Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский
Академический университет Российской академии наук»
Центр высшего образования

Кафедра математических и информационных технологий

Подгузов Никита Владимирович

Определение наилучшего ответа на StackOverflow

Бакалаврская работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Омельченко А. В.

Научный руководитель:
Курбанов Р. Э.

Рецензент:
Артамонов А. С.

Санкт-Петербург
2018

SAINT-PETERSBURG ACADEMIC UNIVERSITY
Higher education centre

Department of Mathematics and Information Technology

Nikita Podguzov

Best answer detection on StackOverflow

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Alexander Omelchenko

Scientific supervisor:
Rauf Kurbanov

Reviewer:
Alexey Artamonov

Saint-Petersburg
2018

Оглавление

Введение

С распространением и развитием интернета стали популярны системы вопросов и ответов — вид сайтов, позволяющий пользователям задавать вопросы и отвечать на вопросы, заданные другими пользователями. Среди наиболее популярных подобных систем можно выделить многопрофильные (такие как *Yahoo Answers*, *Quora*) и узкоспециализированные (например, *StackOverflow* [1]). В рамках данной работы мы сконцентрируемся на системе StackOverflow, которая в первую очередь специализируется на вопросах по программированию и смежным областям.

На сайтах системы StackOverflow у автора есть возможность после того, как он задал интересующий его вопрос, отметить ответ, решивший его проблему, как правильный.

Несмотря на это, достаточно большая часть вопросов остается без подобной отметки даже при наличии хороших ответов-кандидатов, что может усложнить поиск нужной информации по данному вопросу другим пользователям. Кроме того, пользователю, задавшему вопрос по какой-то не самой популярной теме, может быть сложно оценить правильность и релевантность полученных ответов.

Таким образом, перед нами появляется задача определения правильного ответа для вопросов со StackOverflow.

Мотивация

В связи с многократно возросшими вычислительными мощностями в последнее время в области машинного обучения приобрели популярность нейронные сети. Сфера применимости моделей, использующих нейронные сети, очень широка: от распознавания образов на изображениях [2] до предсказания курсов валют на бирже [3]. В частности, многие современные системы, связанные с обработкой естественного языка, например, платформы машинного перевода [4] или чатботы [5], используют нейронные сети.

Несмотря на наличие работ, исследующих задачу предсказания лучшего ответа для систем вопросов и ответов, в основном в них используются более классические методы машинного обучения, хотя нейронные сети и показывают достойные результаты в задачах классификации текстов.

В данной работе исследуется применимость нескольких моделей, основанных на нейронных сетях и использующих такие особенности сайта StackOverflow, как технический язык вопросов и наличие фрагментов кода в вопросах и ответах, к поставленной задаче.

Также хочется подчеркнуть, что данная работа ставит своей целью определение правильного ответа на основе текстовых признаков и без использования какой-либо внешней информации о рейтинге, то есть используя только содержание и дату пуб-

ликации вопросов и ответов. Это делает подобную систему более универсальной: во-первых, она абстрагируется от конкретной системы рейтингов и репутации, а во-вторых, позволяет отвечать на новые или непопулярные вопросы, в которых подобная информация может быть недоступна. Поэтому в рамках данной работы не рассматриваются рейтинги ответов или репутация пользователей-авторов, хотя это и могло бы повысить качество классификатора.

Кратко о последующих главах

В главе 1 производится обзор имеющихся решений и исследований по данной теме, их преимуществ и недостатков.

В главе 2 описывается формальная постановка задачи, рассматриваются ее особенности и ставятся основные цели и задачи работы.

В третьей главе описана работа по предварительной обработке и очистке данных с сайта StackOverflow, а также их анализ.

Глава 4 содержит описание и детали реализации основных методов и подходов к поставленной задаче.

В последней главе приведены результаты экспериментов с различными конфигурациями реализованных моделей, а также проведен анализ полученных результатов и сделаны выводы о применимости модели к задаче определения правильного ответа.

1. Обзор предметной области

1.1. Обзор существующих решений

Существует множество работ, исследующих различные аспекты систем вопросов и ответов: определение наилучшего ответа [6, 7, 8], определение качества ответа [9, 10], выбор ответов к новым вопросам из уже существующих [11, 12, 13], определение качества вопроса [9, 14], а также вероятности получения ответа на него [15].

Тем не менее, в рамках темы данной работы стоит обратить внимание на исследования, касающиеся первого обозначенного аспекта: определения наилучшего ответа. Все статьи, фокусирующиеся на этой теме, имеют общую структуру: они извлекают различные признаки из текстов ответов, а также метаинформацию, а затем используют некоторый бинарный классификатор, обученный на размеченных данных.

Были подробно рассмотрены особенности следующих работ:

- В одной из первых работ [7], анализирующих данную проблему в разрезе сайта StackOverflow, использовался классификатор «случайный лес» по признакам трех типов: соответствующих содержанию ответа, измеряющих похожесть вопроса и ответа, а также использующих метаинформацию. Отдельное внимание авторы уделили признакам, сравнивающим текущий ответ с остальными: например, то, насколько он похож на остальные ответы.

- Также достаточно важна работа [8], в которой, во-первых, проведено масштабное исследование 21 различного сайта системы StackExchange, а во-вторых, представлена так называемая идея дискретизации признаков: процесс получения нового признака путем сортировки значений одного из старых и замены его значений на порядковый номер в получившемся списке (эта идея будет раскрыта более подробно далее). Тем не менее, в рамках этой работы авторы никак не учитывают текст вопроса, хотя это, безусловно, важная составляющая при определении правильности ответа на вопрос.

- Наконец, нужно упомянуть исследование [6], проводившееся на наборах данных с трех сайтов системы StackExchange. Подход, представленный в работе, использует большое количество различных признаков, добиваясь наилучших результатов с помощью такой метаинформации, как рейтинги ответа или пользователя, отвечавшего на вопрос, которые не использовались в нашей работе. Тем не менее, авторы предоставляют полученные метрики экспериментов для различных подмножеств признаков, поэтому мы все равно можем сравнить полученные нами результаты с представленными в статье. В качестве классификатора авторы использовали чередующиеся решающие деревья [16]. Эта работа также не использует текстовые признаки вопроса.

Проанализировав имеющиеся исследования, можно отметить некоторые недостатки существующих решений:

- Несмотря на то, что во всех работах анализируется содержание текста ответа, в них никак не учитываются семантические значения слов, а также порядок слов в предложении и их контекст, авторы используют лишь некую глобальную информацию, извлекаемую из текста.
- Некоторые работы также не используют специфику сайта StackOverflow, вопросы на котором написаны техническим языком и с достаточно большим количеством терминов, а также могут содержать фрагменты исходного кода.

1.2. Векторное представление текстов

При работе с текстами в задачах обработки естественного языка требуется представлять слова и тексты в виде векторов, чтобы с ними было удобнее работать в дальнейшем. Существует несколько подходов к векторному представлению текстов.

1.2.1. Bag of Words

Bag of Words [17] — это одно из самых простых представлений, первые упоминания о котором датируются еще 1950-ми годами.

Основная идея состоит в том, что каждый документ представляется в виде вектора, размерность которого равна размеру словаря, а i-я компонента равна 1, если i-ое слово из словаря присутствует в документе, и 0 — иначе. При этом существуют различные модификации этого метода: могут не учитываться так называемые «стоп-слова», которые встречаются слишком часто и не несут в себе никакой информации о содержании документа; вместо бинарных значений компонент могут также использоваться количество вхождений слова в документ либо мера `TF-IDF` [18], когда значение i-й компоненты определяется по формуле:

$$w_i = tf_i \cdot \log \frac{N}{df_i} \tag{1}$$

где tf_i — количество вхождений слова в документ,

N — количество документов в корпусе данных,

df_i — количество документов, в которых встречается данное слово.

Подобная модификация позволяет учитывать, насколько то или иное слово часто встречается в других документах: давать больший вес более редким и информативным словам и игнорировать шумовые слова.

Пример применения базового представления Bag of Words со стоп-словами можно видеть на рисунке 1.

