Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Московский государственный технический университет
имени Н. Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э.Баумана)

Факультет «Информатика и системы управления»
Кафедра «Программное обеспечение ЭВМ и информационные
технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
НА ТЕМУ:

МЕТОД ОБНАРУЖЕНИЯ ВЫБРОСОВ ВРЕМЕННЫХ РЯДОВ

Студент ИУ7-82 _____ А. И. Капустин

Руководитель ВКР _____ А. А. Оленев

Консультант _____ _____

Консультант _____ _____

Нормоконтролер _____ _____

Москва, 2020

ТЗ

ТЗ

план

Реферат

РПЗ 50 страниц, 10 рисунков, 9 таблиц, 27 источников.

Цель квалификационной работы бакалавра – разработка программного комплекса поиска аномалий. В квалификационной работе бакалавра проведён обзор алгоритмов поиска аномалий, а так же обзор алгоритмов для их улучшения, проанализирована предметная область. Предложен новый метод поиска аномалий. Создан программный продукт, реализующий предложенный метод,а так же вспомогательный программный продукт, которые осуществляет сбор данных для анализа.

Содержание

Введение 9
1 Аналитический раздел 10
1.1 Цель и задачи работы 10
1.2 Что такое спортивное прогнозирование 10
1.3 Сложность прогнозирования результатов 10
1.4 Основные правила игры в теннис 11
1.5 Основные подходы прогнозирования теннисных матчей . . 12
1.5.1 Метод подсчёта очков 12
1.5.1.1 Вероятность выиграть гейм 12
1.5.1.2 Вероятность выиграть сет 13
1.5.1.3 Вероятность выиграть матч 13
1.5.2 Методы попарного сравнения 14
1.5.3 Машинное обучение 15
1.5.3.1 Логистическая регрессия 16
1.5.3.2 Нейронные сети 17
1.5.3.3 Метод опорных векторов 19
1.5.3.4 Проблемы машинного обучения 19
1.5.3.5 Оптимизация гиперпараметров 20
1.5.3.6 Выбор оптимальных свойств 20
1.5.4 Предварительная обработка данных 22
1.5.4.1 One-Hot Encoding 22
1.5.4.2 Dense Embedding Vectors 24
1.5.4.3 Улучшения word2vec 26
1.5.4.4 Недостатки Dense Embedding Vectors 27
1.5.5 Нормализация данных 28
1.5.6 Выводы 28
2 Конструкторский раздел 30
2.1 Обработка текстовых данных 30
2.2 Определение тональности текста 31
2.3 Нормализация данных 32
2.4 Выбор оптимального набора параметров 32
2.5 Прогнозирование результатов 33
3 Технологический раздел 34
3.1 Выбор средств разработки 34
3.1.1 Выбор целевой платформы 34
3.1.2 Выбор языка программирования 34
3.1.3 Выбор среды разработки и отладки 34
3.2 Система контроля версий 35
3.3 Формат файлов 35

3.4 Используемые библиотеки 36
3.5 Установка программного обеспечения 36
3.6 Пример работы программы 37
4 Исследовательский раздел 38
4.1 Поиск оптимального набора статистических свойств 38
4.2 Влияние тональности на точность прогнозов 39
4.3 Сравнение алгоритмов прогнозирования теннисных матчей 39
Заключение 41
Список использованных источников 42
А Приложение А 46

Глоссарий

Априорная информация — Информация, которая была получена ранее рассматриваемого момента времени [1].

Теннис — Спортивные турниры, проводимые по правилам Международной Федерации Тенниса [2]

— Рейтинг ATP - официальная, еженедельно обновляемая система подсчета достижений спортсменов, используемая Ассоциацией теннисистов-профессионалов [3]

— Датасет - обработанный набор очищенных данных, пригодных для обработки алгоритмами машинного обучения [4]

— Embedding — это сопоставление произвольной сущности (например, узла в графе или кусочка картинки) некоторому вектору [5]

— Словарь текста - множество всех слов текста, где каждое слово, принадлежащая заданному тексту, упоминается ровно 1 раз.

Введение

Спортивные соревнования - это события на результат которых влияет огромное количество факторов. Результаты соревнований носят недетерминированный характер. Научиться прогнозировать исходы спортивных событий - первый шаг к прогнозированию исходов событий, влияющих напрямую на жизни людей (политика, сложные международные конфликты).

Результатами спортивных соревнований интересуются миллионы людей каждый день, от результатов команд зависит заработок спонсоров спортивных клубов, спортсменов, букмекеров и многих других. Теннис имеет простой набор правил, небольшое количество игроков, а также всего два возможных исхода, что позволяет разрабатывать алгоритмы прогнозирования без большого погружения в специфику спорта. Добиться стопроцентной точности прогнозов невозможно априори, но можно её повышать путем улучшения существующих алгоритмов, их совмещения (например, путем ансамблирования). В данной работе предлагается новый метод, позволяющий прогнозировать результаты теннисных матчей.

1 Аналитический раздел

1.1 Цель и задачи работы

Целью данной работы является создание метода прогнозирования результатов теннисных матчей на основе априорной информации. Для достижения данной цели необходимо решить следующие задачи:

— проанализировать предметную область и существующие методы прогнозирования результатов теннисных матчей

— разработать метод прогнозирования результатов теннисных матчей

— создать ПО, обрабатывающее данные для анализа

— создать ПО, реализующее разработанный метод прогнозирования результатов теннисных матчей

1.2 Что такое спортивное прогнозирование

««««< HEAD Спортивное прогнозирование предполагает предугадывание результатов предстоящих спортивных событий или контрольных результатов, которые спортсмен(или команда спортсменов) показывает на спортивных соревнованиях[6]. Прогнозы даются на конкретные события в конкретные моменты времени, на результат совокупности событий,ограниченных во времени(например, соревновательный сезон). Наиболее распространены прогнозы на результаты конкретного матча и сезона в целом. Прогнозы могу осуществляться на основе алгоритмов анализа информации, экспертной оценке, а так же комбинацией экспертной оценки. В данной работе будет рассмотрено прогнозирование на основе анализа априорной информации. Т.е. прогноз на какое-либо событие будет даваться до его начала и текущая информация о ходе события не будет учитываться.

1.3 Сложность прогнозирования результатов

Большинство исследователей используют различную статистическую информацию для составления прогнозов. Из массива накопленных данных они выбирают небольшое количество наиболее важных показателей(или же останавливаются только на одном из них) за ограниченный промежуток времени, которые подаются на вход алгоритму. Какие показатели считать значимыми определяет автор алгоритма на основе экспертной оценки или каких-то дополнительных алгоритмов "отбора наиболее важных показателей". Например, в случае американского футбола[7] это могут быть

а) Время владения мячом
б) Проводился ли матч дома или в гостях
в) Общее количество ярдов

г) Разница в атакующих ярдах

д) Количество потерь мяча

Или же в случае баскетбола, например, может браться следующий набор показателей:

а) Количество травмированных игроков

б) Количество выигранных матчей подряд перед данной игроков

в) Усталость команды. Показатель считается на основе расстояния, которое пришлось преодолеть команде, чтобы провести последние 7 игр

г) Средний домашний, гостей и общий процент побед

д) Рейтинг команды в "рейтинге нападения" , "рейтинге защиты" и "общем рейтинге" (подробнее методики подсчёта рейтингов описываются в [8])

Большая часть статей представляет команду как некое единое целое, упуская из виду каждого игрока команды. Учёт информации каждом игроке - нетривиальная задача. Поэтому для упрощения создания алгоритма прогнозирования с учётом каждого игрока команды был выбран вид спорта - одиночный теннис. В каждой команде по одному игроку, два возможных исхода матча - победа или поражения.

1.4 Основные правила игры в теннис

Ниже приведены основные правила, которые могут повлиять на составление алгоритмов прогнозирования. С полной версии правил можно ознакомится на сайте международной теннисной федерации[2].

а) Мяч должен приземлиться в пределах теннисного поля, чтобы продолжить игру. Если игрок мяч приземляется за пределами поля, то это приводит к потере им очков.

б) Игроки не могут дотронуться до сетки или перейти на сторону противника

в) Игроки не могут носить мяч или ловить его ракеткой

г) Игроки не могут дважды ударить по мячу

д) Игрок должен дождаться пока мяч пересечёт сетку, прежде чем ударить по нему

е) Игрок, который не возвращает мяч на половину поля противника, прежде чем он отскочит дважды, считается проигравшим

ж) Если мяч ударяет или касается игроков, то это карается штрафом

з) Перед тем как отбить подачу, мяч должен отскочить от поля.

и) Очки -наименьшая единица измерения. Приращение очков идёт в формате 0-15-30-40-гейм.

к) Гейм состоит из 4 очков и выигрывается, когда игрок набирает 4 очка с преимуществом не менее двух очков.

л) Сет состоит 6 геймов и выигрывается игроком, который набирает 6 геймов с минимальным отрывом в 2 очка.

м) Дополнительный сет - сет, который разыгрывается при достижении игроками счёта 6-6 по геймам.

н) Максимальное количество сетов в матче может достигать 3 или 5. По достижении 2 или соответственно 3 выигранных сетов матч заканчивается.

о) "Ровно" происходит когда достигнут счёт по очкам 40-40. Чтобы выиграть гейм, игрок должен выиграть два последовательных очках. Если игрок выигрывает одно очко, то счёт в гейме становится "больше" но если он теряет следующее очко, то счёт возвращается к "ровно" .

п) Когда в гейме достигнут счёт 6-6, то играется тай-брейк. В розыгрыше тай-брейке используются особые правила набора очка. Победителей считается игрок, набравший не менее 7 очков с преимуществом два очка над оппонентом.

1.5 Основные подходы прогнозирования теннисных матчей

Существует три основных подхода к прогнозированию теннисных матчей: методы попарного сравнения, методы вычисления очков и методы машинного обучения.

1.5.1 Метод подсчёта очков

Методы вычисления очков(также их называют иерархическими методами) фокусируются на оценке вероятности выигрыша каждого очка в матче. В этом подходе предполагается, что достаточно знать вероятность выигрыша игроком А p_a^r очка на своей подаче и вероятность выигрыша очка игроком В на своей подаче p_b^r. Аналогичные подходы применяются при подсчете вероятности выигрыша игрока в бадминтоне[9] и сквоше[10].Впервые такой метод был применен к теннису в работах Кси и Бюрича[11], Картера и Крю[12], Полларда[13]. Такие работы одинаково определяют вероятность выигрыша очка в матче как равномерную случайную величину, т.е. вероятности p_a^r и p_b^r принимаются за постоянные величины на протяжении всего матча. Анализ статистических показателей показывает, что такое предположение допустимо[14], т.к. отклонения от ожидаемой величины невелики.

