МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теории функций и стохастического анализа

АНАЛИЗ ТЕКСТОВЫХ ДАННЫХ С ПОМОЩЬЮ NLP МОДЕЛИ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССА РАССЛЕДОВАНИЯ НЕПОЛАДОК СИСТЕМЫ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 248 группы
направления 09.04.03 — Прикладная информатика
механико-математического факультета
Терехова Павла Олеговича

Научный руководитель
доцент, к. ф.-м. н., доцент _____ Л. В. Борисова

Заведующий кафедрой
д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2021

ВВЕДЕНИЕ

Актуальность работы обусловлена огромной популярностью нейронных сетей. Они получили широкое распространение. Создание нейронных сетей вызвано попытками понять принципы работы человеческого мозга и, без сомнения, это будет влиять и на дальнейшее их развитие. Однако, в сравнении с человеческим мозгом нейронная сеть сегодня представляют собой весьма упрощенную модель, но несмотря на это весьма успешно используются при решении самых различных задач: автоматизация процесса классификации, автоматизация прогнозирования, автоматизация процесса распознавания, автоматизация процесса принятия решений; управление, кодирование и декодирование информации; аппроксимация зависимостей и др. Хотя решение на основе нейронных сетей может выглядеть и вести себя как обычное программное обеспечение, они различны в принципе, поскольку большинство реализаций на основе нейронных сетей «обучается», а «не программируется»: сеть учится выполнять задачу, а не программируется непосредственно.

Целью магистерской работы, является создание системы для автоматизации процесса расследования неполадок системы с помощью модели для обработки естественного языка, которая выделяет ключевые слова из логов и осуществляет взаимодействие с агрегатором логов.

Объектом исследования являются виды и методы обучения искусственных нейронных сетей, для решения задач обработки языка и выделения ключевых слов.

Предмет исследования - особенности реализации методов обучения нейронных сетей.

Для достижения поставленных целей в работе необходимо создать систему и инфраструктуру:

— придумать и реализовать архитектуру системы с помощью облачных технологий;
— написать программное обеспечение для каждого компонента системы;
— написать алгоритм для считывания логов;

Для этого необходимо решить следующие **задачи**:

— определить необходимые технологии;

— определить необходимые вычислительные ресурсы;
— создать и обучить нейронную сеть;

Теоретической основой послужила совокупность научных работ западных ученых и инженеров конца 20 начала 21 века.

— F.Rosenblatt: The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain (1958)
— Gradient-Based Learning Applied to Document Recognition by Yann LeCun, Leon Botton, Yoshua Bengio and Patrick Haffner
— Extracting and Composing Robust Features with Denoising Autoencoders. Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol, Universite de Montreal, Dept. IRO, CP 6128, Succ. Centre-Ville, Montral, Qubec, H3C 3J7, Canada
— Auto-Encoding Variational Bayes. Diederik P. Kingma , Max Welling Machine Learning Group Universiteit van Amsterdam. arXiv:1312.6114v10 [stat.ML] 1 May 2014.

Основное содержание работы

Выпускная квалификационная работа состоит из введения, шести теоретических разделов и пяти практических, заключения, списка использованных источников и двадцати двух приложений.

Введение содержит основные положения: обоснование актуальности темы работы, формулируется цель, объект и предмет исследования.

В **первом - четвертом** разделах, рассматриваются основные понятия связанные с обработкой естественного языка:

— история NLP;
— применение NLP;
— основы NLP;
— постановка задачи работы;

Определяется понятие естественного языка, его обработки, рассматриваются задачи из этой области. Рассматриваются *несколько способов векторизации текста* как важный элемент в подготовке данных для решения задачи. Определяются их достоинства и недостатки.

Во **пятом - шестом** разделах, рассматриваются основные понятия связанные с системами мониторинга, а так-же несколько моделей:

— определение лог-файлов;
— виды лог-файлов и их расположение;
— определение агрегаторов логов и их виды;
— определение моделей для обработки текста, их виды, архитектуры, сравнение их между собой;

Рекуррентные нейронные сети (Recurrent Neural Networks, RNNs) — популярные модели, используемые в обработке естественного языка (NLP).