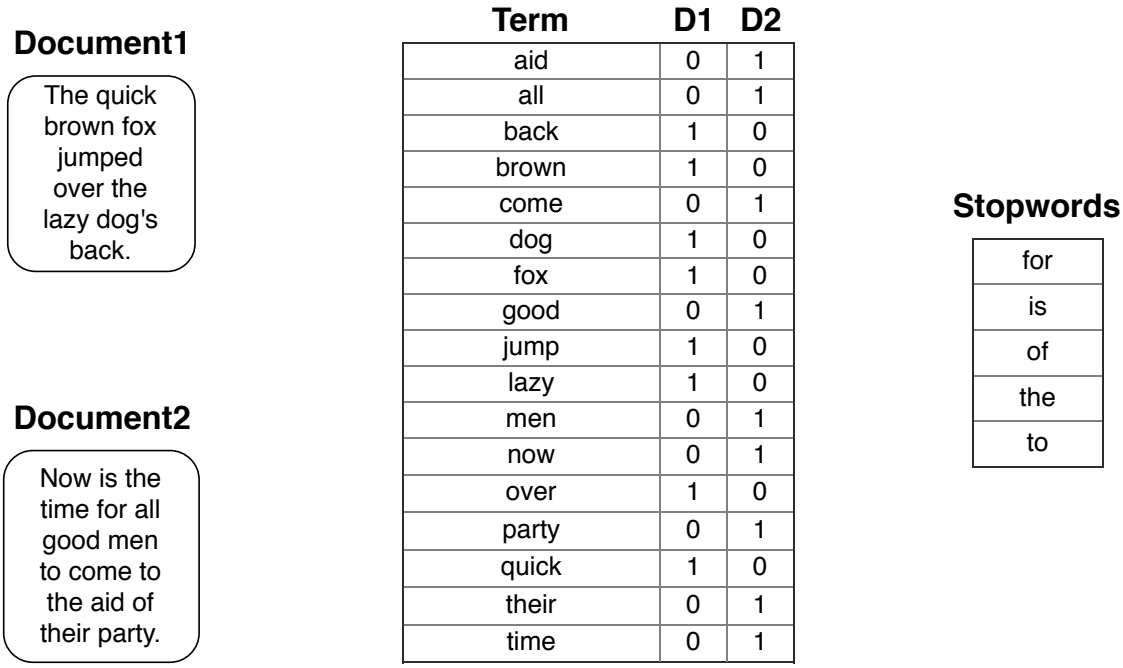


Рис. 1: Пример работы представления Bag of Words

1.2.2. Word2Vec

Другой подход состоит в последовательной векторизации слов текста, а затем работе с этой последовательностью векторов.

В 2013 году был представлен Word2Vec [19] — способ векторизации слов, основанный на нейронных сетях. Основная идея метода состоит в предположении о том, что слова, находящиеся в похожих контекстах, скорее всего будут значить похожие вещи, то есть будут семантически близкими. В данном случае под контекстом подразумевается набор слов из окна фиксированный ширины вокруг текущего слова. Тем не менее, этот контекст можно использовать для обучения двумя разными способами, поэтому в Word2Vec используется один из двух типов алгоритма:

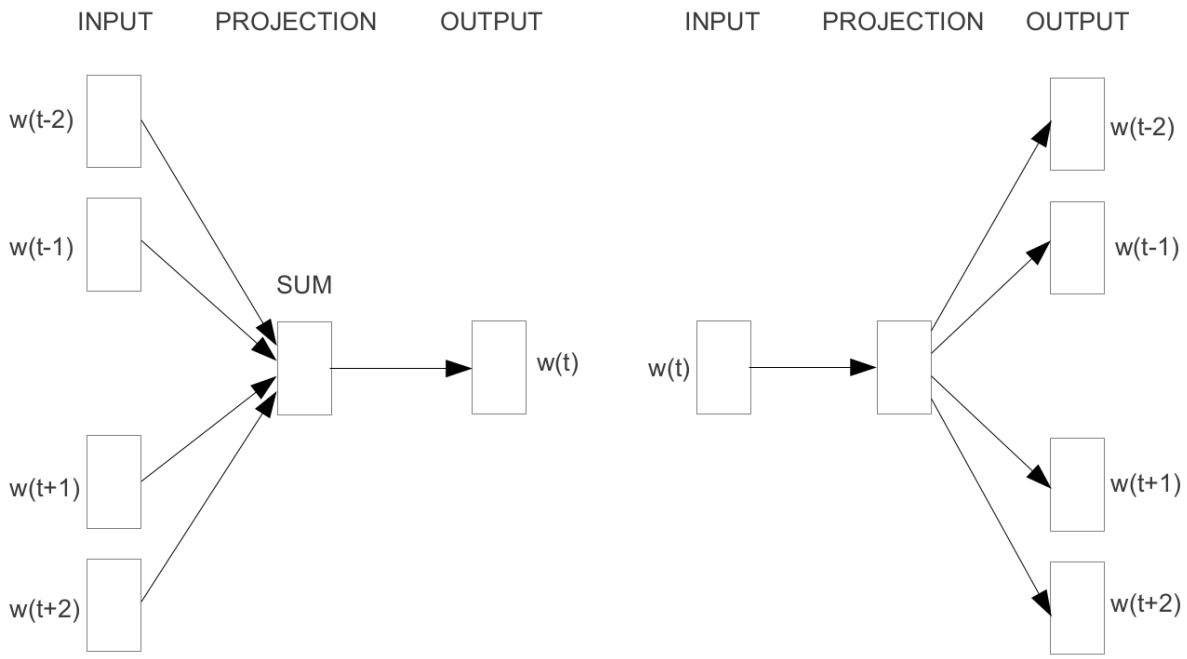
- **Модель CBOW (Continuous Bag of Words)** (рисунок 2a).

  В данной модели используется нейронная сеть с одним скрытым полносвязным слоем с функцией активации `ReLU` [20]. Обучение происходит следующим образом: корпус имеющихся текстов обрабатывается скользящим окном, выделяя контексты фиксированной ширины, на вход сети подается контекст, представленный в виде вектора размерности длины словаря, полученный по модели Bag of Words, описанной ранее, а на выходе ожидается вектор, в котором i-я компонента обозначает вероятность того, что в центре окна i-е слово. Для повышения производительности алгоритма используется негативное семплирование, позволяющее вместо расчета всех возможных контекстов для функции потерь использовать фиксированное количество случайных контекстов. В качестве итогового векторного представления слов используется матрица весов скрытого слоя.

Таким образом, мы пытаемся по данному контексту научиться предсказывать слово.

- **Модель Skip-gram** (рисунок 2b).

  Альтернативой первой модели является модель Skip-gram. Она имеет схожую архитектуру, но при этом основывается на обратной идее: предсказании контекста по слову, то есть получая на вход слово как единичный вектор, она пытается предсказать вероятности вхождения каждого из слов в контекст текущего.



(a) Архитектура модели CBOW (b) Архитектура модели Skip-gram

Рис. 2: Модели Word2Vec

Стоит отметить, что для получения векторных представлений хорошего качества требуется обучение Word2Vec на большом корпусе неразмеченных текстов (больше миллиарда слов). Интересной особенностью полученных векторизаций является не только близкое расположение векторов, отвечающих за близкие по смыслу слова, но и сохранение семантических отношений между словами: например, разность векторов, соответствующих словам «король» и «королева» очень близка к разности векторов «мужчина» и «женщина».

1.2.3. Fasttext

У предыдущей модели есть один достаточно важный недостаток: она не позволяет получать векторные представления слов, которые не встречались в обучающей выборке.

Для решения этой проблемы используется модификация Word2Vec — Fasttext [21]. Если раньше в качестве представления слова мы использовали единичный вектор, то в модели Fasttext из слова также выделяются все n-граммы — непрерывные подстроки длины n (в базовой версии для всех n от 3 до 6), и в качестве представления слова берется взвешенная сумма векторов, соответствующих n-граммам.

Подобный трюк позволяет получать векторизации для слова не из словаря, используя векторизации его n-грамм.

1.3. Нейронные сети

За последние годы в области обработки естественных языков, в частности для задачи классификации текстов, нейронные сети стали показывать блестящие результаты. Ключевым эффектом является векторное представление целого текста, которое отражает его семантику, то есть смысл, заложенный в нем.

Для этого используются две принципиально разные архитектуры, описания которых представлены ниже.