1.5.1.1 Вероятность выиграть гейм

Игрок А может выиграть гейм у игрока В со счётом [4,0], [4,1],4,2] в случае если игрок А выиграл 4 очка за гейм, а игрок В выиграл не более двух.

Если же счёт в гейме стал [3,3], то такая ситуация называется "ровно" . В этом случае игрок А может выиграть гейм, закончив его с итоговым счетом [n + 5, n + 3], где $n \geqslant 0$. Чтобы посчитать вероятность p_G^R выигрыша игроком А гейма на своей подаче, введем следующие обозначения: p_A^R - вероятность выигрыша очка на своей подаче игроком А, $q_A^R = 1 - p_a^R$, $q_A^G = 1 - p_A^G$, $p_A^G(i,j)$ - вероятность того, что i очков в гейме наберет игрок А, а j очков в гейме наберет игрок В. В результате получим следующую формулу:

$$p_A^G = \sum_{j=0}^{2} p_A^G(4,j) + p_A^G(3,3) \sum_{n=0}^{\inf} p_a^{DG}(n+2,n) \tag{1.1}$$

Пусть $p_A^{DG}(n+2,n)$ - вероятность того, что игрок А выиграет с преимуществом в 2 мяча после того как был достигнут счет [3,3] на подаче игрока А.

$$p_A^{DG}(n+2,n) = \sum_{j=0}^{n} (p_A^R q_A^R)^j (q_A^R p_A^R)^{n-j} \frac{n!}{j!(n-j)!} (p_A^R)^2 = (p_A^R)^2 (p_A^R q_A^R)^n 2^n \tag{1.2}$$

В результате из формул 1 и 2 можно вывести следующую формулу - вероятность выигрыша игроком А гейма на своей подаче.

$$p_A^G = (p_A^R)^4 (1 + 4q_A^R + 10(q_A^R)^2) + 20(p_A^R q_A^R)^3 (p_A^R)^2 (1 - 2_A^R q_A^R)^{-1} \tag{1.3}$$

К

1.5.1.2 Вероятность выиграть сет

Пусть p_A^S - вероятность того, что игрок А выиграет сет у игрока В, если игрок А начинает подавать первым, $q_A^S = 1 - p_A^S$. Чтобы вывести p_A^G из p_A^G и p_B^G, определим $p_A^S(i,j)$ как вероятность того, что в розыгрыше сета игрок А выиграет i геймов, игрок В j геймов . Тогда

$$p_A^S = \sum_{j=0}^{4} p_A^S(6,j) + p_A^S(7,5) + p_A^S(6,6) p_A^T \tag{1.4}$$

Где p_A^T вероятность того, что игрок А выиграет 13-очковый тай-брейк , начинающийся с подачи игрока А.

1.5.1.3 Вероятность выиграть матч

Пусть p_A^M - вероятность того, что игрок А выиграет матч у игрока В. Определим $p_{AB}^M(i,j)$ как вероятность, что игрок а выиграет в матче после того как количество выигранных им сетов достигнет i, а количество выигранных сетов у игрока В будет j, где матч начинается с подачи игрока А, а заканчивается на подаче В. Аналогично - p_{AA}^M.

Аналогично зададим вероятности победы для сетов $p^S_{AB}, p^S AA, p^S_{BA}, p^S_{BB}$, где P^S_{XY} - вероятность того, что игрок сет закончится выигрышем игрока X, начинающийся с подачи X и заканчивающийся на подаче У. Очевидно, что количество разыгранных геймов для p^S_{XX} будет нечетным. Ограничим формулу 1.4

$$p^S_A = \sum_{j=1,3} p^S_A(6,j) + p^S_A(6,6)p^T_A \tag{1.5}$$

Если игрок X подает в первом сете матча, а Y в последнем, количества геймов в сете будет нечетным. Преобразуем 1.4.

$$p^S_A = \sum_{j=0,2,4} p^S_A(6,j) + p^S_A(7,5) \tag{1.6}$$

Итоговая формула для матчей где для достижения победы надо первым выиграть 3 сета(матчи мужского тенниса), будет выглядеть следующим образом(с более детальным выводом формул можно ознакомиться в [13], [14]).

$$p^M_A = \sum_{j=0,2,4}^{2} (p^M_{AA}(3,j) + p^M_{AB}(3,j)) \tag{1.7}$$

Для матчей до 2 выигранных сетов(женский теннис)

$$p^M_A = \sum_{j=0}^{1} (p^M_{AA}(2,j) + p^M_{AB}(2,j)) \tag{1.8}$$

1.5.2 Методы попарного сравнения

Основная идея методов попарного сравнения заключается в том, что каждого игроку присваивается некий положительный рейтинг, который характеризует уровень его навыков. Эти методы основаны на алгоритме, предложенном Бредли-Терри[15] и адаптированном для тенниса Ианом МакХолом[16]. Основная идея заключается в том, что вероятность победы игрока A над игроком B рассчитывается как

$$p_A = \frac{\alpha_A}{\alpha_A + \alpha_B} \tag{1.9}$$

Где α_A - рейтинг игрока A, а α_B - игрока B. Вариативность алгоритмов данного класса достигается за счёт различных методик подсчёта этого рейтинга. Например, при помощи ЭЛО-рейтинга[17]. Рассмотрим один из вариантов - подсчёт рейтинга, основанный на количестве выигранных геймов. Преимуществом этого метода является, тот факт, что берется в расчёт информация о ходе матча. Т.е. разница между матчами проигранным со счётом 6-0,6-0 и

со счётом 7-6, 7-6 существенна. Тогда вклад можно оценить по следующей формуле

$$p_A^M = L(\alpha_A,\alpha_B) \propto \frac{\alpha_A^{g_a}\alpha_b^{g_b}}{(\alpha_A+\alpha_B)^{g_A+g_B}} \tag{1.10}$$

где g_A-количество выигранных геймов игроком A, а g_B - игроком B(тай-брейк считается за обычный гейм). Приведем формулу для расчёта рейтинга игрока из одной из последних работ, посвященной данному классу методов

$$L(\alpha_a(t,); a = 1,...n) = \prod_{k\in Z_t}(\frac{\alpha_a(t,S)^{g_A}+\alpha_B(t,S)^{g_B}}{(\alpha_A(t,S)+\alpha_B(t,S))^{g_A+g_B}})^{exp(\epsilon(t-t_k))S_k} \tag{1.11}$$

Где t - время матча на поверхности теннисного корта S, k - индекс сыгранного матча, t_k - продолжительность матча k, $Z_t = k :< t_k < t$, n - количество игроков в модели, ϵ - параметр зависящий от S.

1.5.3 Машинное обучение

Машинное обучение - это область искусственного интеллекта (ИИ), которая изучает алгоритмы, которые обучаются на основе данных. Алгоритмы машинного обучения с учителем ставят свой задачей вывести получить функцию преобразования данных из помеченных данных, где помеченные данные представляют собой пары : вектор значений - результат. В теннисе обычно используют исторические данные для формирования данных для обучения. Например, вектор значений может содержать информацию о матче и игроках, а результатом соответственно будет служить исход матча. Выбор оптимальных параметров для формирования вектора напрямую влияет на точность получаемой функции. К проблеме прогнозирования теннисных матчей можно подойти двумя способами:

— Регрессионный подход, где результатом будет являться действительное число - вероятность выиграть матч. Вероятности выиграть матч неизвестны для исторических данных, что вынуждает помечать их в исторических данных метками 0 и 1.

— Подход бинарной классификации. Тогда результатом работы алгоритма будет предсказание выигрыша или проигрыша матча.

1.5.3.1 Логистическая регрессия

Несмотря на своё названия, логистическая регрессия является алгоритмом классификации. Свойства логической функции являются ключевыми для алгоритма. Логистическая функция определяется как:

$$\sigma = \frac{1}{1 + e^{-t}} \tag{1.12}$$

Логистическая функция отображает вещественные числа в диапазоне $(-\inf, +\inf)$ в диапазон $[0,1]$. Выходное значение логистические функции может интерпретироваться как вероятность.

Логистическая регрессия для прогнозирования матчей состоит из N свойств $x = (x_1, x_2, ..., x_n)$ и вектора n+1 вещественных параметров модели $\beta = (\beta_0, \beta_1, ..., \beta_n)$. Чтобы сделать прогноз при помощи модели, нужно преобразовать точке в n-мерном пространстве свойств в вещественной число

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ...\beta_n x_n \tag{1.13}$$

Подставляя z в формулу 1.14 получим

$$p = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.14}$$

Обучение модели состоит в оптимизации параметров β, таким образом, чтобы модель "как можно точнее" воспроизводила результаты матчей из данных для обучения. Это делается при помощи минимизации функции логистических потерь 1.15, которая показывает меру ошибку прогнозирования результатов матчей тренировочных данных.

$$L(p) = \frac{-1}{N} \sum_{i=1}^{N} p_i log(y_i) + (1 - p_i) + (1 - p_i) log(1 - y_i) \tag{1.15}$$

Где N - количество матчей в обучающих данных, p_i - предсказанная вероятность выиграть матч i, y_i - фактический исход матча(0 - поражение, 1 - победа).

В зависимости от размера датасета, выбирается одна из двух функций минимизации:

— Стохастический градиентный спуск - медленный итеративный метод, подходящий для больших датасетов

— Метод максимального правдоподобия - быстрый численный метод, не подходящий для работы с большими датасетами.

Большинство публикаций, в которых используется методы машинного обучения для прогнозирования теннисных матчей, используют логистическую

регрессию. Например, Кларк и Дит[18] построили логистическую регрессионную модель, основанную на разнице набранных очков в рейтинге ATP для прогнозирования результатов розыгрыша сета. Другими словами, они использовали одномерную регрессию x=(разница_в_рейтинге) и оптимизировали β таким образом, чтобы функция $\sigma(\beta_1 x)$ показала наилучший результат на обучающем датасете.Параметр β_0 был исключён их модели на основании того, что разница в рейтинге 0 может давать вероятность выигрыша матча 0.5. Вместо того, чтобы прогнозировать результаты матча, они предпочли предсказывать результаты каждого коконкретного сета, тем самым увеличивая набор данных для обучения.