Идея RNN заключается в последовательном использовании информации. В традиционных нейронных сетях подразумевается, что все входы и выходы независимы. RNN называются рекуррентными, потому что они выполняют одну и ту же задачу для каждого элемента последовательности, причем выход зависит от предыдущих вычислений. Еще одна интерпретация RNN: это сети, у которых есть "память" , которая учитывает предшествующую информацию. Теоретически RNN могут использовать информацию в произвольно длинных последовательностях, но на практике они ограничены лишь несколькими шагами.

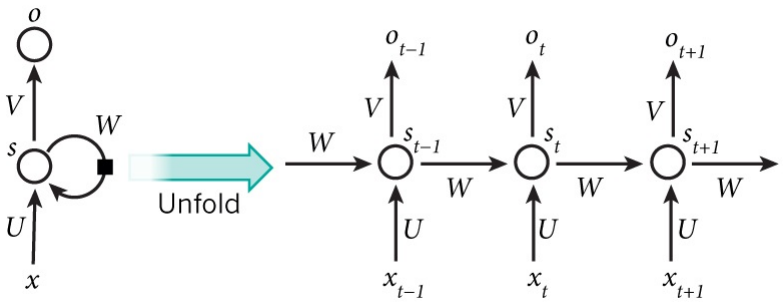

Рисунок 1 – Общая схема рекуррентной нейронной сети и ее развертка (unfold)

Обозначения, представленные на рисунке 2:

— x_t — вход на временном шаге t. Например x_1 может быть вектором с одним горячим состоянием (one-hot vector), соответствующим второму слову предложения;
— s_t — это скрытое состояние на шаге t. Это "память сети". s_t ,как функция, зависит от предыдущих состояний и текущего входа x_t: $s_t = f(Ux_t + Ws_{t-1})$. Где f - функция активации. s_{-1}, которое требуется для вычисление первого скрытого состояния, обычно инициализируется нулем (нулевым вектором);

— o_t — выход на шаге t.

LSTM (long short-term memory, дословно (долгая краткосрочная память) — разновидность архитектуры рекуррентной нейросети, созданная для более точного моделирования временных последовательностей и их долгосрочных зависимостей, чем традиционная рекуррентная сеть. LSTM-сеть не использует функцию активации в рекуррентных компонентах, сохраненные значения не модифицируются, а градиент не стремится исчезнуть во время тренировки. Часто LSTM применяется в блоках по несколько элементов. Эти блоки состоят из 3 или 4 затворов (например, входного, выходного и гейта забывания), которые контролируют построение информационного потока по логистической функции. LSTM были представлены Зеппом Хохрайтер и Юргеном Шмидхубером (Jurgen Schmidhuber) в 1997 году, впоследствии усовершенствованы и популяризированы другими исследователями, хорошо справляются со многими задачами и до сих пор широко применяются.

LSTM специально разработаны для устранения проблемы долгосрочной зависимости. Их специализация — запоминание информации в течение длительных периодов времени, поэтому их практически не нужно обучать.

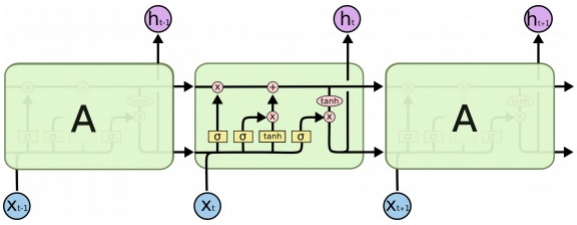

Рисунок 2 – Общая схема LSTM сети

Ключевым понятием LSTM является состояние ячейки: горизонтальная линия, показанная в верхней части рисунка 3.

Состояние ячейки по сути является хранилищем. Оно проходит через всю цепочку, подвергаясь незначительным линейным преобразованиям.

В LSTM уменьшает или увеличивает количество информации в состоянии ячейки, в зависимости от потребностей. Для этого используются тщательно настраиваемые структуры, называемые гейтами.