1.3.1. Рекуррентная нейронная сеть

Базовые идеи архитектуры рекурретных нейронных сетей (RNN — recurrent neural network), были разработаны еще в 1980-е годы. В основе этой архитектуры лежит использование рекуррентных слоев (рисунок 3), позволяющих работать с данными, которые являются некоторой последовательностью. Для этого в качестве входных данных на каждую следующую ячейку RNN поступает не только новый элемент последовательности, но и некоторая информация из состояния предыдущей ячейки.

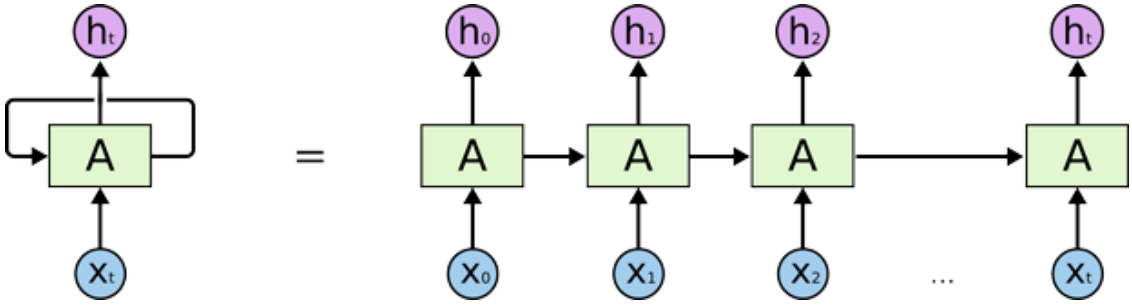


Рис. 3: Рекуррентная нейронная сеть

Данный механизм может помочь и в решении задачи классификации текстов — последовательности слов, поскольку позволяет учитывать связь различных слов и предложений друг с другом. Для векторизации слов может быть использован один из методов из подраздела 1.2.

Тем не менее, в текстах часто важен достаточно широкий контекст, а обычные RNN на практике оказываются неспособны захватывать подобные связи из-за проблемы затухающего градиента: при увеличении длины контекста норма градиента может экспоненциально убывать, из-за чего влияние текущего входа не может распространяться слишком далеко. Чтобы избежать подобной проблемы, были придуманы LSTM-ячейки [22] (рисунок 4), которые способны сохранять информацию в течение длительного времени.

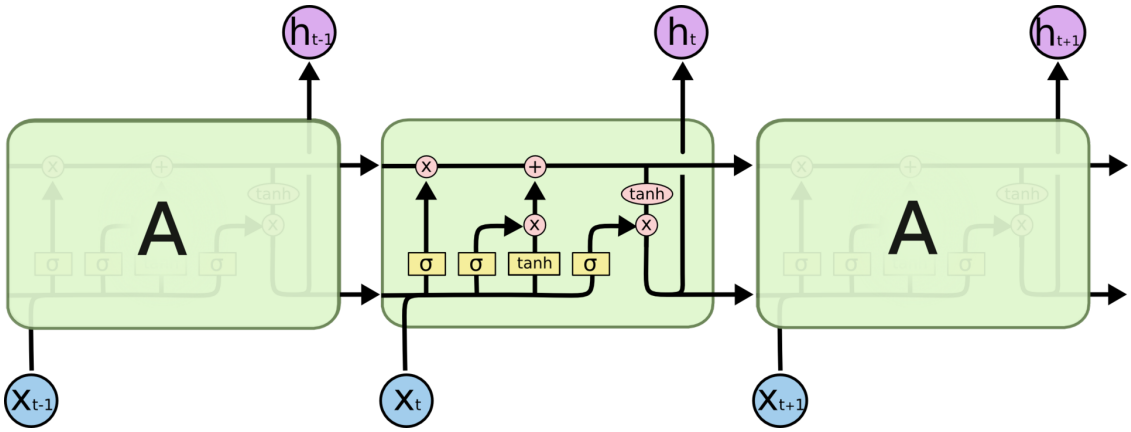


Рис. 4: LSTM-ячейка

Подобная архитектура позволяет захватывать контекст слева, хотя при анализе текста важен контекст, окружающий слово с обеих сторон. Для этого придумана модификация обычного рекуррентного слоя — двунаправленный рекуррентный слой [23], который представляет из себя два слоя, в один из которых данные подаются на вход в прямом порядке, а в другой — в обратном, а в качестве выхода используется конкатенация выходов с этих двух слоев.

1.3.2. Сверточная нейронная сеть

Основой для сверточных нейронных сетей [2] (CNN — convolutional neural network) служат сверточные слои, которые работают следующим образом: на вход этому слою подается тензор, после чего каждый фрагмент этого тензора скалярно умножается на ядро свертки, и получившийся тензор передается следующему слою сети.

Несмотря на то, что изначально этот механизм применялся для работы с изображениями, исследования последних лет [24] показали, что он также эффективен и при работе с текстом.

На рисунке 5 показан пример архитектуры, используемой для классификации текстов. В качестве входной матрицы подаются векторизованные представления слов текста, после чего к ним применяется свертка размером $k \cdot D$, где D — размерность векторного представления, а k — параметр сети. Обычно используется несколько сверток с различными ядрами, при этом используются k, равные $2, 3, 5, 7$, что позволяет

захватить контекст вокруг слова. После сверточного слоя используется глобальный `Max-Pooling` слой, который оставляет максимальное значение из каждой свертки, тем самым позволяя уменьшить размерность данных. Получившиеся значения конкатенируются и отдаются на вход следующему слою. В данной архитектуре используется обычный полносвязный слой, с помощью которого определяется принадлежность текста к одному из двух классов.

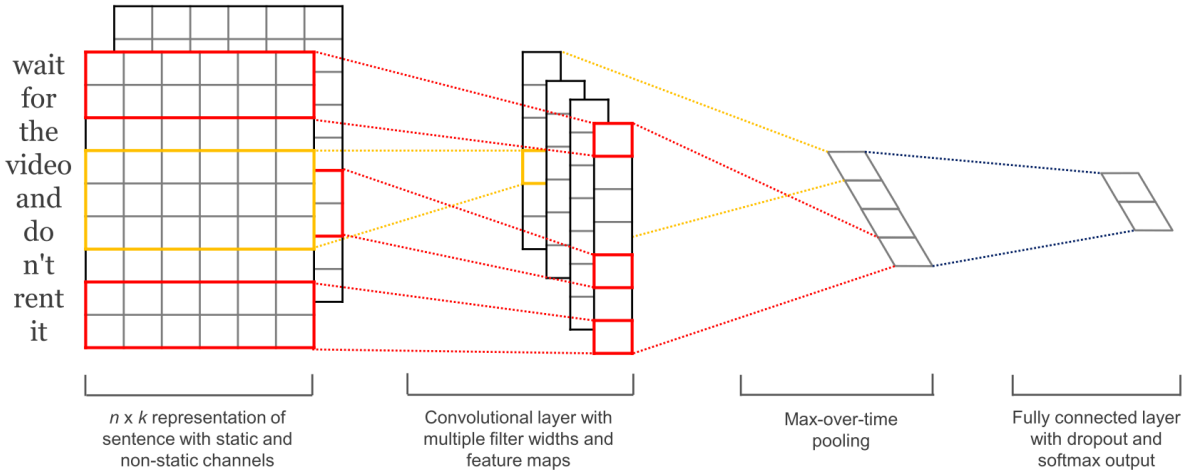


Рис. 5: Сверточная нейронная сеть для классификации текстов

2. Цель и задачи

Рассмотрев особенности уже существующих решений, постараемся избавиться от их недостатков, использовав для этого передовой инструмент последних лет, применяющийся для классификации текстов, — нейронные сети.

Таким образом, целью данной работы является разработка системы, позволяющей определять правильность ответа на StackOverflow, используя глубокое обучение для анализа содержания вопросов и ответов.

Для достижения описанной выше цели нам потребуется решить следующие задачи:

- Собрать и проанализировать данные, которые будут использоваться в дальнейшем для обучения и тестирования моделей.
- Разработать несколько моделей бинарных классификаторов на основе нейронных сетей, предсказывающих правильность ответа по содержанию вопроса и ответа.
- Произвести обучение и тестирование различных конфигураций моделей.
- Сравнить полученные результаты с базовыми алгоритмами и результатами других исследований.

