МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
(СГУ)

Кафедра математической
кибернетики и компьютерных наук

ТОНОВЫЙ АНАЛИЗ КОММЕНТАРИЕВ В СОЦИАЛЬНЫХ СЕТЯХ С ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ СЕТЕЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Ивановой Александры Денисовны

Научный руководитель
к. ф.-м. н., доцент _____ А. С. Иванов

Заведующий кафедрой
к.ф.-м.н., доцент _____ С. В. Миронов

Саратов 2021

ВВЕДЕНИЕ

Каждый день в мире происходит множество событий. В настоящее время почти у каждого человека есть доступ в Интернет, где практически везде можно свободно высказывать мнение об той или иной ситуации. Благодаря множеству комментариев в социальных сетях можно оценить отношение пользователей к тому или иному событию.

Обрабатывать все эти тексты вручную невозможно, ибо с ростом сети Интернет и растет количество данных. Однако эту задачу можно автоматизировать, и уже существует огромное количество методов, которые позволяют обрабатывать естественный язык. Анализ тональности текстов является очень важной задачей в обработке естественного языка. Благодаря ему можно анализировать ситуацию и отслеживать реакцию пользователей на какое либо событие. Результаты могут быть использованы в таких сферах как политика, маркетинг и другие важные сферы. Именно поэтому автоматизация анализа тональности текстов очень актуально в настоящее время.

Целью данной работы является рассмотреть методов тонового анализа текстов и создание своего тонового классификатора комментариев в социальной сети «ВКонтакте» с помощью рекуррентных нейронных сетей.

Были поставлены следующие задачи

- рассмотреть методы тонового анализа текста;
- построение нейронных сетей, в т.ч. рекуррентных;
- найти данные для обучение;
- обучить сеть и создать приложение, которое будет оценивать настроение пользователей в определенный промежуток времени

Работа состоит из введения, двух глав, заключения, списка из 21 источников и двух приложений. В главе «Тоновый анализ текста и нейронные сети» рассматриваются основные методы тонового анализа текста, а так же построения нейронных сетей. Рассматриваются различные архитектуры сетей, а так же методы оптимизации обучения. В главе «Практическая реализация» рассматривается структура нейронной сети, а так же приложение, для получения данных и вывода графиков. Описывается и иллюстрируется работа приложения.

1 Тоновый анализ текста и нейронные сети

В данном разделе приведен обзор основных методов анализа тональности текста, а так же построения нейронных сетей

1.1 Методы тонального анализа текста

Эмоциональная оценка, которая выражается в тексте, называется тональностью. Данная оценка в целом основывается на тональности каждой единицы в тексте, тональность всего текста зависит от тональности каждого отдельного слова или определенных словосочетаний. Обычно под тональным анализом подразумевается классификация фрагментов текста на позитивные и негативные.

Выделяют следующие группы подхода к тоновому анализу текста

- основанные на словарях и правилах;
- основанные на машинном обучении;

1.1.1 Подходы, основанные на правилах и словарях

Подходы, **основанные на правилах и словарях** появились самыми первыми. В данных методах используются в основном вручную заданные правила, которые на основе эмоциональных ключевых слов из специально размеченных словарях, а так же их совместное использования с другими рядом стоящими словами, рассчитывают эмоциональную окраску какого либо текста.

При данных подходах сначала создается словарь слов и выражений, у каждого из которых должна быть определена оценка тональности.

Для составления словаря могут быть использованы следующие источники:

- существующие словари оценочной лексики, которые можно сопоставить с имеющимся корпусом текстов и отобрать необходимую лексику, относящуюся к данной предметной области
- корпуса текстов, в которых происходит отбор оценочной лексики;
- существующие словари на других языках, переведенные на русский;
- вручную созданный словарь, при создании которого был привлечен эксперт;

Далее анализируемый текст сравнивается со словарем. Для каждого предложения в тексте выдается собственная оценка тональности, которая зависит от тональности каждого слова, а так же их контекста. При анализе могут

быть использованы специальные правила для определения контекста.

1.1.2 Подходы, основанные на машинном обучении

Подходы, основанные на машинном обучении разделяют на два типа.

- обучение с учителем; Тексты вручную отбираются и размечаются по тональности, далее текст обрабатывается для подачи его в классификатор и производится его обучене. После этого производится обучение классификатора
- обучение без учителя;

Эффективность анализа тональности, с помощью методов машинного обучения, определяется на некотором тестовом наборе данных.