Ма, Лу и Тан[19] использовали многомерную регрессию, использующую 16 свойств, поделённых на 3 категории: умения игрока, характеристики игрока и характеристики матча. Модель была обучена на матчах между 1991 и 2008 годом и была использована для улучшения тренировок игроков.

Логистическая регрессия привлекает исследователей тем, что она позволяет быстро обучать модели, почти не влечёт проблем переобучения моделей, а также позволяет получить легко интерпретируемый выходной параметр - вероятность выиграть матч. Тем не менее, при наличии сложных взаимосвязей между входными параметрами, использование логистической регрессии может стать нетривиальной задачей.

1.5.3.2 Нейронные сети

Искусственная нейронная сеть - это система взаимосвязанных "нейронов" вдохновлённая биологическими нейронами. Каждый нейрон преобразует входные данные в какое-то выходное значение, которое в дальнейшем могут быть использованы в качестве входных данных для других нейронов. Нейронная сеть обычно имеет несколько слоёв,с нейроном на каждом не входном слое, соединённом с нейронами в предыдущих слоях. Каждая связь в сети имеет свой вес. Нейрон использует входные значения и веса чтобы посчитать выходное значение. Типичным методом композиции является взвешенная сумма.

$$f(x) = K(\sum_i w_i x_i) \tag{1.16}$$

Нелинейная функции активации K позволяет нейронной сети решать нетривиальный задачи, используя небольшое количество нейронов. Логистическая функция 1.12 может использоваться в качестве функции активации.

Прогноз результата матча осуществляется на основе передачи в нейронную сеть неких свойств игрока и свойств матча. Если используется логичестическая функция активации, то выходное значение нейронной сети

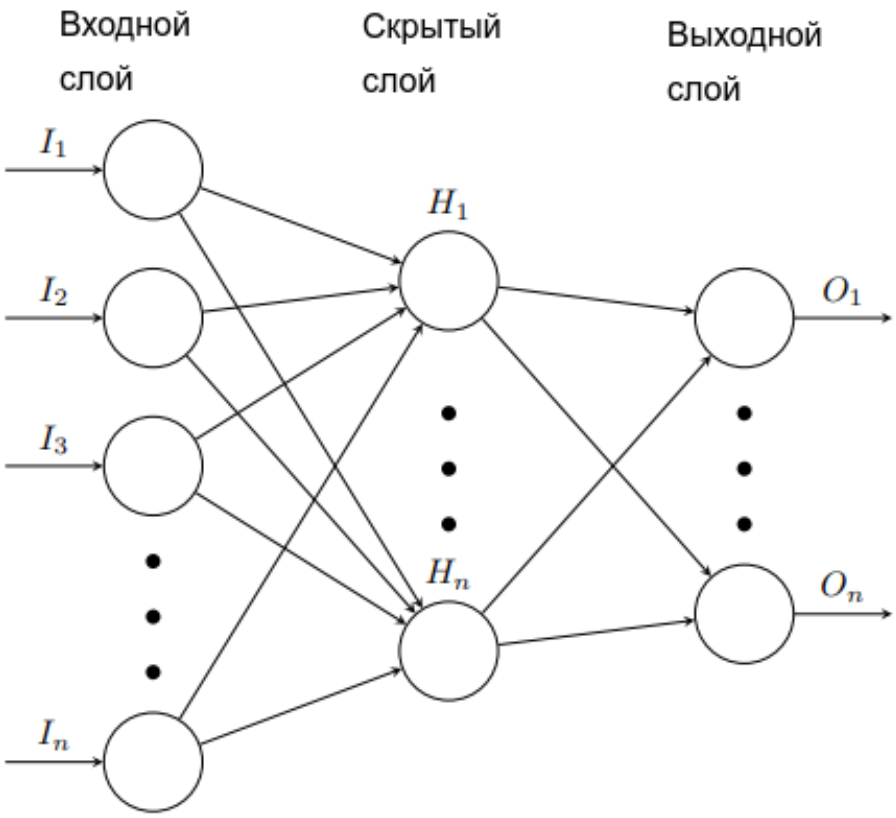

Рисунок 1.1 — Трехслойная нейронная сеть с прямыми связями

можно интерпретировать как вероятность выиграть матч. Есть большое количество различных алгоритмов обучения, которые оптимизируют набор весов нейронной сети в целях получении наилучшего выходного значения. Например, алгоритм обратного распространения использует градиентный спуск для уменьшения среднего квадрата ошибки между целевыми значениями и выходами сети.

Например, Сомбоонфокафан[20] сконструировал трёхслойную сеть прямой связи, использую алгоритм обратного распространения ошибки. Протестировал на различных архитектурах сети и различных наборах свойств. Нейронная сеть, показавшая наилучший результат, имеет 27 входных нейронов. На вход подавались свойства, описывающие обоих игроков, а так же свойства матча. В результате была получена точность предсказания 75% на исторических данных турниров "Большого шлема" в 2007-2008 годах.

Нейронная сеть можно установить наличие сложных взаимоотношений между различными свойствами, подаваемыми на вход. Но она работает по принципу "черного ящика" т.е. механизм определения таких взаимоотношений не будет известен. К недостаткам нейронных сетей так же можно отнести то, что они склонны к переобучению, зачастую требует большого количества данных для обучения. Кроме того, разработка структуры сети за-

частую является эмпирической и выбор гиперпараметров модели часто идёт методом проб и ошибок.

1.5.3.3 Метод опорных векторов

Основная идея метода опорных векторов заключается перевод исходных наборов свойств в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей. Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора. На данный момент в открытом доступе нет работ по прогнозированию теннисных матчей при помощи метода опорных векторов[21]. Метод опорных векторов имеет несколько преимуществ по сравнению с нейронными сетями. Во-первых, обучение не заканчивается на локальном минимуме, что зачастую бывает с нейронными сетями. Во-вторых, при наличии большого количество данных для обучения, метод опорных векторов обычно превосходит нейронные сети в прогнозировании [21]. Тем не менее, время обучения модели метода опорных векторов намного выше, а создание и задание необходимых параметров модели является нетривиальной задачей.

1.5.3.4 Проблемы машинного обучения

Наличие большого количества данных для обучения не означает получение более точных результатов прогнозирования результатов теннисных матчей, т.к. большое количество данных означает, что теннисист играл много лет матчи. Со временем его навыки меняются и результаты последних матчей представляют набольший интерес. Так же в теннисе важную роль играет покрытие на которой играется матч, т.е. для модели необходимы данные о последних матча игрока именно на этом покрытии. Поэтому машинное обучение может страдать от недостатка данных, что может приводить к переобучению модели, с которым может быть сложно бороться. В частности нейронные сети сильно подвержены переобучению если количество слоёв/нейронов сравнительно велико относительно размера данных для обучения.

Чтобы бороться с проблемой переобучения, нужно использовать только важные свойства для обучения. Процесс отбора важный свойств набора данных называется "отбор признаков" для этого существует большое количество различных алгоритмов. Удаление незначащих свойств значительно уменьшает время обучения

1.5.3.5 Оптимизация гиперпараметров

Модель имеет параметры как получаемые в процессе обучения путём их оптимизации(например, веса в нейронных сетях), так и статически задаваемые параметры такие как количество скрытых слоёв, количество нейронов в каждом слое. Статически настраиваемые параметры называют гиперпараметрами. Процесс получения оптимальных гиперпараметров может быть как эмпирическим, так и алгоритмическим. Пример алгоритмического поиска гиперпараметров - поиск по сетке. Он происходит на заранее определённом пространстве гиперпараметров. Для построения модели прогнозирования с высокой точностью, необходимо тщательно подобрать необходимые гиперпараметры.

1.5.3.6 Выбор оптимальных свойств

Каждый набор данных может содержать большое количество различных свойств. Каждое свойство вносит определенный вклад в итоговый результат. Выбор оптимального количества наиболее значимых свойств - приём очень часто применяемый в машинном обучение. От количества свойств зависит производительность получаемой модели. Свойства, не влияющие на итоговый результат, или влияющие незначительно, негативно влияют на производительность модели, при включении их в оную. Так же нерелевантные свойства могут ухушить точность модели, заставить её обучаться на нерелевантных данных[22].

Преимущества выбора оптимального набора свойства

— Уменьшает вероятность переобучения. Использование меньшего количества избыточных данных уменьшает вероятность принятия решения на основе "шума" .

— Повышение точности.

— Сокращает время обучения- меньшее количество данных снижают сложность модели.

Выбор свойств может производится как в ручном, так и автоматическом режиме.

Существуют 3 основных класса алгоритмов выбора свойств.

— Фильтр. Рассчитывается определенная метрика и свойства "фильтруются" на основании этой метрики

— Обертка. "Оберточные" методы рассматривают выбор оптимального набора свойств

Корреляция Пирсона - фильтр-метод, который использует в качестве метрики следующую формулу:

$$r = \frac{\sum(x - \overline{x})(y - \overline{y})}{\sqrt{\sum(x - \overline{x})^2 \sum(y - \overline{y})^2}} \tag{1.17}$$

Хи-квадрат метод - фильтр-метод, который использует в качестве метрики следующую формулу:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{1.18}$$

Где O_i - фактическое количество наблюдений класса i, а E_i - ожидаемое количество наблюдений класса i, при условии отсутствия зависимости между свойством и ожидаемым значением. Рассмотрим пример подсчёта метрики Хи-квадрат. Предположим, у нас есть следующие данные

Таблица 1.1 — Фактическое количество наблюдений результатов игроков

	Кол-во побед	Кол-во поражений	Всего
Игрок1	20	5	25
Игрок2	40	35	75
Всего	60	40	100

Для того чтобы посчитать значения χ^2, посчитаем "ожидаемое" значение в каждой ячейки, если бы свойства были действительно независимы. Для этого нужно просуммировать значения каждой строки и поделить на общее количество записей. Получим следующую таблицу;

Таблица 1.2 — Ожидаемое количество наблюдений результатов игроков

	Кол-во побед	Кол-во поражений	Всего
Игрок1	15	10	25
Игрок2	45	25	75
Всего	60	40	100

Лассо-метод(сокращение от англ. least absolute shrinkage and selection operator) - встроенный метод. На основе линейной регрессии происходит пошаговый выбор оптимальных параметров модели. Идея метода заключается в том, что накладывается ограничение на сумму абсолютных значений свойств модели, сумма должна быть меньше некоторого фиксированного значения. Для этого того, чтобы этого добиться, применяется методы

сжатия(регуляризации), в процессе которого коэффициенты регрессионных переменных уменьшаются. Свойства, получившие в результате регуляризации нулевой коэффициент, отбрасываются.