3. Описание данных

Для задач, решаемых методами машинного обучения с учителем, к числу которых принадлежат и нейронные сети, важно наличие данных, на которых можно обучать, а затем и тестировать алгоритм.

3.1. Сбор данных

Так как большая часть имеющихся работ проводила тестирование на данных одного из сайтов системы StackExchange — Server Fault [25], который специализируется на теме администрирования веб-серверов, то для удобства сравнения результатов мы также решили использовать эти данные. Кроме того, этот набор данных удобен тем, что содержит достаточное количество данных для тренировки, при этом с ним удобно работать во время предобработки данных, в отличие от базы данных вопросов сайта StackOverflow, размер которой превосходит 50 гигабайт.

Для сбора данных использовалась база вопросов Server Fault, размещенная в открытом доступе [26] и содержащая данные с апреля 2009 года по март 2018 года. База представляет из себя `XML`-файл размером ≈ 0.9 гигабайт, каждая строка которого соответствует либо вопросу, либо ответу. В таблице 1 кратко описаны атрибуты имеющегося набора данных.

Атрибут	**Описание**
Id	Идентификатор поста
PostTypeId	Тип поста
AcceptedAnswerId	Идентификатор выбранного автором ответа (если есть)
ParentId	Идентификатор вопроса, на который был дан текущий ответ
CreationDate	Дата создания поста
Score	Рейтинг поста
ViewCount	Количество просмотров поста
Body	Текст поста в виде `HTML`-текста
OwnerUserId	Идентификатор автора поста
Title	Название поста
Tags	Теги вопроса
AnswerCount	Количество ответов на вопрос
CommentCount	Количество комментариев к посту

Таблица 1: Описание полей `XML`-файла с данными

3.2. Анализ данных

После анализа было получено, что в имеющемся наборе данных содержится:

- 684 тысячи постов, из них 257 тысяч вопросов и 427 тысяч ответов.

- 130 тысяч вопросов (51%) без выбранного автором вопроса ответа.
- 28 тысяч вопросов (11%), у которых нет ни одного ответа.

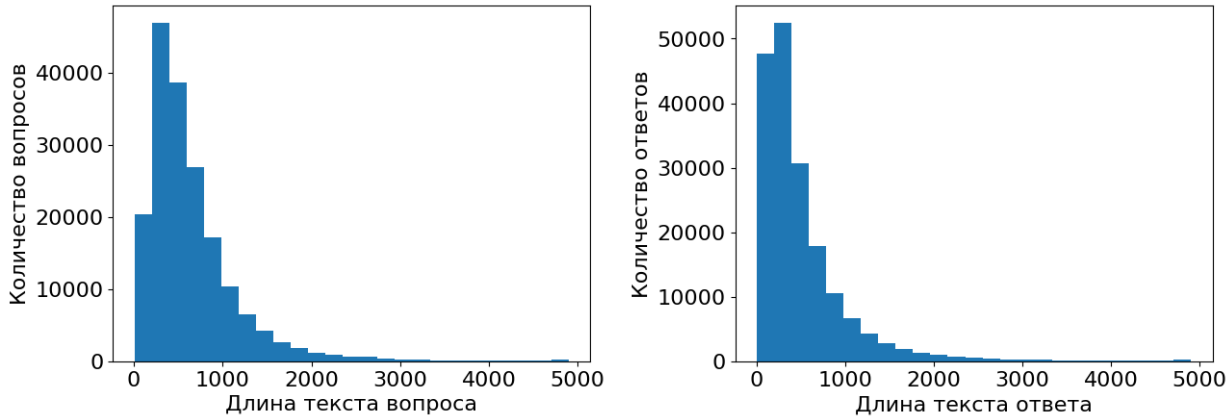
Кроме того, не учитывались вопросы, у которых нет ни одного ответа (так как они не давали никаких новых данных в рамках поставленной задачи), а также вопросы с отрицательным рейтингом, так как вероятнее всего это шумовые данные, в которых заданный вопрос либо плохо сформулирован, либо не соответствует тематике сайта.

После подобной фильтрации осталось 80 тысяч вопросов и 180 тысяч ответов.

Также было проанализировано содержание текстов вопросов и ответов, в ходе которого выяснилось, что тексты содержат большое количество терминов и слов с опечатками.

На рисунке 6 можно увидеть распределения длин текстов вопросов и ответов, которые понадобятся нам в дальнейшем для построения моделей классификаторов.

Медианным значением длины текста является 500 символов для вопросов и 350 символов для ответов.



(a) Длины вопросов

(b) Длины ответов

Рис. 6: Распределения длин текстов

3.3. Предобработка данных

Описанный выше набор данных нужно привести к форме, с которой будет удобно работать в дальнейшем, для этого был написан скрипт на языке Python, который сохраняет данные в формате `CSV`-файла, каждая запись в котором содержит:

- Признаки, относящиеся к ответу: идентификатор и текст ответа, был ли ответ выбран автором вопроса, как правильный.
- Признаки, относящиеся к вопросу: идентификатор, название и текст вопроса, на который был дан текущий ответ.

- Признаки, являющиеся метаинформацией: количество ответов на вопрос, возраст ответа, то есть разница в миллисекундах между датами создания ответа и вопроса.

Так как в исходном `XML`-файле текст хранился в виде `HTML`-кода, был написан обработчик дерева DOM [27], который:

- Очищает текст от тегов.
- Игнорирует содержимое тегов `del`, `s` и `strike`.
- Заменяет содержимое тега `code` на специальное слово `CODE_LEXEM`.

Все полученные подобным образом тексты были очищены от любых небуквенных символов, приведены к нижнему регистру, разбиты на отдельные слова и сохранены в отдельном `CSV`-файле.

3.4. Разделение данных для эксперимента

Для решения задачи машинного обучения итоговая выборка ответов была подразделена на 3: обучающую, проверочную и тестовую.

Сначала из всех ответов была выделена тестовая подвыборка размером 20% от всей выборки, а затем с таким же соотношением оставшаяся выборка была разделена на тренировочную и проверочную. Размер подвыборки 20% был выбран, как один из наиболее распространенных для задач машинного обучения.

Итого, имеющийся набор данных был разбит на три выборки:

- Обучающая выборка: 51000 ответов.
- Проверочная выборка: 13000 ответов.
- Тестовая выборка: 16000 ответов.

4. Методы и реализация

4.1. Извлечение признаков

Напомним, что после предобработки мы получили данные, в которых присутствуют тексты ответа и вопроса, к которому относится текущий ответ, а также некоторая метаинформация.

Нейронные сети работают с числовыми признаками, поэтому в следующих подразделах будет, во-первых, описан использованный подход к векторизации слов, а во-вторых, перечислены дополнительные признаки, которые могут помочь в поставленной перед нами задаче классификации.

4.1.1. Векторное представление вопроса и ответа

Как уже говорилось в подразделе 1.2, существует несколько способов векторизации слов, но нас будет интересовать подход Word2Vec, который помимо прочего описывает семантические значения слов.

Для обучения этих представлений требуется большой корпус неразмеченных данных, в данной работе использовалось обучение на базе данных [28], состоящей из статей Wikipedia до октября 2017 года. В этом корпусе присутствует ≈ 4.9 миллиона статей, в которых содержатся ≈ 132 миллиона предложений и ≈ 2.5 миллиарда слов. При обучении игнорировались слишком короткие предложения (длиной менее 5), а также использовалась модель CBOW.

В связи со спецификой Server Fault в текстах вопросов и ответов встречается множество технических слов и терминов, кроме того, как и на любом форуме, часто пользователи пишут слова с ошибками или опечатками. Это приводит к тому, что модель Word2Vec, обученная по корпусу Wikipedia, не содержит множества слов, присутствующих в текстах вопросов и ответов, в том числе терминов, которые являются важной информацией. В среднем, модель не содержала векторных представлений ≈ 10 слов на каждую пару вопроса и ответа.

Чтобы это исправить, было предпринято два шага:

- Так как часто новые слова являются конкатенацией слов из словаря, например, *texthtml* или *powercenter*, а слова с опечатками отличаются от слов из словаря всего одним символом, то есть множество n-грамм у них сильно пересекается с множеством n-грамм слов из словаря, то вместо использования Word2Vec мы решили использовать модификацию, которая работает с n-граммами — Fasttext.

