестиц пенсион накоплен год положительн порядк распределя доходн положительн несмотр обва офз рынк облигац т д сказа структур портфел пенсион накоплен фонд дан момент распредел след образ банковск депозит корпоративн облигац окол акц окол накоплен инвестирова краткосрочн финансов инструмент остальн долгосрочн

**Новость в виде вектора индексов слов** [346, 315, 467, 913, 1, 669, 314, 346, 669, 576, 914, 14, 175, 971, 11, 176, 91, 467, 913, 64, 19, 258, 111, 331, 48, 424, 168, 175, 49, 42, 49, 913, 823, 50, 533, 761, 549]

## Н. Пример просто анализируемых текстов новостей

- Число выданных жилищных кредитов увеличилось в прошлом месяце на 14% к февралю 2014 года — до 3,235 тысячи штук. Сбербанк России — крупнейший банк в России, на его долю приходится около трети активов всего российского банковского сектора. Учредителем и основным акционером Сбербанка является Центральный банк РФ, владеющий 50% уставного капитала плюс одна голосующая акция. Остальными акциями банка владеют российские и международные инвесторы.

*Вероятность роста: 0.83210963*

- В ноябре прошлого года финансовый директор Сбербанка Александр Морозов говорил, что ситуация в российской экономике и на Украине вряд ли позволят банку выплатить щедрые дивиденды по итогам 2014 года. Глава ЦБ Эльвира Набиуллина в феврале заявляла, что банкам с госучастием в 2015 году необходимо сделать послабления по дивидендам. Также на заседании будет рассмотрен ряд традиционных вопросов, среди которых отчет банка по МСФО, кандидаты в наблюдательный совет, созыв годового собрания акционеров.

*Вероятность роста: 0.18917511*

# I. Пример трудно анализируемых текстов новостей

- "Меня часто спрашивают, а что с кредитованием, что происходит с кредитованием в кризис? Я посмотрел за прошлую неделю, мы выдали кредитов на 7 миллиардов рублей за неделю. Сравнил с 2014 годом, это где-то средняя цифра по 2014 году", — рассказал Шаров в эфире "Коммерсант FM". При этом он отметил, что в кризис существенно изменилось направление кредитования. "В основном это оборотные средства. И для нас, и для правительства, я думаю, и для Центрального банка это серьезный вызов", — заявил представитель Сбербанка.

*Вероятность роста: 0.52203059*

- В расчет этого показателя Сбербанк включает чистые активы украинских подразделений группы, а также инвестиции в финансовые и долговые инструменты украинского правительства и корпоративных клиентов Украины. По состоянию на 31 декабря 2013 данный показатель составлял 0,8%. "Текущая ситуация в Украине и ее последующее негативное развитие может негативно воздействовать на финансовый результат и финансовое положение группы, и эффект данного воздействия на данный момент сложно определить", — отмечается в отчете. Руководство Сбербанка неоднократно заявляло, что крупнейший российский банк не планирует уходить с украинского рынка, несмотря на сложную политическую ситуацию.

*Вероятность роста: 0.51229823*