Гейт — это "ворота пропускающие или не пропускающие информацию. Гейты состоят из сигмовидного слоя нейронной сети и операции поточечного

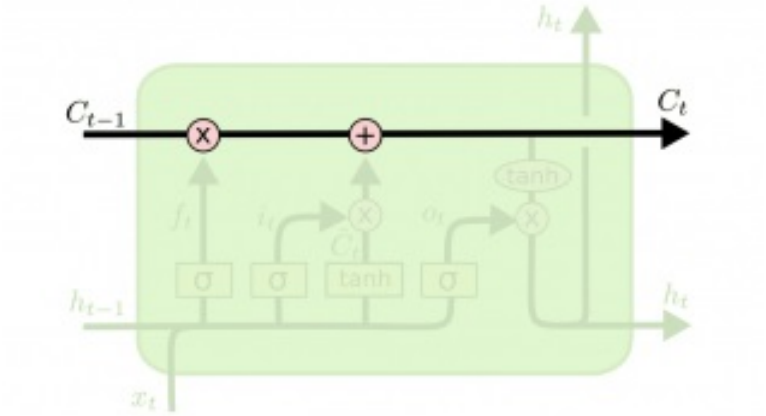

Рисунок 3 – Схема состояния LSTM сети

умножения, схема представлена на рисунке 4.

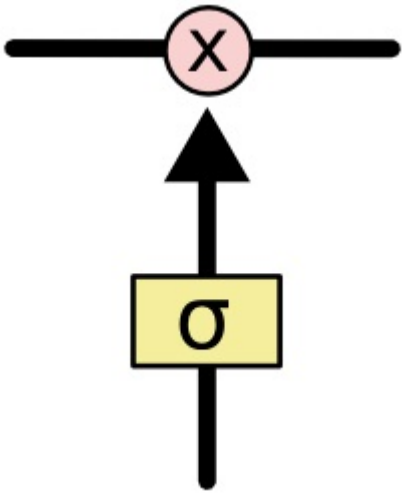

Рисунок 4 – Схема гейта LSTM сети

На выходе сигмовидного слоя выдаются числа от нуля до единицы, определяя, сколько процентов каждой единицы информации пропустить дальше. Значение "0" означает "не пропустить ничего" , значение "1" — "пропустить все" .

LSTM имеет три таких слоя для контроля состояния ячейки.

На первом этапе LSTM нужно решить, какую информацию мы собираемся выбросить из состояния ячейки. Это решение принимается сигмовидным слоем, называемым "слоем утраты" . Он получает на вход h и x и выдает число от 0 до 1 для каждого номера в состоянии ячейки C. Схема представлена на рисунке 5.

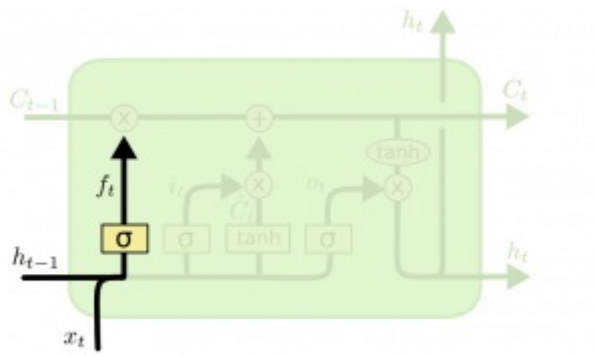

Рисунок 5 – Слой утраты

На следующем шаге нужно решить, какую новую информацию сохранить в состоянии ячейки. Процесс разделяется на две части. Сначала сигмоидный слой, называемый "слоем гейта входа" , решает, какие значения требуется обновить. Затем слой tanh (гиперболический тангенс, как одна из функций активации) создает вектор новых значений-кандидатов C, которые добавляются в состояние. На следующем шаге эти два значения объединяются для обновления состояния.

BERT (Bidirectional Encoder Representations from Transformers) — двунаправленная модель с transformer-архитектурой, с более быстрым подходом на основе механизма внимания.