Классическими алгоритмами для анализа тональности являются наивный байесовский классификатор, дерево решений (Decision Tree), логистическая регрессия и метод опорных векторов. [1]

В последние годы все чаще применяются методы, основанные на глубоком обучении, которые значительно превосходят традиционные методы в анализе тональности. В данном случае используются рекуррентные (RNN) и сверточные(CNN) нейронные сети, также применяется transfer learning(дообучение готовой модели на собственных данных).

При использовании методов машинного обучения можно автоматически извлекать признаки. В простых подходах для представления текста в векторном пространстве обычно используется модель «мешок слов» (bag of words). В более сложных системах для генерирования векторных представлений слов применяются такие модели, как, Word2Vec, GloVe или FastText. Также есть алгоритмы генерирования векторных представлений на уровне предложений или параграфов, которые предназначены для transfer learning в разных задачах обработки естественного языка. К таким алгоритмам относятся ELMo, Universal Sentence Encoder, Bidirectional Encoder Representations from Transformers (BERT), ERNIE и XLNet.

В данной работе будет применяться подход, основанный на машинном обучении с учителем с помощью рекуррентных нейронных сетей. Данный подход был выбран исходя из того, что рекуррентные нейронные сети, наряду с сверточными, показывают самые точные результаты среди всех, рассмотренных выше.

1.2 Нейронные сети

1.2.1 Нейронные сети прямого распространения

Сети прямого распространения (Feedforward neural network) — искусственные нейронные сети, в которых сигнал распространяется строго от входного слоя к выходному. В обратном направлении сигнал не распространяется.

Однослойные сети прямого распространения. Однослойная сеть — сеть, образованная нейронами, которые лежат в одной плоскости. В однослойных нейронных сетях сигналы от входного слоя сразу же поступают на выходной слой, который делает необходимые расчеты. Результаты этих расчетов сразу же отправляются на выходы.

Многослойные сети прямого распространения. Данные сети характеризуются наличием одного или нескольких *скрытых слоев* (hidden layer).

1.2.2 Сверточные нейронные сети

Сверточные сети — это тип нейронной сети которые особенно хороши при анализе изображений. Нейроны в слоях сверточной нейронной сети расположены в трех измерениях: высота, ширина и глубина.

Название данной сети образовалось от названия операции - свертка, которая часто используется для обработки изображений.

Ее можно описать следующей формулой

$$(f * g)[m, n] = \sum_{k,l} f[m - k, n - l] \cdot g[k, l] \tag{1}$$

Здесь f - исходная матрица изображения, g - матрица свертки.

1.2.3 Рекуррентные нейронные сети

Рекурентные сети — вид нейронных сетей, где связи между элементами образуют направленную последовательность. Данные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. Данные сети изначально были предложены для обработки последовательностей однотипных данных. Сейчас данные сети используются в таких задачах, как задачи распознавания текста(обработка последовательности зву-

ков и обработка текстов естественного языка) и задачи компьютерного зрения (обработка видео, некоторые задачи обработки изображений) [2]

Существует несколько разновидностей рекуррентных нейронных сетей

— Один к одному. Является обычной нейронной сетью
— Один ко многим. Может применяться для генерации музыки или задачах описания изображения.
— Многие к одному. Может применяться для определения тональности рецензий.
— Многие ко многим(различные вариации в зависимости от входных и выходных параметров). Может применяться для машинного перевода или классификации слов.

Самая простая рекуррентная нейронная сеть представляет собой цикл for, который повторно использует величины, полученные на предыдущей итерации. $s_t = tanh(U * x_t + W * s_t - 1 + b)$

У данной архитектуры существует существенная проблема: теоретически в каждый момент времени t он должен хранить информацию о входных данных за многочисленные предыдущие интервалы времени, но на практике такое большое количество данных замедляет и полностью останавливает процесс обучения. Это связано с проблемой затухания градиента. [3]

Для решения этой проблемы были созданы LTSM и GRU архитектуры.

LTSM сети - особая разновидность архитектуры рекуррентных нейронных сетей, которые способны к обучению долговременным зависимостям. Данные сети способны долго хранить информацию, и в любой момент получить к ней доступ. [4]

Главный компонент LSTM – это состояние ячейки (cell state). Состояние ячейки напоминает конвейерную ленту. Она проходит напрямую через всю цепочку, участвуя лишь в нескольких линейных преобразованиях. Информация может просто идти далее, не подвергаясь изменениям.