- Также у нас есть специализированные тексты, которые можно использовать для обучения нашей модели, поэтому модель была дотренирована на текстах вопросов и ответов из обучающей подвыборки.

Как результат, обученная подобным образом модель Fasttext содержала векторизации всех слов из данных и позволила избежать проблемы неизвестных слов.

Английский — аналитический язык, что значит, что в нем грамматические отношения передаются через синтаксис, то есть через отдельные служебные слова. Например, в отличие от синтетического русского языка в нем нет склонений.

А исходя из того, что корпус текстов Wikipedia большой, то отпадает потребность производить стемминг, который наоборот может лишь ухудшить качество векторизации, склеивая несколько различных слов в одно.

4.1.2. Лингвистические признаки

Кроме содержания, часто важно и грамотное оформление ответа: логичное разбиение на предложения и параграфы, проставленные ссылки на источники и удобочитаемость. Поэтому в качестве дополнительных лингвистических были извлечены следующие признаки, описывающие либо структуру `HTML`-кода, либо текста ответа:

- Количество ссылок в тексте, то есть количество тегов `<a>`.
- Количество фрагментов кода в тексте, то есть количество тегов `<code>`.
- Количество параграфов в тексте, то есть количество тегов `<p>`.
- Доля заглавных букв в тексте, то есть отношение количества заглавных букв к длине текста.
- Доля строчных букв в тексте, то есть отношение количества строчных букв к длине текста.
- Доля пробелов в тексте, то есть отношение количества пробельных символов к длине текста.
- Длина текста.
- Длина самого длинного предложения.
- Количество слов в самом длинном предложении.
- Среднее количество слов в предложении.
- Средняя длина слова в тексте.
- Количество предложений.
- Автоматический индекс удобочитаемости [29]:

$$ARI = 4.71 \cdot \frac{characters}{words} + 0.5 \cdot \frac{words}{sentences} - 21.43 \tag{2}$$

- Индекс удобочитаемости Флеша [30]:

$$FRE = 206.835 - 1.015 \cdot \frac{words}{sentences} - 84.6 \cdot \frac{syllables}{words} \tag{3}$$

- Индекс `SMOG` [31]:

$$SMOG = 1.043 \cdot \sqrt{\frac{30 \cdot polysyllables}{sentences}} + 3.1291 \tag{4}$$

- Индекс Флеша-Кинкейда [32]:

$$FK = 0.39 \cdot \frac{words}{sentences} + 11.8 \cdot \frac{syllables}{words} - 15.59 \tag{5}$$

- Индекс Колман-Лиау [33]:

$$CLI = 5.88 \cdot \frac{characters}{words} - 29.6 \cdot \frac{sentences}{words} - 15.8 \tag{6}$$

- Фог-индекс [34]:

$$FOG = 0.4 \cdot \left(\frac{words}{sentences} + 100 \cdot \frac{polysyllables}{words} \right) \tag{7}$$

- Индекс `LIX` [35]:

$$LIX = \frac{words}{sentences} + 100 \cdot \frac{longwords}{words} \tag{8}$$

В упомянутых выше формулах использованы следующие обозначения:

- $characters$ — количество символов в тексте.
- $words$ — количество слов в тексте.
- $sentences$ — количество предложений в тексте.
- $syllables$ — количество слогов в тексте.
- $polysyllables$ — количество слов с хотя бы тремя слогами.
- $longwords$ — количество слов длины хотя бы 7.

На рисунке 7 можно увидеть распределение значений индекса удобочитаемости `LIX` правильных и неправильных ответов, из которого видно, что подобные индексы действительно влияют на решение пользователя о выборе правильного ответа.

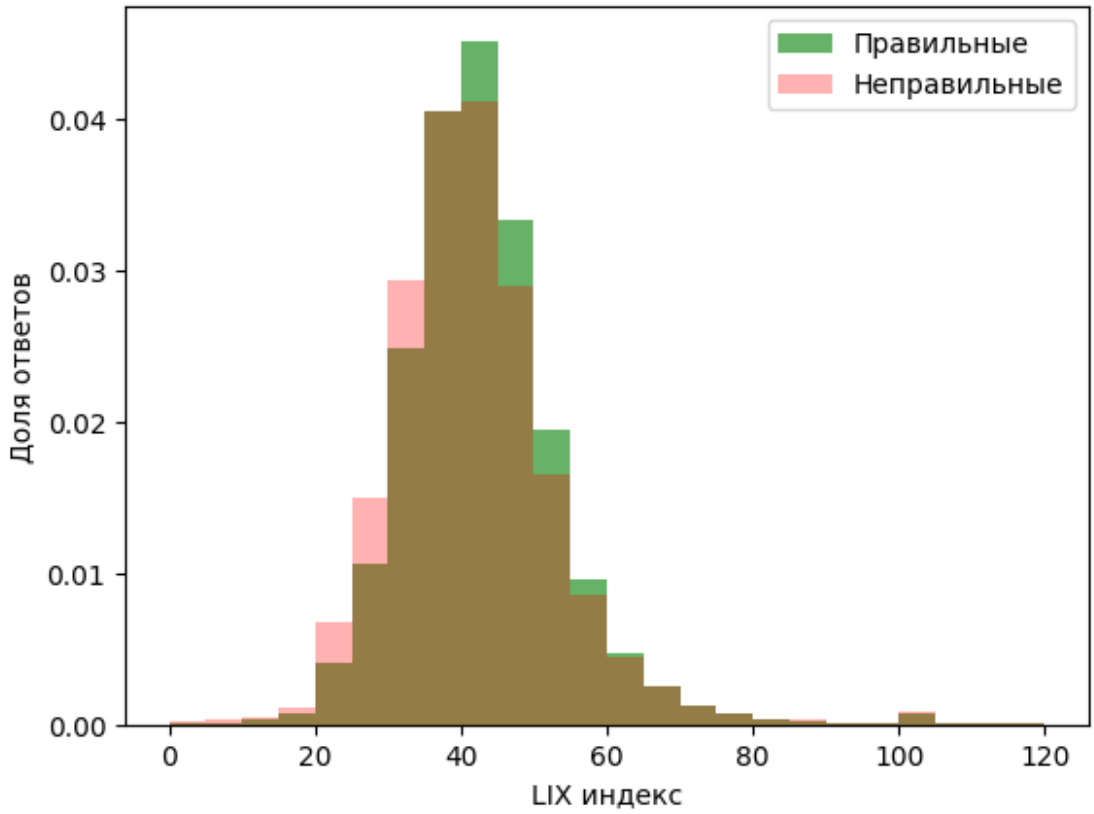


Рис. 7: Распределение значений индекса `LIX`

4.1.3. Словарные признаки

Как было замечено ранее, в текстах встречается множество терминов, поэтому для оценки того, насколько ответ соответствует заданному вопросу, было также добавлено 4 дополнительных словарных признака:

- Коэффициент Жаккара [36] для текстов вопроса и ответа:

$$J = \frac{|V_Q \cap V_A|}{|V_Q \cup V_A|} \tag{9}$$

  где V_Q — набор слов вопроса, а V_A — набор слов ответа.

- Коэффициент Жаккара для текстов вопроса и ответа без учета стоп-слов.
- Модификация первого словарного признака с учетом `IDF`, то есть теперь вес слова не единичный, а обратно пропорционален частоте употребления этого слова.
- Аналогичная модификация второго словарного признака.

4.1.4. Метаинформация

В качестве метаинформации используется всего 2 признака:

- Количество ответов на заданный вопрос.
- Возраст ответа, то есть количество времени в миллисекундах, которое прошло от момента создания вопроса до публикации текущего ответа.

4.1.5. Нормализация

Перед использованием извлеченных признаков необходимо произвести их нормализацию (то есть приведению их значений к отрезку $[0, 1]$). Были посчитаны три вида нормализации:

- **Глобальная нормализация**.

  Классическая нормализация по каждому признаку x: $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$, где x_{min} — минимальное значение признака, а x_{max} — максимальное.

- **Нормализация по вопросу**.