Методы на основе деревьев относятся к классу встроенных методов. Например, для определения оптимального наборы свойств может использоваться алгоритму случайного леса. Принцип работы встроенных методов нетривиален и выходит за рамки данной работы, подробнее с их принципами работы можно ознакомится в [23]

1.5.4 Предварительная обработка данных

Перед тем как начать с работать с данными, их нужно предварительно обработать.Данные могут хранится в числовом, текстовом виде, в формате изображений, аудио/видео файлов и других многочисленных форматах. Алгоритмы прогнозирования работают исключительно с числовыми данными. Поэтому данные в других форматах нужно перевести в числовой формат. В данной работе будет рассмотрено как перевести текст В данной работе мы будем рассматривать только данные, которые хранятся в текстовом виде. Перед тем как данные переводятся в числовой вид, они предварительно очищаются.

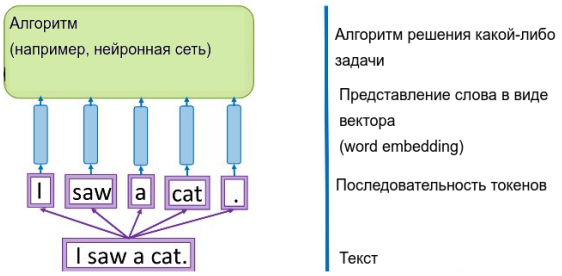

Рисунок 1.2 — Схема перевода текста в слова

1.5.4.1 One-Hot Encoding

One hot encoding - это процесс в помощью которого некоторые категориальные переменные(в нашем случае - слова), преобразуются в форму, понятную алгоритмам машинного обучения. Берется достаточно большой набор заранее известных слов(обычно речь идёт о десятках тысячах слов). На основании этого набора создаётся вектор чисел, где каждому слову соответствует определенная позиция в этом векторе с изначальным значением 0. Текст проверяется на наличие таких слов. Если слово присутствует в тексте, то соответствующая позиция устанавливает в 1, иначе в 0. На выходе получается разряженная матрица в виде вектора. Так же некоторые модификации

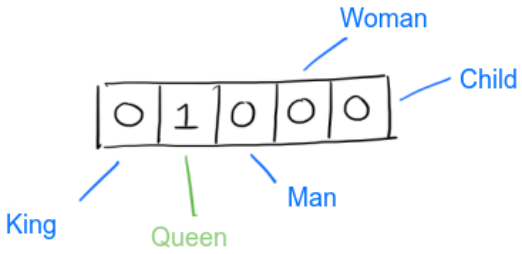

Рисунок 1.3 — Представление предложение "The queen entered the room" в виде вектора в формате one-hot encoding

этого метода используют размерности матрицы отличные от 1 по высоте. К недостаткам данного метода можно отнести:

— Большое количество затрачиваемой памяти. Текст из 40 слов и 4000 будут заниматься одинаковое количество место. Решается оптимизацией хранения разреженных матриц

— Потерю смысла слов. Одно и тоже слово может иметь несколько значений в зависимости от контекста

— Потерю порядка слов.

— Потеря количества определенных слов в тексте. Слово, употребленное в тексте один раз, имеет такой же вес, как и многократно употреблённое слово.

Так же к недостаткам стоит отнести трудоёмкость предварительной подготовки текста. Для этого метода необходимо привести слова в один падеж, одно число.Например, "кота" - > "кот" "коты" -> "кошка" . Так же слова-синонимы заменяются на один конкретный вариант. Например, "автомобиль->"машина". Это может быть нетривиально, особенно когда синонимы состоят из нескольких слов.

Основным преимуществом данного метода считается, что полученных данных достаточно для решения многих задачл ественной обработки языка[24], в частности классификации. Т.е. для многих задач анализа естественного языка вышеописанные недостатки несущественны. Так же такой формат данных удобен для работы с нейронными сетями и другими алгоритмами машинного обучения.

Развитием алгоритма one-hot encoding является так называемый "мешок слов" . Для его получения OHE-вектора каждого слова в тексте складываются. .е. на выходе получим просто подсчет количества различных слов в тексте в одном векторе. Такой подход называется "мешок слов" (bag of words, BoW), потому что мы теряем всю информацию о взаимном расположении слов внутри текста.

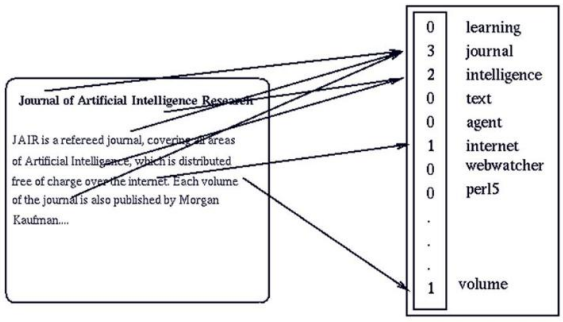

Рисунок 1.4 — Мешок слов

1.5.4.2 Dense Embedding Vectors

Вышеописанные подходы были (и остаются) хороши для областей, где количество текстов мало и словарь ограничен, хотя, конечно, они имеют свои недостатки Но в современном мире количество слов огромно, слова постоянно появляются и исчезают С этим надо было что-то делать, а ранее известные модели не могли справиться с таким объемом текстов. Предположим, что в английском языке сейчас примерно около миллиона слов. Матрица совместных встречаемостей(такие матрицы используются для сравнения текстов) только пар слов будет иметь размер $10^6 x 10^6$. Такие матрицы занимают очень много памяти, с ними неудобно работать. Конечно, придумано множество способов, упрощающих или распараллеливающих обработку таких матриц, всё-таки до сих пор с ними крайне тяжело работать[5].

Подход, решаюший данную проблему, называется Dense Embedding Vectors Этот подход основан на гипотезе, которую в науке принято называть гипотезой локальности — Эслова, которые встречаются в одинаковых окружениях, имеют близкие значенияЭ. Близость в данном случае понимается очень широко, как то, что рядом могут стоять только сочетающиеся слова. Например, для нас привычно словосочетание "заводной будильник". А сказать “заводной арбуз” нельзя — эти слова не сочетаются. Представим, что мы сожмём количество измерений вектора, используемого для представления миллиона уникальных слов с миллиона до 50-100. При таком подходе каждое слово не будет иметь своего измерения, а будет отображаться на некий вектор. Нейронная сеть вычисляет вероятность того, что определённое слово может встретиться в данном контексте.

	King	Queen	Woman	Princess
Royalty	0.99	0.99	0.02	0.98
Masculinity	0.99	0.05	0.01	0.02
Femininity	0.05	0.93	0.999	0.94
Age	0.7	0.6	0.5	0.1

Рисунок 1.5 — Контекстные вероятности

В 2013 году был предложен новый подход в Dense Embedding Vectors, названный word2vec. Модель, предложенная Миколовым[25], проста. Она основывается на предсказании вероятности слова по его окружению (контексту). То есть ожидается получение таких векторов слов, чтобы вероятность, присваиваемая моделью слову была близка к вероятности встретить это слово в этом окружении в реальном тексте. Приведем формулу "мягкого максимума" (дифференцируемого). Максимум должен быть дифференцируем, чтобы модель могла обучаться при помощи алгоритма обратного распространения ошибки.

$$P(w_o|w_c) = \frac{e^{s(w_0,w_c)}}{\sum_{w_i \in V} e^{s(w_0,w_c)}} \tag{1.19}$$

Где w_0 - вектор целевого слова, w_c - некоторый вектор контекста вычисленный(например, путем усреднения) из векторов окружающих данное слово слов, $s(w_1, w_2)$ - некая функция, получающая на вход 2 вектора и отображающая их в одно число. Например, это может быть функция косинусного расстояния.

Модель обучается следующим образом: берется последовательность 2k + 1 слов, где слово в центря является словом, которое должно быть предсказано. Окружащюие слова являются контекстом длины K с каждой стороны от искомого слова. В процессе обучения каждому слову сопоставляется уникальный вектор, который меняется в процессе обучения.

Такой подход называется CBOW(англ. continuous bag of words). Является подвидом метода "мешка слов" т.к. порядок слов в контексте не учитывается.

Так же существует развитие данного метода - skip-gram[25]("словосочетание с пропуском"). Оно основывается на том, что из данного нам слова мы пытаемся "угадать контекст" . В остальном алгоритм не претерпевает измненений.

Натренированная модель word2vec может улавливать некоторые семантические и синтаксические свойства слов, несмотря на то что в модель явно не заложено никакой семантики. Слово "мужчина"(англ. man) относит-

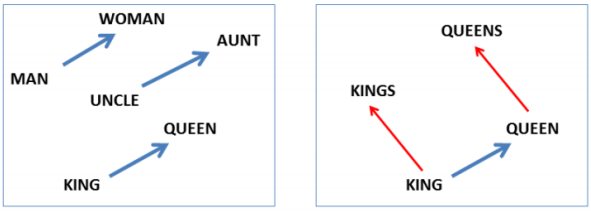

Рисунок 1.6 — word2vec пример семантической корреляции[25]

ся к слову "женщина"(англ. woman) так же, как слово "дядя"(англ. uncle) к

слову "тётя"(англ. aunt). Для человека это естественно и понятно, но в других моделям добиться такого же соотношения векторов можно только с помощью специальных ухищрений.