LSTM так же может удалять информацию из состояния ячейки; Данный процесс регулируется такими структурами, как фильтры(gates).

Фильтры позволяют пропускать информацию на основании некоторых условий. Они состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения.

Сигмоидальный слой возвращает числа от нуля до единицы, которые обозначают, какую долю каждого блока информации следует пропустить дальше по сети.

В LSTM присутствует три фильтра, позволяющих защищать и контролировать состояние ячейки.

GRU сети. Являются упрощенным LSTM. Данные виды сетей работают быстрее, чем LSTM, так как имеют меньше параметров из-за отсутствия «выходного фильтра». Эффективность GRU в задачах моделирования полифонической музыки, моделирования речевых сигналов и обработки естественного языка оказалась аналогичной LSTM.

1.2.4 Методы обучения

Обучение ИНС — настройка весов w в соответствии с обучающим множеством (x, c) и важным элементом этой процедуры является способ оценки работы или функция потери (loss function)

В качестве функции потери обычно используется *среднеквадратичная ошибка*

$$E = \frac{1}{2}\sum_{j}(y_j - c_j)^2 \tag{2}$$

где y_j - выход сети номер j, c_j - правильный ответ для выхода j

Кроме того, для сетей с выходным слоем softmax используют среднюю кросс-энтропию

$$E = \frac{1}{k}\sum_{i=1}^{k}(-log(p_i)) \tag{3}$$

1.2.5 Методы оптимизации обучения

В глубоком обучении существует такая переменная, как loss (потеря), показывающая насколько плохо модель работает в данный момент. Для определения значения данной переменной задается специальная функция - функция потери. Процесс минимизации (или максимизации) любого математического выражения называется оптимизацией, и теперь нам нужно увидеть, как мы можем использовать эти методы оптимизации для нейронных сетей.

Самым простым алгоритмом оптимизации является градиентный спуск, или же стохастический градиентный спуск. Представляет собой итеративный

процесс, а значить обновляться значение каждого веса будет несколько раз, прежде чем потеря сойдется в подходящем значении.

$$w_t = w_{t-1} - \alpha * \frac{\partial loss}{\partial w_t} \tag{4}$$

где loss - значение функции потерь,

а α - скорость обучения.

Данное значение показывает скорость оптимизации сети. Если скорость обучения большая, то минимум потерь достигается быстрее, потому что делаются большие шаги, однако в результате можно не достичь минимума, так как из-за больших шагов его можно перескочить. Меньшая скорость обучения решит эту проблему, но потребуется много шагов, чтобы потери сети снизились до хорошего значения. Следовательно, необходимо поддерживать скорость обучения на оптимальном уровне.

Однако при использовании обычного градиентного спуска может возникнуть большая проблема. Градиентный спуск будет работать только до локальных минимумов, поскольку частная производная (градиент) вблизи локальных минимумов близка к нулю. Следовательно, он будет оставаться там после достижения локальных минимумов и не будет пытаться достичь глобальных минимумов.

Однако эту проблему может помочь решить метод моментов.

Оптимизатор импульса В данном оптимизаторе собирается некоторая информация о предыдущих обновлениях, через которые прошел вес, перед выполнением текущего обновления. По сути, если вес постоянно движется в определенном направлении (увеличивается или уменьшается), он будет медленно накапливать некоторый «импульс» в этом направлении. Следовательно, когда он сталкивается с некоторым сопротивлением и фактически должен идти в противоположном направлении, он все равно будет продолжать движение в исходном направлении в течение некоторого времени из-за накопленного импульса.

Алгоритмы оптимизации, в которых параметр скорости обучения постоянно меняется для адаптации весов относятся к категории *адаптивных оптимизаторов*

Оптимизатор AdaGrad (Adaptive Gradient) - это метод, учитывающий историю значений градиента, который необходим для того, чтобы подстраивать скорость обучения для каждого параметра, исходя из того, как часто он встречается [5]. В данном оптимизаторе Скорость обучения изменяется во время каждого обновления таким образом, что она будет уменьшаться, если вес обновляется слишком часто за короткий промежуток времени, и увеличиваться, если вес обновлялся не часто.