  Так как признаки ответов могут сильно отличаться (например, длина ответа может быть от нескольких десятков до нескольких тысяч символов), а ответы для конкретного вопроса выбираются из небольшого подмножества, то была добавлена нормализация по вопросу: процесс, когда все ответы группируются по вопросу, к которому они относятся, а затем проводится нормализация внутри каждой подгруппы.

- **Дискретизация**.

  Кроме того, могут быть важны не конкретные значения признаков, а лишь их порядок относительно признаков других ответов на текущий вопрос. Для учета этого, были добавлены дискретизированные признаки: если обозначить множество ответов в одной подгруппе за A, то все признаки ответов внутри A сортируются и переводятся в множество $\left\{\frac{1}{|A|}, \frac{2}{|A|}, \ldots, \frac{|A|-1}{|A|}, 1\right\}$ с учетом порядка, то есть минимальное значение признака переходит в $\frac{1}{|A|}$, а максимальное — в 1.

  Например, если применить дискретизацию к признаку "возраст ответа", то мы получим позицию ответа, что является важным признаком при классификации.

4.2. Методы

Для векторизаций текстов вопросов и ответов использовались архитектуры, представленные в подразделе 1.3: либо сверточные слои с ядрами различных размеров для учета контекстов различной длины, либо двунаправленный рекуррентный слой. Стоит отметить, что увеличение количества слоев (например, добавление еще одного

рекуррентного слоя) сильно замедляло время обучения, при этом не улучшая метрики на проверочной выборке, поэтому в итоговой модели использовались именно такие конфигурации слоев.

Было рассмотрено несколько различных архитектур, которые по-разному используют признаки первого типа — векторные представления текста вопроса и ответа.

- Первый подход не использует эти признаки вообще: сеть представляет из себя двуслойную нейронную сеть со скрытым полносвязным слоем и линейной функцией активацией, на вход которой подаются лингвистические, словарные и метаинформационные признаки, а на выходе с помощью сигмоидной функции активации получается вероятность принадлежности к классу правильных ответов.

- Тем не менее, мы хотим учитывать в том числе и тексты, поэтому в следующем подходе для учитывания текста ответа использовались сверточные либо рекурретные слои, после чего полученный вектор, сконкатенированный вместе с остальными признаками использовался в описанной в первом подходе нейронной сети.

- Однако, зачастую просто векторизация текста ответа сама по себе — ни о чем не говорящий признак.

  При добавлении векторного представления вопроса, возникает два важных момента, которые хочется рассмотреть подробнее:

  – Должны ли параметры рекуррентных или сверточных слоев обучаться совместно для текстов вопросов и ответов или это должны быть одни параметры для вопросов и другие для ответов?

    Для ответа на этот вопрос были опробованы оба способа, анализ результатов которых можно найти в подразделе 5.4.

  – Как совместно учитывать семантические представления вопроса и ответа?

    На выходе из предыдущих слоев мы получаем два вектора Q и A, соответствующие вопросу и ответу соответственно. Были рассмотрены несколько функций преобразования этих векторов в один вектор-признак:

    * Конкатенация:

    $$Out = Concat(Q, A) \tag{10}$$

    * Сумма:

    $$Out = Q + A \tag{11}$$

    * Косинусный коэффициент:

    $$Out = \frac{QA^T}{||Q||||A||} \tag{12}$$

* GESD [37]:

$$Out = \frac{1}{1 + ||Q - A||} \cdot \frac{1}{1 + \exp(-1 - QA^T)} \tag{13}$$

* AESD [37]:

$$Out = \frac{0.5}{1 + ||Q - A||} + \frac{0.5}{1 + \exp(-1 - QA^T)} \tag{14}$$

Для того, чтобы избежать эффекта переобучения, после получения векторных представлений текстов использовался Dropout-слой.

4.3. Реализация модели

В этом подразделе будут продемонстрированы две финальные архитектуры, которые были использованы для сравнения с результатами других работ.

В обоих случаях в качестве функции потерь использовалась бинарная кросс-энтропия, а в качестве оптимизатора использовался Adam [38]. Для определения похожести вопроса и ответа использовалась функция $AESD$, введенная в предыдущем подразделе. В качестве длин текстов, подаваемых на вход, были взяты медианные значения, посчитанные в подразделе 3.2.

4.3.1. Сверточная нейронная сеть

На рисунке 8 указана архитектура, основанная на сверточной сети. В качестве оптимальных параметров были использованы 128 фильтров: по 32 с размерами 2, 3, 5 и 7, размер скрытого слоя равен 128, а коэффициент Dropout равен 0.5.

4.3.2. Рекуррентная нейронная сеть

На рисунке 9 указана архитектура, основанная на рекуррентной сети. В качестве оптимальных параметров были использованы двунаправленная рекуррентная сеть с выходной размерностью, равной 128, размер скрытого слоя равен 256, а коэффициент Dropout равен 0.5.

Стоит отметить, что эта модель требует в разы больше времени на обучение, что объясняется наличием двунаправленного рекуррентного слоя.

4.3.3. Детали реализации

Все вышеобозначенные модели и их модификации были реализованы с помощью фреймворка Keras [39].

Для обучения и работы с моделями Word2Vec и Fasttext использовался фреймворк Gensim [40].

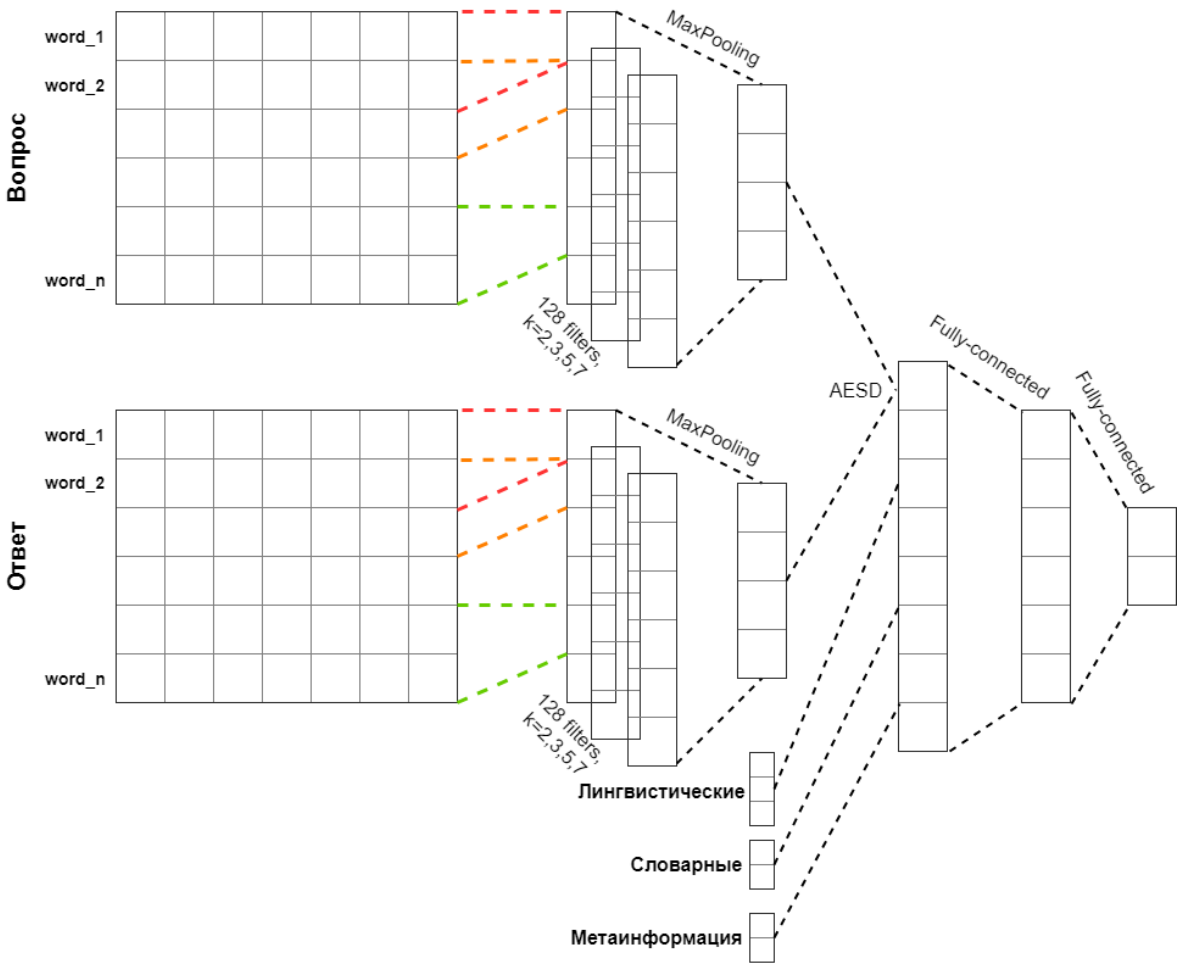


Рис. 8: Архитектура, основанная на сверточных слоях

Для предобработки текста использовалась библиотека NLTK [41], а для извлечения различных лингвистических признаков была использована модификация библиотеки textstat [42].