1.5.4.3 Улучшения word2vec

Модель CBOW с функцией оптимизации(минимизацией) в виде дивергенции Кульбака-Лейблера называют моделью CBOW с negative sampling[26].

$$KL(p||q) = \int p(x) log \frac{p(x)}{q(x)} dx \tag{1.20}$$

Где $p(x)$ — распределение вероятностей слов, а $q(x)$ - распределение словвзятое из модели(которая была получена на основе текста). Эта формула имеет описывает насколько одно распределение не похоже на другое(англ. divergence - расхождение). Т.к. распределение получается на основе подсчёта слов, оно является дискретным. Поэтому можно заменить интеграл на сумму:

$$KL(p||q) = \sum_{x \in V} p(x) log \frac{p(x)}{q(x)} dx \tag{1.21}$$

К сожалению, эта формула имеет недостаток - большую вычислительную сложность. Прежде всего, из-за того, что $q(x)$ рассчитывается как softmax на основании всего словаря текста. Именно поэтому Томасом Милковым[26] была предложена оптимизация вышеприведенной формула, названная Negative Sampling. Как уже отмечалось ранее, очевидно, что многие слова вместе не встречаются. Поэтому большая часть вычислений в softmax являются избыточными. Это позволяет ввести следующую оптимизацию: максимизируется вероятность встречи слова для нужного слова в типичном для него контексте(в том котором он чаще всего встречается в заданном тексте). А вероятность встречи слова в нетипичном для него контексте, соотвественно, минимизируется.

$$NegS(w_0) = \sum\nolimits_{i=1, x_i \sim D}^{i=k} - log(1 + e^{s(x_i, w_0)}) + \sum\nolimits_{j=1, x_j \sim D}^{i=k} - log(1 + e^{s(x_j, w_0)}) \tag{1.22}$$

Где $s(x,w)$ - аналогичен оригинальной формуле, D - haспределение совместной встречаемости слова w и остальных слов текста. Формлуа состоит из двух частей - $+s(x,w)$ и негативной $-s(x,w)$. Позитивная часть отвечает за типичные контексты, а негативная за атипичные контексты(т.е. редкие словосочентания). Они порождатся из распределения , D‘ которое берется как равномерное по всем словам словаря корпуса. Было показано, что такая функция приводит при своей оптимизации к результату, аналогичному стандартному softmax[26].

Так же возможен вариант оптимизации подсчёта модели word2vec через более эффективный подсчёт softmax. Например, используя дерево Хаффмана[27]

Идея оптимизации заключается в том, что на основе словаря текста строится дерево Хаффмана, в каждом листе дерева слово, всего V слов. Пусть путь до слова w2 - L(w), а j-тая вершина по пути к w2 - n(w,j). С

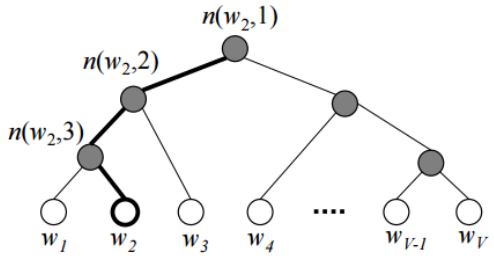

Рисунок 1.7 — Дерево Хаффмана

помощью иерархического softmax вектора $u_{n(w,j)}$ можно рассчитать для V - 1 внутренних вершин. Вероятность того, что w окажется выходным словом можно рассчитать по формуле:

$$p(w = w0) = \prod_{j=1}^{L(w)-1} = \sigma([n(w,j+1) = lch(n(w,j))]v_{n(w,j)}^{T}u) \qquad (1.23)$$

Где $\sigma(x)$ - функция softmax, lch(n) - левый сын вершины n, $u = v_{wI} = \frac{1}{h}\sum_{k=1}^{h} v_{wI,k}$ т.е. усредненный вектор контекста при использовании CBOW.

Стандартный алгоритм softmax имеет ассиптотическую сложноность O(V). Улучшенный алгоритм softmax позволяет вычислисть вероятность слова при помощи последовательных вычислений за O(log(V)).

1.5.4.4 Недостатки Dense Embedding Vectors

Методы Dense Embedding Vectors изначально были придуманы для английского языков, то там не так остро стоит проблема словоизменения для языков вроде русского. В вышеописанных методах предполагается, что разные слова одного слова считаются одним словом. Только тогда нейронная сеть сможет установить корректно семантические и синтаксические связи. Чтобы избежать проблем при формирования связей, используют стеммирование(обрезание окончания слова, оставление только основы) и/или лемматизация(замену слова его начальной формой) на этапе предварительной подготовки текста.

1.5.5 Нормализация данных

После получения предварительно обработанного датасета для обучения модели, данные необходимо нормализовать. Нормализация данных предназначена для устранения зависимости от выбора единицы измерения и заключается в преобразовании диапазонов значений всех атрибутов к стандартным интервалам([0,1] или [-1,1])[28]. Нормализация данных направлена на придание всем атрибутам одинакового "веса".

Рассмотрим основные методы нормализации данных.

а) Min-max нормализация заключается в применении к диапазону значений атрибута x линейного преобразования, которое отображает [min(x),max(x)] в [A,B].

$$x'_i = \tau(x_i) = \frac{x_i - min(x)}{max(x) - min(x)} * (B - A) + A \tag{1.24}$$

$$x \in [min(x), max(x)] \Rightarrow \tau() \Rightarrow [A,B] \tag{1.25}$$

Min-max нормализация сохраняет все зависимости и порядок оригинальных значений атрибута. Недостатком этгого метода является то, что выбросы могут сжать основную массу значений к очень маленькому интервалу

б) Z-нормализация основывается на приведении распределения исходного атрибута x к центрированному распределению со стандартным отклонением, равным 1 [28] .

$$x'_i = \tau(x_i) = \frac{x_i - \overline{x}}{\sigma_x} \tag{1.26}$$

$$M[x'] = 1 \tag{1.27}$$

$$D[\overline{x}'] = 0 \tag{1.28}$$

Метод полезен когда в данных содержатся выбросы.

в) Масштабирование заключается в изменении длины вектора значений атрибута путем умножения на константу [**?**] .

$$x'_i = \tau(x_i) = \lambda * x_i \tag{1.29}$$

Длина вектора x уменьшается при $|\lambda| < 1$ и увеличивается при $|\lambda| > 1$

1.5.6 Выводы

Существует большое количество алгоритмом прогнозирования результатов спортивных соревнований и теннисных матчей в частности. Однако, современные разработки в основном ведутся в области машинного обучения, в частности крайне популярны нейронные сети. Методы на основе нейронных сетей крайне гибки и позвляют легко разрабатывать собственные модификации уже известных методов.

Таблица 1.3 — Точность прогнозов различных методов

	Точность прогноза	Возможность модификации
Метод подсчёта очков	60-65%	Низкая
Методы попарного сравнения	63-67%	Средняя
Методы машинного обучения	69%+	Высокая

Алгоритмы могут использовать в качестве входных данных как числовые данные, так и текстовые данные, переведенных в вид чисел. При этом для перевода текстовых данных в численный вид оптимально использовать современные методы word embedding такие как word2vec. Область знаний прогнозирования спортивных соревнований и в частности прогнозирования результатов теннисных матчей активно развивается как часть современной науки, остается ещё много простора для исследования алгоритмов, модификации и создания новых.

Анализ текста будет проводится на основе англоязычных текстов в связи с тем, что современные алгоритмы заточены на работу с английским языком, а также большинство данных о теннисе представлены на английском языке.

2 Конструкторский раздел

В этом разделе приводится подробное описание разрабатываемого метода, выделяются основные его компоненты, описываются метрики, оценивающие метод. В выводе аналитической части предлагается разработать новый метод прогнозирования результатов теннисных матчей. Новый метод будет включать в себя построение модели на основе численных статистических данных, а так же текстовых данных.

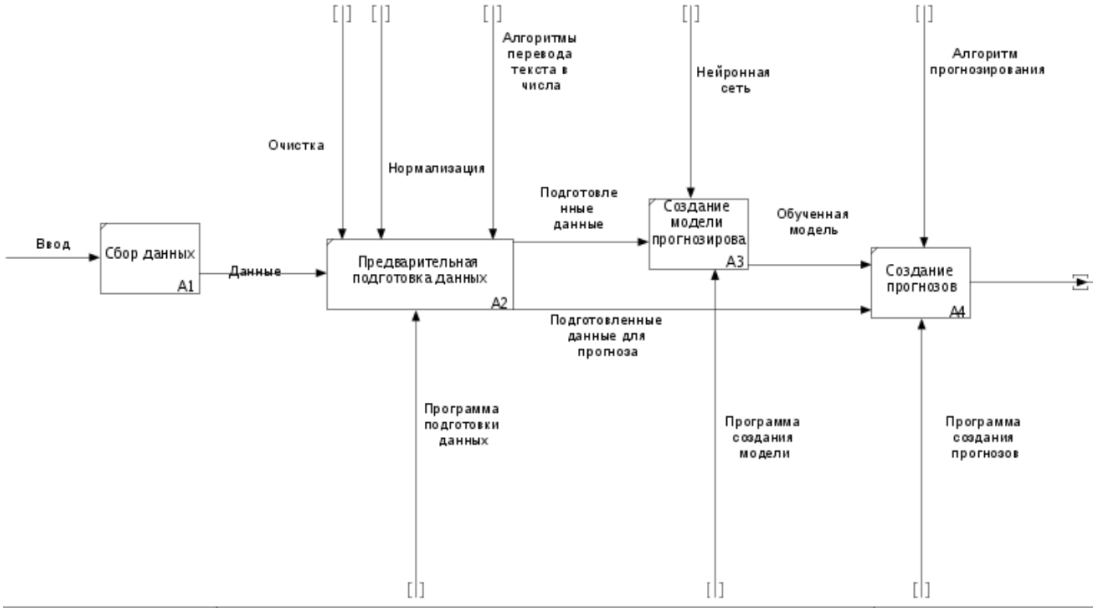

Рисунок 2.1 — Общая схема программного комплекса

2.1 Обработка текстовых данных

Работа осуществляется с текстами только на английском языке. Для того чтобы использовать преобразовать текстовые данные в числовое представление, нужно выполнить следующие действия:

— Удалить все нерелевантные символы (например, любые символы, не относящиеся к цифро-буквенным).

— Токенизировать текст, разделив его на индивидуальные слова.

— Удалить нерелевантные слова — например, гиперссылки.

— Перевести все символы в нижний регистр для того, чтобы слова «привет», «Привет» и «ПРИВЕТ» считались одним и тем же словом.

— Произвести лемматизацию, т. е. сведения различных форм одного слова к словарной форме (например, «машина» вместо «машиной», «на машине», «машинах» и пр.)

— Удалить слова, не несущих самостоятельной смысловой нагрузки. Например, артиклей в английском языке.

— Произвести стемминг текста, т.е. заменить исходные слова их основами.