Оптимизатор RMSProp. Данный оптимизатор является улучшенной версией AdaGrad так как изменена формула обновления хэша, которая вместо полной суммы обновлений использует усредненный квадрат градиента В эту формулу добавляется новый параметр

Оптимизатор Adam - является комбинацией RMSProp и оптимизатора импульсов. [6]

В практической части данной работы будет использован именно оптимизатор Adam. Он является одним из самых эффективных алгоритмов оптимизации в обучении нейронных сетей.

1.3 Векторное представление слов

На вход сети подается текст, однако модели глубокого обучения не могут принимать на вход обычный текст, так как работают только с числовыми тензорами. Для того чтобы можно было подать на вход сети текст, нужно преобразовать его в эти самые числовые тензоры. Сделать это можно несколькими способами:

- разбить текст на слова и преобразовать каждое слово потом в вектор;
- разбить текст на символы и преобразовать каждый символ в вектор;
- извлечь n-граммы из слов и преобразовать каждую n-грамму в вектор;

N-граммы и мешки слов N-граммы слов - группы по N последовательных слов, которые можно извлечь из предложения.

Прежде всего, это может помочь в принятии решения, какие N-граммы можно объединить, чтобы сформировать единое целое (например, «Нижний Новгород» или «средняя школа», объединенные в одно слово).

Word Embedding — способ связывания вектора со словом. В отличии от векторов, полученных прямым кодированием - векторные представления слов в этом случае это малоразмерные векторы вещественных чисел.

Есть два способа получения векторных представлений

- Конструирование векторных представлений в процессе решения основной задачи, когда создаются случайные векторы слов и обучаются в процессе;
- Использование предобученных моделей векторных представлений(например word2vec или GloVe);

При конструировании представлений в процессе решения необходимо, чтобы текст документа был очищен и подготовлен таким образом, чтобы каждое слово было закодировано только один раз. Векторы инициализируются небольшими случайными числами и в процессе обучения корректируются посредством обратного распространения ошибки. После обучения на выходе получается пространство векторов, которые предназначены для конкретной задачи.

Однако, если данных не очень много, то лучшим решением будет использовать уже предобученные векторные представления, например word2vec

1.3.1 word2vec

Данная модель представляет собой набор инструментов для расчета векторных представлений. [7] В нем реализуются две основные архитектуры векторизации текста — Continuous Bag of Words (CBOW); — Skip-gram;

Это работает следующим образом: word2vec принимает на вход большой корпус текста и сопоставляет каждое слово с вектором, давая координаты слов на выходе. Сначала он генерирует словарь корпуса, а затем вычисляет векторное представление слов, «изучая» входящие тексты. Векторное представление основано на контекстной близости: слова, которые появляются в тексте рядом с одними и теми же словами, будут иметь близкие векторы. Полученные векторные представления слов используются в задачах обработки естественного языка и в машинном обучении.

2 Практическая реализация

В данном разделе описана практическая реализация нейронной сети и парсера для социальной сети ВКонтакте.

2.1 Предобработка данных и обучение сети

2.1.1 Данные для обучения

Для обучения был выбран корпус коротких текстов Юлии Рубцовой [8]. В качестве источника текстов в данном корпусе была выбрана платформа Twitter. Корпус представляет собой две таблицы, одна из которых содержит негативные твиты в количестве 111923 записей.

2.1.2 Подготовка тренировочных данных

Для того, чтобы упростить обучение необходимо отчистить данные от «мусора», а так же привести их к простой форме. Для этого необходимо:

- привести текст к нижнему регистру;
- удалить ссылки;
- удалить упоминания (т.е. @nickname);
- удалить знаки препинания и другие символы;

Для удаления ссылок, упоминаний и лишних символов используются регулярные выражения. Для всего остального стандартные функции для строк.

После подготовки сырых данных, необходимо разделить их на тренировочную и тестовую выборку. В данном случае данные будут разделены в отношении 70% для тренировочных данных и 30 % для тестовых

2.1.3 Токенизация

Теперь необходимо преобразовать текст в векторное представление. Это необходимо для того, чтобы подавать сеть в Embeding Layer. [9]

Для того, чтобы можно было подавать в Embeding Layer текст, в виде двумерного тензора, необходимо, чтобы все подаваемые последовательности были одинаковой длинны. Е

2.1.4 Подготовка Embedding слоя

В данном случае будет использовано векторное представление слов.