Большинство конкурентных моделей нейросети передачи последовательности имеют структуру энкодер-декодер.

Энкодер отображает входную последовательность представлений символов $(x_1, ..., x_n)$ в последовательность непрерывных представлений $z = (z_1, ..., z_n)$. Учитывая z, декодер затем генерирует выводную последовательность $(y_1, ..., y_m)$ символов по одному элементу за раз. На каждом этапе модель авторегрессивна, то есть использует ранее сгенерированные символы в качестве дополнительных входных данных при генерации следующего выхода.

Трансформер в данной архитектуре, использует многоуровневый self-attention и point-wise слой, полностью соединенные слои для энкодера и декодера. Подробная схема представлена на рисунке 6.

Энкодер состоит из стека 6 одинаковых слоев. Каждый слой имеет два подслоя. Первый - это self-attention механизм с несколькими входами,

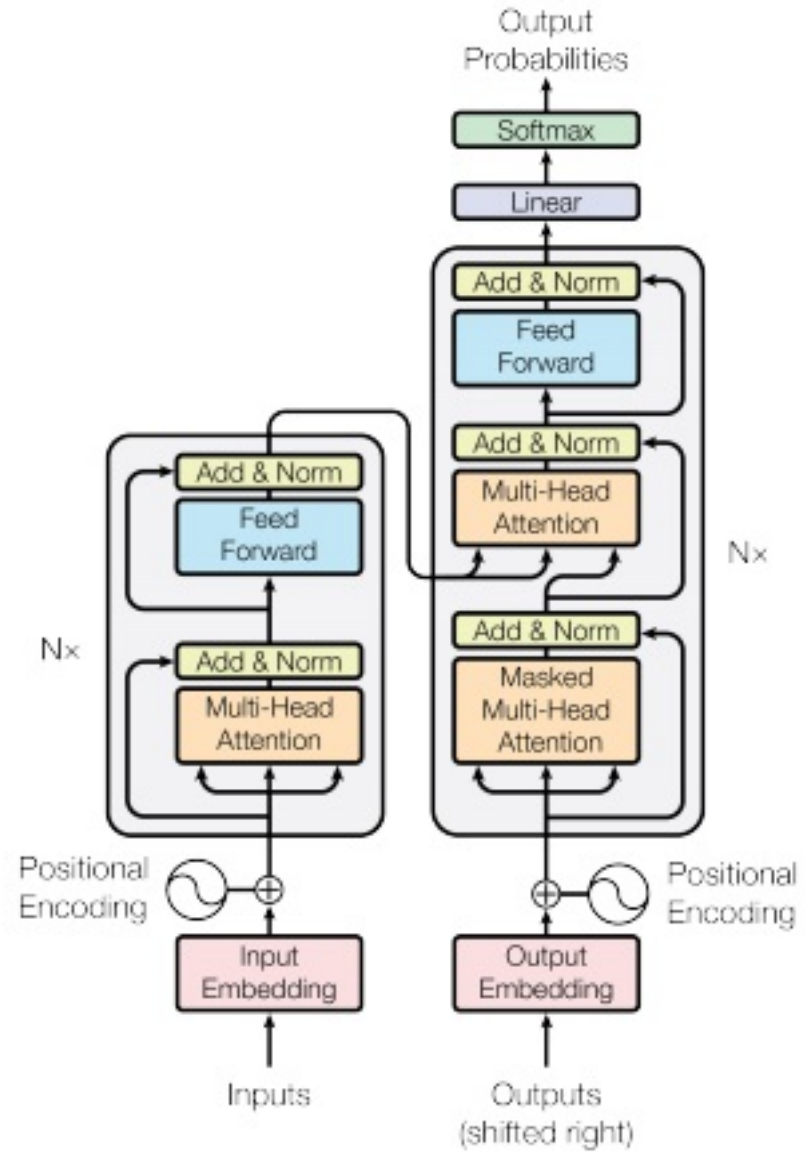

Рисунок 6 – Архитектура модели Tranfsormer

а второй - простая, полносвязная нейронная сеть. Мы используем остаточную связь вокруг каждого из двух подуровней с последующей нормализацией слоя. То есть выходной сигнал для каждого подуровня рассчитывается как $LayerNorm(x + Sublayer(x))$, где $Sublayer(x)$ является функцией, реализованной самим подуровнем. Чтобы упростить эти остаточные соединения, все подслои в модели, а также слои эмбендингов (преобразования слова, предложения, параграфа или целого текста в набор чисел – числовой вектор) преобразуют выходные данные в вектора размерности $dmodel = 512$.