2.2 Определение тональности текста

Используем численную метрику тональности текста для её добавления в численный набор данных. Для этот используем вышеописанный метод word2vec с алгоритмом CBOW с размером контекста 1. Первоначально каж-

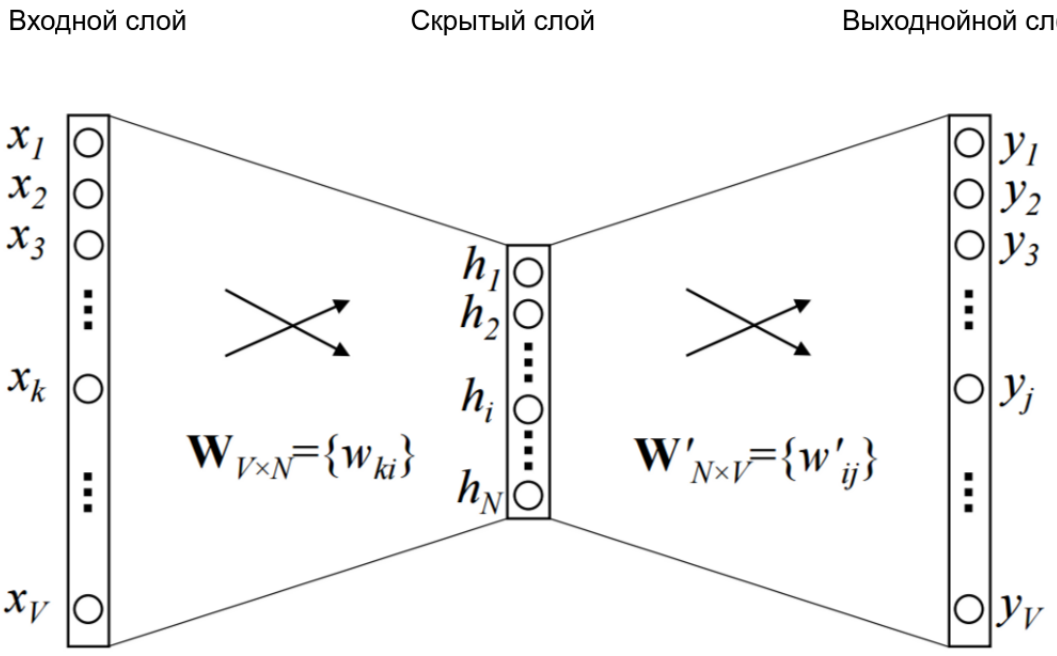

Рисунок 2.2 — Архитектура нейронной сети word2vec CBOW с размером контекста 1

дое слово в словаре – случайный N-мерный вектор. Во время обучения алгоритм формирует оптимальный вектор для каждого слова с помощью метода CBOW. После обучения модели на размеченных данных на выходе получаем обученную модель. Использование модели для анализа конкретного слова даёт на выходе некоторый вектор значений фиксированного размера. Для каждого текста усредняем вектор вероятностей полученных слов.

Листинг 2.1 — Псевдокод алгоритма усреднения слов

```
1 def buildWordVector(text, size):
2     vec = np.zeros(size).reshape((1, size))
3     count = 0.
4     for word in text:
5         try:
6             vec += model[word].reshape((1, size))
7             count += 1.
8         except KeyError:
9             continue
```

```
10      if count != 0:
11          vec /= count
12      return vec
```

Для оценки тональности текста необходимо перевести полученные вектора слов в некое число.Для это понадобится бинарный классификатор, который на выходе будет давать вероятность того, что эмоция текста негативна. Для этого применим вероятностный калибратор, основанный на логистической регрессии(sklearn.calibration.CalibratedClassifierCV из библиотеки scikit-learn). Преобразуем наш исходный датасет в пары вида: вектор чисел - результат. После этого обучим наш классификатор на новом наборе данный. Обученный классификатор получает на вход некоторый текст, а на выходе даёт вероятность того,что текст является негативным.

2.3 Нормализация данных

Полученный набор оценок для каждого текста, а так же все статистические свойства нормализуем при помощи min-max нормализации по формуле 1.24.

2.4 Выбор оптимального набора параметров

В нашем набора данных имеется набор статистических параметров, выберем из них N наиболее значимых параметров при помощи фильтр-метода Пирсона1.17. В нашем датасете имеется Y следующих параметров:

— Рейтинг ATP 1-го игрока
— Рейтинг ATP 2-го игрока
— Где проводится турнир: в помещении или на улице
— Покрытие корта
— Максимальный коэффициент ставки на первого игрока
— Максимальный коэффициент ставки на второго игрока
— Средняя ставка на первого игрока
— Средняя ставка на второго игрока
— Стадия турнира

Из этого набора признаков необходимо выделить N оптимальных свойств(на правила данный момент для выбора оптимальных количествой свойств являются открытым вопросом в современной науке[29]), поэтому оптимальное количество слов подбирается эмпирическим путем).

2.5 Прогнозирование результатов

В нейронную сеть на вход подаются нормализованные статистические данные, совмещенные с текстовыми данными. Текстовые данные переведены в численный вид и нормализованы. Нейронная сеть для прогнозирования результатов теннисных матчей имеет следующие параметры параметры:

— Количество слоёв - 3, из них 2 скрытых.

— Количество нейронов на каждом слое: 64->32->1

— Функции активации по слоям: линейный выпрямитель(англ. relu)->линейный выпрямитель->сигмоида

Листинг 2.2 — Псевдокод нейронной сети для осуществления прогнозов

```
1
2 network = models.Sequential()
3 network.add(layers.Dense(units=64, activation='relu',
      input_shape=(len(features.columns),)))
4 network.add(layers.Dense(units=32, activation='relu'))
5 network.add(layers.Dense(units=1, activation='sigmoid'))
6
7 network.fit(train_features, train\_target,
8 epochs=1000, verbose=0, batch\_size=128,
9 validation_data=(test_features, test_target), callbacks=[es, mc])
```

3 Технологический раздел

3.1 Выбор средств разработки

3.1.1 Выбор целевой платформы

Программный продукт не затачивается под работу на конкретной ОС. Его корректная работа протестирована на ОС Windows и OS Linux Данный выбор обусловлен широкой распространенностью ОС Windows и Linux, большим количеством средств разработки для данных платформ, что дает возможность выбирать язык программирования и сопутствующие инструменты, ориентируясь на возможность простого и надежного решения рассматриваемой задачи, а не на ограничения целевой платформы.

3.1.2 Выбор языка программирования

Для написания комплекса программ создания прогнозов был выбран язык Python. Python - универсальный мультипарадигменный скриптовый язык программирования. Так же для Python написано огромное количество библиотек машинного обучения, а так же для работы с тектовыми данными, что особенно важно для данной работы. Преимущества данного языка:

— Кроссплатформенность. Python – это интерпретируемый язык, его интерпретаторы существуют для многих платформ. Поэтому с запуском его на любой ОС не должно возникнуть проблем.

— С Python доступно огромное количество сервисов, сред разработки, и фреймворков. Легко можно найти подходящий продукт для работы.

— Возможность подключить библиотеки, написанные на C. Это позволяет повысить эффективность, улучшить быстродействие.

Так же для работы с данными использовался языка bash. Bash - это мощный и простой в использовании язык сценариев. Он позволяет легко перемещать курсор и редактировать текст команды в командной строке, поддерживает историю команд: дает возможность повторить или при необходимости изменить команду, которая была введена в командной строке ранее. Позволяет легко указать команде, откуда брать входные данные и куда направлять выходные данные. Поддерживаются псевдонимы - создание кратких обозначения для однострочных команд.

3.1.3 Выбор среды разработки и отладки

Из-за использования сразу нескольких языков, в процессе работы над программным комплексом, было использовано несколько сред разработки. Pycharm - современная кросс-платформенная среда разрабтки, которая предоставляет широкие возможности разработки программ на python, а так

же их отладки. Pycharm так же поддерживает работу со сторонними плагинами, что позволило использовать плагин работы с системой контроля версий.

В процессе работы с различными библиотеками приходится сталкиваться с различными языками программирования, например, с C. Для работы с различными вспомогательными языками был использован текстовый редактор Visual Studio Code. Он имеет плагины для поддержки большинства современных языков программирования, а набор плагинов к текстовому редактору позволяет осуществлять запуск и отладку кода прямо из редактора, а так же позволяет осуществлять автоматическое форматирование кода.

Для работы с bash использовался nano - консольный текстовый редактор для Unix и Unix-подобных операционных систем, основанный на библиотеке curses. В настоящее время включен в дистрибутивы Ubuntu по умолчанию и в установке не нуждается.

3.2 Система контроля версий

В процессе разработки программы использовалась система контроля версий Git. Система контроля версий позволяет вносить в проект атомарные изменения, направленные на решения каких-либо задач. В случае обнаружения ошибок или изменения требований, внесенные изменения можно отменить. Кроме того, с помощью системы контроля версий решается вопрос резервного копирования. Особенности Git:

— Предоставляет широкие возможности для управления изменениями проекта и просмотра истории изменений

— Данная система контроля версий является децентрализованной, что позволяет иметь несколько независимых резервных копий проекта.

— Поддерживается хостингом репозиториев GitHub и Gitlab.

— Поддерживается средой разработки Pycharm и Visual Studio Code.

3.3 Формат файлов

На вход программе для осуществления прогнозов подаётся англоязычный текст длинною от 10 до 2000 символов.

Листинг 3.1 — Пример входных данных для программы оценки тональности текста

```
1 I thought I started a little slow, but I thought he played really
    well today. I knew his game plan, I got onto it pretty quickly.
    He directed most of his serves to my forehand. He hasn't really
    done that in the past. It's definitely my weaker return. He was
    definitely trying to stay away from my backhand return a lot.
2
3 But I thought just on big points, I mean, he played well. Hit his
    forehand extremely well. When Rafa plays well, he hits his
```

```
    forehand line extremely well. I thought today he was on fire
    with that shot. Every time he redirected, he was hitting balls
    just within the line, so...
4
5 I thought it was a high-level match. Two tiebreaks. I played a
    couple loose points here or there. That's all it takes against a
    player like that. He was just too good today.
```

Так же программе на вход подаются данные о двух спортсменах в фомате csv, где вместо F_name% подставляется название соотвествующей метрики;

Листинг 3.2 — Пример входного файла программы поиска аномалий для проверки корректности алгоритма

```
1 Surname1,Surname2, F_name1,F_name2,F_name3, F_name4,F_name4,F_name5
2 Player1,Player2 1,14.23,1.71,2.43,15.6
```

В начале строки две указываются две фамилии спортсменов без пробелов. После этого набор не нормализованных статистических данных.