4.3.4. Репозиторий

Исходный код доступен по ссылке [43].

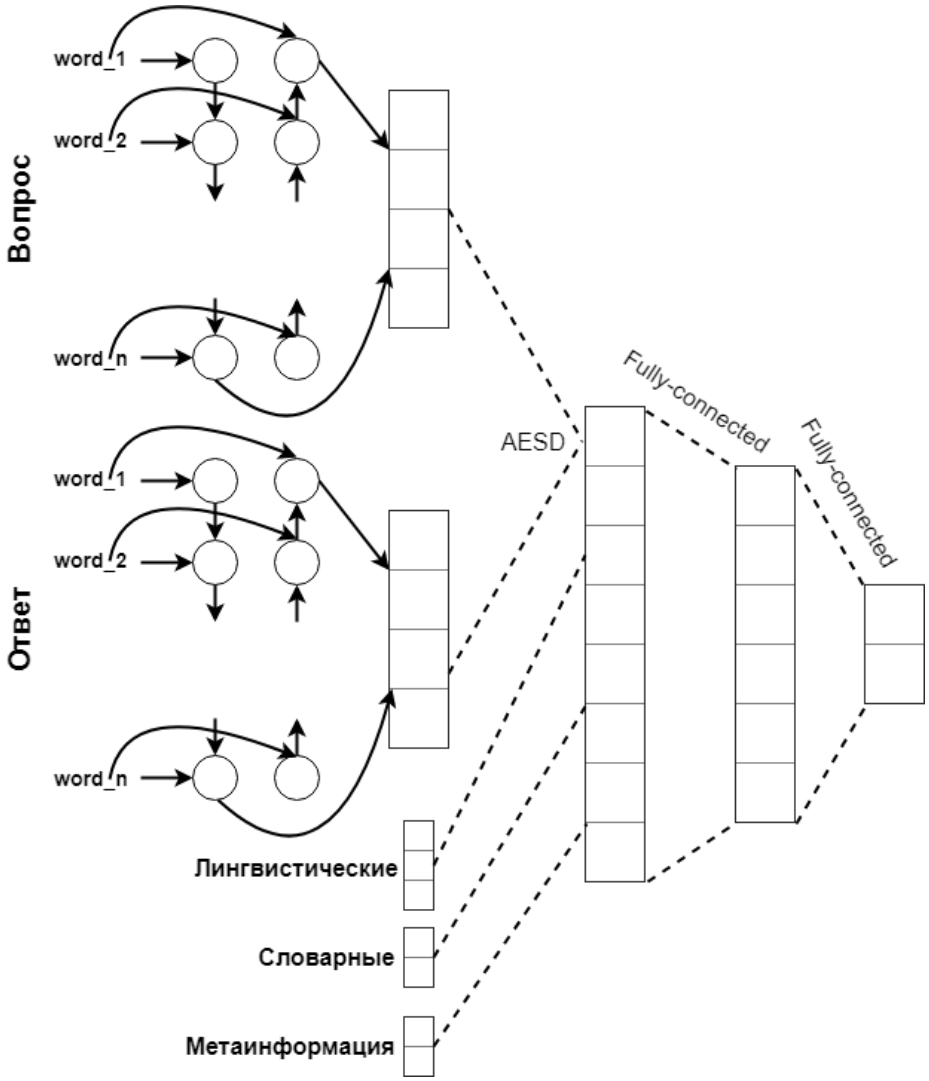


Рис. 9: Архитектура, основанная на рекуррентных слоях

5. Тестирование

Для оценки качества различных модификаций было произведено тестирование обученных нейронных сетей на тестовой выборке.

5.1. Используемые метрики

Для сравнения моделей использовалось несколько метрик, классических для задачи бинарной классификации:

- Аккуратность:
$$Acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{15}$$
- Точность:
$$P = \frac{TP}{TP + FP} \tag{16}$$
- Полнота:
$$R = \frac{TP}{TP + FN} \tag{17}$$
- F-мера, позволяющая объединить точность и полноту в единую метрику:
$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \tag{18}$$
- Площадь под ROC-кривой (AUC) — графиком, показывающим зависимость количества верно классифицированных положительных примеров от количества неверно классифицированных отрицательных примеров при изменении порога решающего правила.

В упомянутых выше формулах использованы следующие обозначения:

- TP — количество верно классифицированных правильных ответов.
- FP — количество неверно классифицированных правильных ответов.
- FN — количество неверно классифицированных неправильных ответов.
- TN — количество верно классифицированных неправильных ответов.

5.2. Производительность

Обучение и тестирование производилось на GPU Tesla V100 с 16 гигабайтами памяти. Обучение одной модели занимало несколько часов (от 4 в случае сверточной сети до 15 в случае рекуррентной).

5.3. Результаты

Было произведено тестирование и сравнение моделей, описанных в разделе 4.2, с показателями исследований, упоминавшихся в подразделе 1.1, а также с двумя алгоритмами:

- Наивный байесовский классификатор, на вход которому подается векторное представление ответа по модели Bag of Words с мерой `TF-IDF`. Для тестирования использовалась реализация из фреймворка Scikit-Learn [44].
- Классификатор, который считает правильным ответ, который был дан раньше остальных.

Модель	**Виды признаков**	**Acc**	**P**	**R**	**F_1**	**AUC**
First-answer	метаинформация	0.66	0.69	0.60	0.64	0.66
Naive-Bayes with TF-IDF	словарные	0.6	0.58	0.54	0.56	0.59
Burel et al. [6]	лингвистические, словарные, пользовательские, метаинформация	-	0.77	0.77	0.76	0.83
Tian et al. [7]	лингвистические, метаинформация	0.72	-	-	-	-
Gkotsis et al. [8]	лингвистические, словарные, метаинформация	-	0.83	0.66	0.74	0.85
CNN	текстовые, лингвистические, словарные, метаинформация	**0.77**	0.84	0.61	0.71	**0.86**
RNN	текстовые, лингвистические, словарные, метаинформация	**0.79**	0.87	0.62	0.73	**0.88**

Таблица 2: Сравнения качества работы различных моделей

Был произведен перебор таких параметров модели, как размерность выхода `LSTM`-ячеек в рекуррентной архитектуре или количество фильтров в сверточной, а также коэффициент Dropout и размер скрытого слоя, и выбраны в качестве используемых в итоге те, которые показывают лучшие результаты на проверочной выборке.

Результаты сравнения приведены в таблице 2.

5.4. Анализ результатов

Как мы видим из таблицы, лучше всего себя показала модель, использующая двунаправленную рекуррентную архитектуру и функцию $AESD$ для анализа сходства текстов вопроса и ответа.

На рисунке 10 представлена ROC-кривая, соответствующая этой модели, из которой, в частности, видно, что благодаря метаинформации наш классификатор выдает высокие оценки вероятности в большинстве своем правильным ответам (кривая смещена влево).