В целом механизм работы энкодера можно описать следующим алго-

ритмом:

- Создаются эмбеддинги для всех слов предложения (вектора одинаковой размерности). В эмбеддинг также добавляется позиция слова в предложении;
- Берется вектор первого слова и вектор второго слова, подаются на однослойную сеть с одним выходом, которая выдает степень их похожести (скалярная величина). Эта скалярная величина умножается на вектор второго слова, получая его некоторую "ослабленную" на величину похожести копию;
- Вместо второго слова подается третье слово и делается тоже самое что в п.2. с той же самой сетью с теми же весами;
- Делая тоже самое для всех оставшихся слов предложения получаются их "ослабленные" (взвешенные) копии, которые выражают степень их похожести на первое слово. Далее эти все взвешенные вектора складываются друг с другом, получая один результирующий вектор размерности одного эмбединга;
- Так как оценка похожести слов всего одним способом (по одному критерию) считается недостаточной, тоже самое (п.2-4) повторяется несколько раз с другими весами. Один attention может определять похожесть слов по смысловой нагрузке, другой по грамматической, остальные еще как-то и т.п.;
- На выходе п.5. получается несколько векторов, каждый из которых является взвешенной суммой всех остальных слов предложения относительно их похожести на первое слово. Все эти вектора конкатинируются и получается один вектор;
- Дальше ставится еще один слой линейного преобразования, уменьшающий размерность результата п.6. до размерности вектора одного эмбединга. Получается некое представление первого слова предложения, составленное из взвешенных векторов всех остальных слов предложения;
- Такой же процесс производится для всех других слов в предложении;
- Так как размерность выхода та же, то можно проделать все тоже самое еще раз (п.2-8), но вместо оригинальных эмбеддингов слов взять то, что

получается после прохода через Multi-head attention слой, а нейросети внутри слоев attention взять с другими весами (веса между слоями не общие).

В **седьмом - десятом** разделах описывается практическая часть данной работы. Основной практической задачей данной работы было создание системы, которая смогла бы считывать логи с агрегаторов логов посредством использования API и создавать новые поисковые запросы по ключевым словам, и инфраструктуры для приложения, которое пишет эти логи в процессе своей работы.

Прежде всего арендуются виртуальные сервера одного из клауд провайдеров, в нашем случае AWS. Далее устанавливаются и настраиваются инструменты CI/CD (Jenkins), с помощью которых код приложения забирается из удаленного репозитория, собирается и выгружается на подготовленные сервера. Далее настраивается агрегатор логов, который собирает логи частей приложения со всего кластера и хранит их в себе. Затем, уже локально, запускается приложение для сбора логов с агрегатора и построения новых поисковых запросов.

Ознакомиться со всеми программными кодами можно на страницах дипломной работы в соответствующих приложениях.

ЗАКЛЮЧЕНИЕ

Разработка и использование нейронных сетей на сегодняшний день является одной из наиболее популярных и обширных отраслей индустрии информационных технологий. Количество и сложность задач, для решения которых создаются и применяются нейронные сети увеличивается с каждым годом. Очевидно, что в ближайшем будущем популярность данной технологии не снизится. В данной работе была изучена и решена одна из наиболее популярных задач, решаемых с помощью нейронных сетей, задача обработки естественного зыка. Для ее решения был рассмотрен специально созданный для подобного рода задач вид сетей - двунаправленная модель с transformer-архитектурой. Для применения этой сети был спроектирован и воссоздан облачный кластер . Для которого также было написано собственное программное обеспечение. Таким образом, цель магистерской работы достигнута.