Вместо того, чтобы обучать данный слой на своих данных, которых не так уж и много, можно взять уже готовый обученные векторные представления.

Данное векторное представление было получено с помощью обучения модели Word2Vec [10] не размеченных текстах в корпуса Юлии Рубцовой.

Модель word2vec содержится в библиотеке gensim.

2.1.5 Определение модели

Для построения моделей используется библиотека машинного обучения — Tensorflow [11] Данная библиотека была выбрана потому что проста в применении, а так же имеет достаточно обширную документацию

В первую очередь для необходимой модели нужно определить входной слой, а так же слой с векторным представлением.

Теперь определяются оставшиеся слои и задается модель, состоящая из управляемого рекуррентного блоков(GRU) [12]. В данном проекте была использована рекуррентная модель, так как в задачах обработки естественного языка(NLP) рекуррентные сети показывают большую точность.

Отчет по обучению показывает что примерная точность модели равна 78-79

2.2 Приложения для построения графиков

На основе этого всего было создано приложение, строящее график настроений по заданной дате и теме. Для получения данных была использована библиотека vk_api [13] для python, а так же pandas для хранения данных. Получение данных производится из групп новостной направленности. При создании графического интерфейса была использована библиотека tkinter

ЗАКЛЮЧЕНИЕ

В ходе данной работы были проанализированы методы анализа тональности текста, а так же реализована программа для анализа тональности комментариев в соц. сети ВКонтакте.

Были решены следующие задачи:

- проанализированы методы тоновой классификации;
- рассмотрены различные архитектуры нейронных сетей, в т.ч. рекуррентных;
- обучена рекуррентная сеть
- на основе этой сети создано приложение, оценивающее настроение пользователей в определенный промежуток времени

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Smetanin, S.* The applications of sentiment analysis for russian language texts: Current challenges and future perspectives / S. Smetanin // *IEEE Access*. — 2020. — Vol. 8. — Pp. 110693–110719.

2 *Хайкин, С.* Нейронные сети: полный курс 2-е издание / С. Хайкин. — Москва: Вильямс, 2006.

3 *Bengio, Y.* Learning long-term dependencies with gradient descent is difficult / Y. Bengio, P. Simard, P. Frasconi // *IEEE Transactions on Neural Networks*. — 1994. — Vol. 5, no. 2. — Pp. 157–166.

4 *Hochreiter, S.* Long short-term memory / S. Hochreiter, J. Schmidhuber // *Neural Computation*. — 1997. — Vol. 9, no. 8. — Pp. 1735–1780.

5 *Duchi, J.* Adaptive subgradient methods for online learning and stochastic optimization / J. Duchi, E. Hazan, Y. Singer // *Journal of Machine Learning Research*. — 2011. — Vol. 12, no. 61. — Pp. 2121–2159. `http://jmlr.org/papers/v12/duchi11a.html`.

6 *Kingma, D. P.* Adam: A method for stochastic optimization // 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings / Ed. by Y. Bengio, Y. LeCun. — 2015. `http://arxiv.org/abs/1412.6980`.

7 *Mikolov, T.* Distributed representations of words and phrases and their compositionality / T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean // *CoRR*. — 2013. — Vol. abs/1310.4546. `http://arxiv.org/abs/1310.4546`.

8 *Рубцова, Ю. В.* Построение корпуса текстов для настройки тонового классификатора / Ю. В. Рубцова // *Программные продукты и системы*. — 2015. — Т. 1. — С. 72–78.

9 *Шолле, Ф.* Глубокое обучение на Python / Ф. Шолле. — СПб: Питер, 2018.

10 *Smetanin, S.* Sentiment analysis of product reviews in russian using convolutional neural networks // 2019 IEEE 21st Conference on Business Informatics (CBI). — Vol. 01. — July 2019. — Pp. 482–486.

11 TensorFlow Core v2.5.0 [Электронный ресурс]. — URL: https://www.tensorflow.org/api_docs/python/tf (Дата обращения 10.05.2021). Загл. с экр. Яз. англ.

12 *Cho, K.* Learning phrase representations using RNN encoder-decoder for statistical machine translation / K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, Y. Bengio // *CoRR*. — 2014. — Vol. abs/1406.1078. http://arxiv.org/abs/1406.1078.

13 Библиотека vk_api[Электронный ресурс]. — URL: https://github.com/python273/vk_api (Дата обращения 10.05.2021). Загл. с экр. Яз. рус.