3.4 Используемые библиотеки

Scikit-learn - это библиотека на Python, которая предоставляет множество неконтролируемых и контролируемых алгоритмов обучения. Содержит большое количество функция для работы с данными.

Keras - высокоуровневая библиотека на Python, представляющая собой надостройку над библиотекой TensorFlow. Позволяет при помощи высокоуровневых абстракций конструировать нейронные сети.

Pandas - библиотека на Python, позволяющая осуществлять манипуляции с данными, предоствляет удобный доступ к табличным данным.

3.5 Установка программного обеспечения

Программа поставляется в двух вариантах:

— В виде docker-образа
— В виде исходных кодов

В случае использования поставки docker-образа необходимо иметь установленный на компьютере Docker не менее 17-ой версии.

В случае использования исходных программных кодов, необходимо иметь на компьютере уставновленный интерпретатор Python не менее версии 3.7

3.6 Пример работы программы

Программа запускается из командной строки. При этом нужно указать три обязательных аргумента:

— interview1 - текст интервью первого спортсмена
— interview2 - текст интервью второго спортсмена
— statistics - статистические данные о спорсменах

Листинг 3.3 — Команда запуска програмы

```
1 python main.py --interview1 text1 --interview2 text2 --staticstics
    stats.csv
```

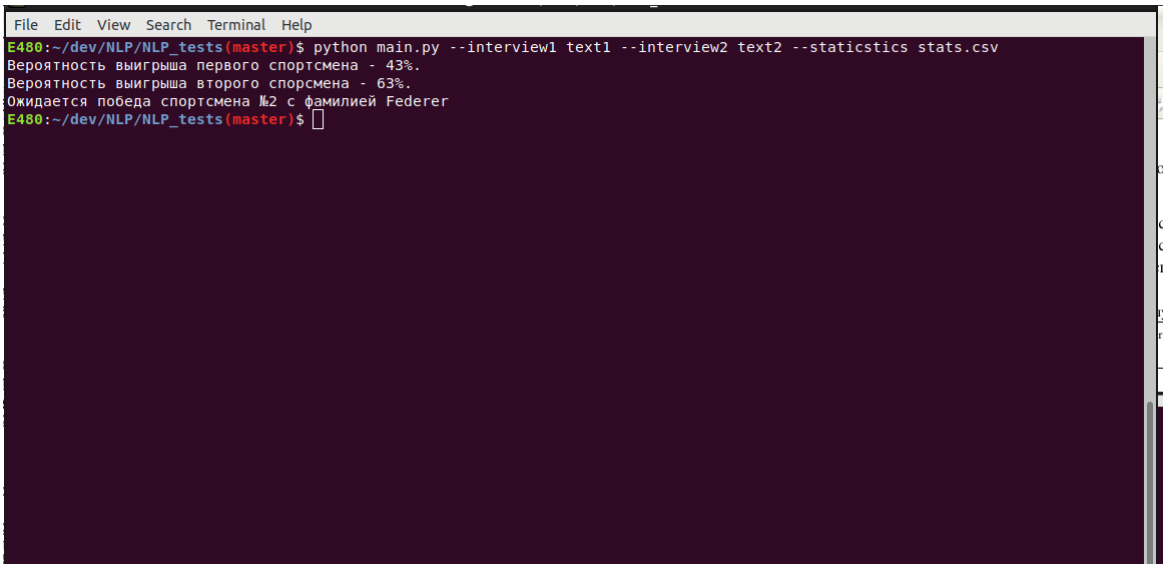

Рисунок 3.1 — Пример работы программы

4 Исследовательский раздел

Для проверки рабоспособности алгоритмов прогнозирования на неразмеченных данных, алгоритмы проверяюлись на размеченных данных. Информация о матчах WTA(международная женская теннистная ассоциация) за период 2007-2019 года была взята с сайта data.world[30], информация о матчах ATP(мужская теннисная ассоциация) была взята с сайта opendatasoft.com[31]. Данные о ставочных коэффициентах были взяты с сайта oddsportal.com[32], теннисные интервью были взяты с сайта ASAP Sport's[33], а так же из статьи Liye Fu[34].В некоторых случаях в качестве предматчевого интервью использовалось послематчевое интервью предыдущего матча. Все вопросы журналистов были вырезаны из интервью. Оставлены только ответы спорсменов на вопросы журналистов Так же для корректировки статистических данных использовался портал tennis-data[35]. Набор данных разбивался на данные для обучения и данные для тестирования в пропорции 4 к 1. На основе сбора данных с вышеприведенных источников был сформирован датасет следующих размеров: 533 матча WTA и 1202 матча ATP.

4.1 Поиск оптимального набора статистических свойств

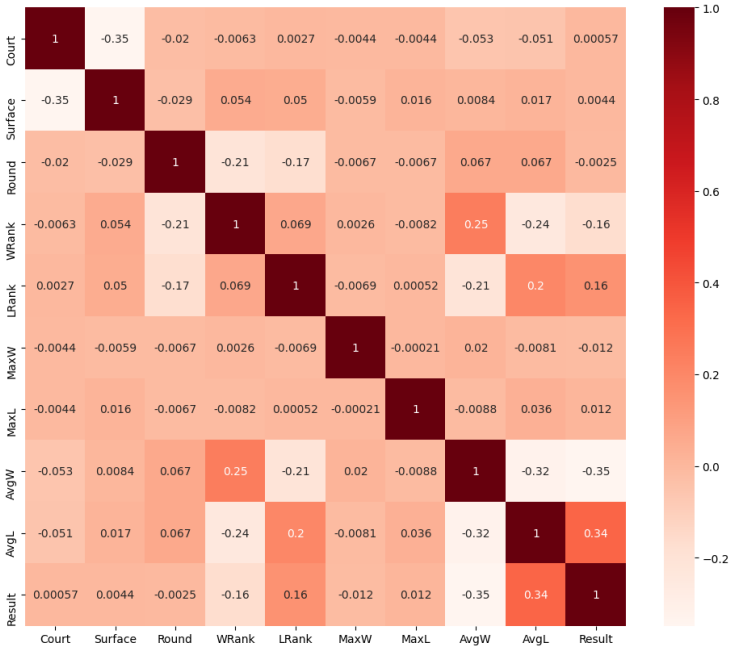

Рисунок 4.1 — Тепловая карта корреляции Пирсона

Таблица 4.1 — Показатели независимости свойств

Свойство	Показатель независимости
В помещении/На улице	0.000569
Стадия турнира	0.002458
Покрытие корта	0.004447
Максимальная ставка на игрока 1	0.011954
Максимальная ставка на игрока 2	0.012071
Ранг игрока 2	0.158269
Ранг игрока 1	0.164985
Средняя ставка на игрока 1	0.342577
Средняя ставка на игрока 2	0.346160

Построим тепловую карту корреляций переменных на основе метода Пирсона рисунок 4.1. На её основе отберем 5 наиболее независимых свойств в таблице 4.1.

4.2 Влияние тональности на точность прогнозов

Была обучена и протестирована нейронная сеть с 5-ю статистическими параметрами и с 5 статистическими параметрами + 1 текстовым(итоговый вариант работы). Из полученных результатов можно заметить, что добавле-

Таблица 4.2 — Влияние тональности на точность прогнозов

Количество свойств	Точность ATP	Точность WTA	Всего
5 статистических	65.14%	68.86%	66.28%
5 стат. + тональность	66.40%	72.64%	68.29%

ние столбца тональности незначительно увеличило точность прогнозов в ATP и почти на 4% улучшилась точность прогнозов в WTA.

4.3 Сравнение алгоритмов прогнозирования теннисных матчей

Сравним эффективность прогнозов полученного метода с другими аналогичными работами. В некоторых работах не указана итоговая точность прогнозирования. Тогда точностью для данной работы считалась общее количество успешно предсказанных матчей, подёлённое на общее число проанализированных матчей[36]. В некоторых работах не уточняется общее число проанализированных матчей. Исходя из вышеприведенных данных можно за-

Таблица 4.3 — Сравнение методов прогнозирования результатов теннисных матчей

Автор работы	Метод	Точность	Кол-во матчей	Турниры
McHale и Morton, 2011 [16]	Модель Бредли-Терри	66.90%	-	ATP
Scheibehenne и Bröder, 2007[36]	Агрегирование рейтингов игроков	70.06%	127	ATP
Knottenbelt, Spanias, Madurska, 2012[37]	Иерархическая модель Маркова	69.38%	686	ATP & WTA
Gu, Saaty,2019[38]	Метод аналатич. сетей	85.10%	94	ATP & WTA
Метод данной работы	Нейронные сети	68.29%	347	ATP & WTA

метить, что точность прогнозов уменьшается с увеличением количества анализируемых данных.

Так же проведем сравнение алгоритмов прогнозирования для различных ассоциаций турниров. Исходя из полученных результатов метрик точно-

Таблица 4.4
Сравнение методов прогнозирования результатов теннисных матчей с разбивкой по ассоциацям

Автор работы	Точн. ATP	Точн. WTA	ATP число матч.	WTA число матч.
McHale и Morton, 2011 [16]	66.90%	-	-	-
Scheibehenne и Bröder, 2007[36]	70.06%	-	127	-
Knottenbelt, Spanias, Madurska, 2012[37]	70.30%	68.75%	270	416
Gu и Saaty,2019[38]	87.4%	80.6%	66	31
Метод данной работы	66.40%	72.64%	241	106

сти для различных ассоциаций, можно рекомендовать использовать данный метод в для прогнозирования турниров WTA и в некоторых случаях для прогнозирования турниров ATP.

Заключение

В данной работе были исследованы различные алгоритмы прогнозирования результатов теннисных матчей, изучены их достоинства и недостатки. Был предложен и реализован новый прогнозирования результатов теннисных матчей, основанный на совмещении статистической и текстовой априорной информации о матчей. Даны рекомендации к использованию нового метода.

Список использованных источников

1. *Справочник технического переводчика*. [Электронный ресурс] / Справочник технического переводчика. — 2018. URL:https://technical_translator_dictionary.academic.ru/8737/apiornaya, дата обращения: 10.03.2020.

2. *Международная теннисная федерация*. IFT[Электронный ресурс] / Международная теннисная федерация. URL:https://www.itftennis.com/, дата обращения: 10.03.2020.

3. *Википедия*. [Электронный ресурс] / Википедия. URL:https://ru.wikipedia.org/wiki/Rating_ATP, дата обращения: 10.05.2020.

4. *Вичугова, Анна*. Отберем то, что нужно Data Mining: как сформировать датасет для машинного обучения [Электронный ресурс] / Анна Вичугова. — 2019. URL:https://www.bigdataschool.ru/bigdata/dataset-data-preparation.html, дата обращения: 10.03.2020.

5. *Малых, Валентин*. Чудесный мир Word Embeddings: какие они бывают и зачем нужны? / Валентин Малых. — 2017. URL:https://habr.com/ru/company/ods/blog/329410/, дата обращения: 09.05.2020.

6. *Крутиков, Александр*. Прогнозирование спортивных результатов в индивидуальных видах спорта с помощью обобщенно-регрессионной нейронной сети / Александр Крутиков. — Молодой ученый. — 2018. — №12., 2018. — с.22-26.

7. *J., Kahn*. Neural Network Prediction of NFL Football Games [Электронный ресурс] / Kahn J. — 2003. URL:http://homepages.cae.wisc.edu/~ece539/project/f03/kahn.pdf, дата обращения: 10.03.2020.

8. *Zdravevski E., Kulakov A.* System for Prediction of the Winner in a Sports Game / Kulakov A. Zdravevski E. — ICT Innovations 2009 — 2010., 2010. — с.55-63.

9. *Keller, J.B.* Tie point strategies in badminton, preprint / J.B. Keller. — 2003.

10. *Enrick, J.R.* Optimal strategies at decision points in singles squash / J.R. Enrick. — Quart. Exercise Sport, 1976.

11. *Burych, B.P. Hsi и D.M.* Games of two players / B.P. Hsi и D.M. Burych. — Journal of the Royal Statistical Society 22(1), 1971.

12. *Crews, W.H. Carter и S.L.* An analysis of the game of tennis / W.H. Carter и S.L. Crews. — Journal of the American Statistical Association 28(4), 1974.

13. *Pollard, G.H.* An analysis of classical and tie-breaker tennis / G.H. Pollard. — Journal of the Australian Mathematical Society 25, 1983.

14. *F.Lassen, R.Magnus*. Are points in tennis independent and identically distributed? Evidence from a dynamic binary panel data model / R.Magnus F.Lassen. — Journal of the American Statistical Association 96(454), 2001.

15. *Terry M.E., Bradley R.A.(*. Rank analysis of incomplete block designs I: the method of paired comparison / Bradley R.A.(Terry M.E. — Biometrika, No. 39, 1952. — c.324-330.

16. *Morton A., McHale I.* A Bradley-Terry type model for forecasting tennis match results / McHale I. Morton A. — International Journal of Forecasting 27, 2011. — c.620.

17. *GlickNan, M.E.* Parameter estimation in large dynamic paired comparison experiments / M.E. GlickNan. — Applied Statistics, 48(3), 1999.

18. *S.Clarke, D. Dyte*. Using official ratings to simulate major tennis tournaments / D. Dyte S.Clarke. — International Transactions in Operational Research 7(6), 2000. — c.585–594.

19. *S. Ma C. Liu, Y. Tan*. Winning matches in Grand Slam men's singles: an analysis of player performance-related variables from 1991 to 2008 / Y. Tan S. Ma, C. Liu. — Journal of sports sciences, 31(11), 2013. — c.1147–1155.

20. *A. Somboonphokkaphan S. Phimoltares, C. Lursinsap.* Tennis Winner Prediction based on Time-Series History with Neural Modeling / C. Lursinsap. A. Somboonphokkaphan, S. Phimoltares. — International Multi-Conference of Engineers and Computer Scientists, Vols I and II, 2009. — c.127-132.

21. *Sipko, Michal*. Machine Learning for the Prediction of Professional Tennis Matches / Michal Sipko. — Imperial College London [Электронный ресурс] URL:https://www.doc.ic.ac.uk/teaching/distinguished-projects/2015/m.sipko.pdf, 2015. — дата обращения:29.03.2020.

22. *Shaikh, Raheel*. Feature Selection Techniques in Machine Learning with Python[Электронный ресурс] / Raheel Shaikh. URL:https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7c дата обращения: 25.04.2020.

23. *Valeria Fonti*. Feature Selection using LASSO[Электронный ресурс] / Valeria Fonti. URL:https://beta.vu.nl/werkstuk-fonti_tcm235-836234, дата обращения: 25.04.2020.

24. *Mayo, Matthew*. Data Representation for Natural Language Processing Tasks / Matthew Mayo. — 2018. URL:https://www.kdnuggets.com/2018/11/data-representation-natural-language-processing.html, дата обращения: 25.04.2020.

25. *"Tomas Mikolov Kai Chen, Greg Corrado*. "Efficient estimation of word representations in vector space-/ Greg Corrado "Tomas Mikolov, Kai Chen, Jeffrey Dean". — 2013.

26. *Tomas Mikolov Kai Chen, Greg Corrado*. "Distributed representations of words and phrases and their compositionality-/ Greg Corrado Tomas Mikolov, Kai Chen, Jeffrey Dean. — 27th Annual Conference on Neural Information Processing System, 2013.

27. *Morin F., Bengio. Y*. Hierarchical Probabilistic Neural Network Language Model / Bengio. Y Morin F. — Aistats № 5, 2005.

28. *Андрей, Гахов*. Интеллектуальный анализ данных / Гахов Андрей. — Харьковский национальный университет имени В.Н. Карамзина, 2014.

29. *Peter C. Austin, Ewout W. Steyerbergd*. The number of subjects per variable required in linear regression analyses / Ewout W. Steyerbergd Peter C. Austin. — Journal of Clinical Epidemiology 68, 2015. — с.627-636.

30. *WOEBKENBERG, TYLER*. WTA Match Data[Электронный ресурс] / TYLER WOEBKENBERG. — 2020. URL:https://data.world/tylerudite/wta-match-data, дата обращения: 26.05.2020.

31. Tennis ATP Results since 2001 [Электронный ресурс]. — 2016. URL:https://public.opendatasoft.com/explore/dataset/atp/information/, дата обращения: 26.05.2020.

32. Tennis Bet Coefficients Info [Электронный ресурс]. — 2020. URL: https://www.oddsportal.com/tennis/, дата обращения: 26.05.2020.

33. Press-conference transcripts [Электронный ресурс]. — 2020. URL:http://www.asapsports.com/showcat.php?id=7&event=yes, дата обращения: 26.05.2020.

34. *Fu, Liye*. Tie-breaker: Using language models to quantify gender bias in sports journalism / Liye Fu, Cristian Danescu-Niculescu-Mizil, Lillian Lee. — IJCAI workshop on NLP meets Journalism, 2016.

35. Tennis statistics [Электронный ресурс]. — 2020. URL:http://www.tennis-data.co.uk/, дата обращения: 26.05.2020.

36. *Benjamin Scheibehenne, Arndt Broder*. "Predicting Wimbledon 2005 tennis results by mere player name recognition-/ Arndt Broder Benjamin Scheibehenne. — International Journal of Forecasting 23, 2013.

37. *William J. Knottenbelt Demetris Spanias, Agnieszka M. Madurska*. "A common-opponent stochastic model for predicting the outcome of professional tennis matches-/ Agnieszka M. Madurska William J. Knottenbelt, Demetris Spanias. — Computers and Mathematics with Applications 64, 2012.

38. *Wei Gu, Thomas Saaty*. "Predicting the Outcome of a Tennis Tournament: Based on Both Data and Judgments-/ Thomas Saaty Wei Gu. — Journal

of Systems Science and Systems Engineering, 2019.

Приложение А Приложение А

Листинг А.1 — Код создания модели Word2Vec

```
import gensim
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from gensim.models.word2vec import Word2Vec
from sklearn.preprocessing import scale
from sklearn.calibration import CalibratedClassifierCV

import numpy as np
import os
import random
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from clText import cleanText

data_path = "data/"
pos_files = os.listdir(data_path + "pos/")
neg_files = os.listdir(data_path + "neg/")

for pos_file in pos_files:  # only for 1 file
    with open(data_path + "pos/" + pos_file, 'r') as infile:
    pos_reviews = infile.readlines()

for neg_file in neg_files:
    with open(data_path + "neg/" + neg_file, 'r') as infile:
    neg_reviews = infile.readlines()

y = np.concatenate((np.ones(len(pos_reviews)),
   np.zeros(len(neg_reviews))))

x_train, x_test, y_train, y_test = train_test_split(
np.concatenate((pos_reviews, neg_reviews)), y, test_size=0.2)

x_train = cleanText(x_train)
x_test = cleanText(x_test)

n_dim = 300
# Initialize model and build vocab
imdb_w2v = Word2Vec(size=n_dim, min_count=10)
imdb_w2v.build_vocab(x_train)

# Train the model over train_reviews (this may take several minutes)
imdb_w2v.train(x_train, total_examples=imdb_w2v.corpus_count,
epochs=imdb_w2v.iter)
```

```
43
44
45 def buildWordVector(text, size):
46     vec = np.zeros(size).reshape((1, size))
47     count = 0.
48     for word in text:
49     try:
50         vec += imdb_w2v[word].reshape((1, size))
51         count += 1.
52     except KeyError:
53     continue
54     if count != 0:
55         vec /= count
56     return vec
57
58
59 train_vecs = np.concatenate([buildWordVector(z, n_dim) for z in
       x_train])
60 train_vecs = scale(train_vecs)
61
62 # Train word2vec on test tweets
63 imdb_w2v.train(x_test, total_examples=imdb_w2v.corpus_count,
64 epochs=imdb_w2v.iter)
65
66 test_vecs = np.concatenate([buildWordVector(z, n_dim) for z in
       x_test])
67 test_vecs = scale(test_vecs)
68
69 lr = CalibratedClassifierCV()
70 lr.fit(train_vecs, y_train)
71
72 print('Test Accuracy: %.2f' % lr.score(test_vecs, y_test))
73 pred_probas = lr.predict_proba(test_vecs)[:, 1]
```