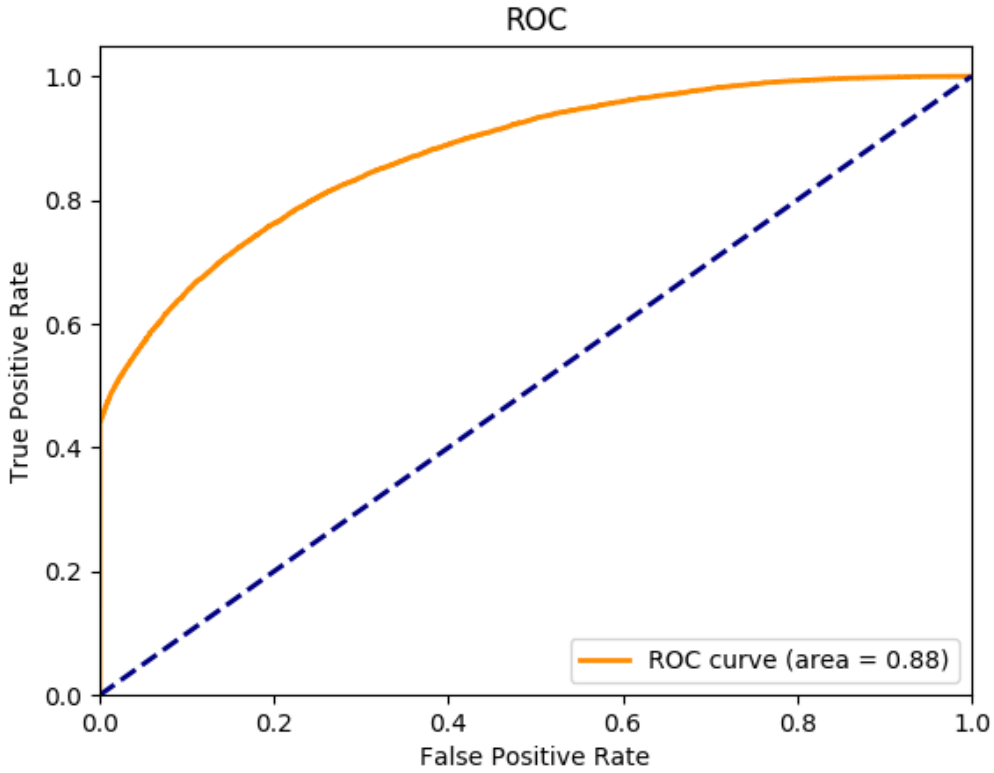


Рис. 10: ROC-кривая

Анализируя полученные результаты, можно отметить следующие вещи:

- Результаты модели, не использующей семантическое представление вопроса совпадают с результатами модели, которая использует только лингвистические и словарные признаки, а также метаинформацию. То есть векторное представление текста ответа в отрыве от вопроса не помогает определить, насколько ответ хорош. В целом, это ожидаемый результат, ведь подобная векторизация создана для передачи смысла текста, что неприменимо в дальнейшем без текста вопроса.

- Были протестированы модели, обучающие параметры сверточных либо рекурентных слоев и совместно для вопросов и ответов, и по отдельности.

  Оказалось, что архитектура с совместными параметрами, во-первых, быстрее обучается, а во-вторых, показывает лучшие результаты на проверочной выборке. Это сходится с показателями других работ по тематике выбора ответа, например [37].

- Были протестированы модели, использующие различные функции определения похожести вопроса и ответа.

  При использовании функций конкатенации и суммы время обучения увеличилось по сравнению с тремя остальными, кроме того, они давали худшие результаты при тестировании на проверочной выборке.

  Несмотря на то, что в работе [37] функция $AESD$ показывала более хорошие результаты, в нашем случае все три остальные функции (косинусный коэффициент, $GESD$ и $AESD$) давали в среднем одинаковые метрики.

- В целом, использование текстовых признаков дало незначительный прирост по сравнению с моделью, которая использует признаки остальных типов. Вероятно, это связано с тем, что большая часть вопросов ожидает описание некоторого процесса, а не какой-то короткий фактологический ответ. Кроме того, даже для человека поставленная задача не является простой: только в 87% случаев ответ с максимальным рейтингом (рейтинг рассчитывается с помощью оценок других пользователей) отмечается автором вопроса как наилучший.

  Тем не менее, реализованная модель превосходит как базовые алгоритмы, так и уже существующие решения, использующие другие методы машинного обучения, по метрикам аккуратности и площади под ROC-кривой, которые наиболее полно описывают качество бинарного классификатора.

Заключение

В рамках данной работы были достигнуты следующие результаты:

- Рассмотрены общие подходы к задаче классификации текстов и существующие решения для поставленной в данной работе задачи, проанализированы их достоинства и недостатки.
- Подготовлен корпус данных, представляющий из себя ответы на вопросы с сайта Server Fault.
- Предложены и посчитаны дополнительные признаки, которые в дальнейшем используются для классификации.
- Разработано несколько моделей бинарных классификаторов на основе рекуррентных и сверточных нейронных сетей.
- Произведено тестирование и сравнение полученных результатов с имеющимися в области исследованиями.

Представленная модель использует идеи рекуррентных нейронных сетей для анализа содержания вопросов и ответов, при этом комбинирует в себе как семантические признаки, так и более классические лингвистические и словарные.

В итоге, в задаче классификации наилучших ответов наш метод позволяет достичь результатов, которые сравнимы по одним метрикам, либо превосходят по другим уже имеющиеся исследования в этой области.

В качестве дальнейшего развития имеющихся моделей можно выделить следующие направления:

- Использование более сложной архитектуры для определения того, решает ли ответ обозначенную в вопросе проблему или нет: например, совместное использование рекуррентных и сверточных слоев, как в работе [45].
- Если пытаться расширить имеющуюся модель на основной сайт StackOverflow, то стоит также учитывать содержание фрагментов кода, которые присутствуют более, чем в половине постов. Для этого может быть применен механизм векторизации для кода [46], который имеет что-то общее с принципом работы Word2Vec.

Список литературы

[1] Stack Overflow // Stack Overflow. — 2018. — Режим доступа: `https://stackoverflow.com` (дата обращения: 01.03.2018).

[2] LeCun Y., Boser B., Denker J. S. et al. Back-propagation applied to handwritten zip code recognition // Neural Computation. — 1989. — Vol. 1, no. 4. — P. 541–551.

[3] Refenes A. N. Managing exchange rate prediction strategies with neural networks // Techniques and Applications of Neural Networks. — 1993.

[4] Wu Yonghui, Schuster Mike, Chen Zhifeng et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation // arXiv preprint arXiv:1609.08144. — 2016.

[5] Vinyals Oriol, Le Quoc V. A Neural Conversational Model // arXiv preprint arXiv:1506.05869. — 2015.

[6] Burel Gregoire, He Yulan, Alani Harith. Automatic Identification of Best Answers in Online Enquiry Communities // ESWC'12 Proceedings of the 9th international conference on The Semantic Web: research and applications. — 2012. — P. 514–529.

[7] Tian Qiongjie, Zhang Peng, Li Baoxin. Towards Predicting the Best Answers in Community-Based Question-Answering Services // Seventh International AAAI Conference on Weblogs and Social Media. — 2013.

[8] Gkotsis George, Stepanyan Karen, Pedrinaci Carlos, Domingue John. It's all in the Content: State of the art Best Answer Prediction based on Discretisation of Shallow Linguistic Features // Proceedings of the ACM conference on Web science. — 2014. — P. 202–210.

[9] Agichtein E., Castillo C., Donato D. et al. Finding High-Quality Content in Social Media // Proceedings of the 2008 International Conference on Web Search and Data Mining. — 2008. — P. 183–194.

[10] Chua A. Y., Banerjee S. So fast so good: An analysis of answer quality and answer speed in community question-answering sites // Journal of the American Society for Information Science and Technology. — 2013. — Vol. 64, no. 10. — P. 2058–2068.

[11] Berger A., Caruana R., Cohn D. et al. Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding // Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. — 2000. — P. 192–199.

[12] Surdeanu M., Ciaramita M., Zaragoza H. Learning to Rank Answers on Large Online QA Collections. — 2008.

[13] Jeon J., Croft W. B., Lee J. H., Park S. A Framework to Predict the Quality of Answers with Non-Textual Features // Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. — 2006. — P. 228–235.

[14] Li B., Jin T., Lyu M. R. et al. Analyzing and Predicting Question Quality in Community Question Answering Services // Proceedings of the 21st International Conference Companion on World Wide Web. — 2012. — P. 775–782.

[15] Agichtein E., Liu Y., Bian J. Modeling Information-Seeker Satisfaction in Community Question Answering // ACM Transactions on Knowledge Discovery from Data. — 2009. — Vol. 3, no. 2. — P. 10:1–10:27.

[16] Freund Yoav, Mason Llew. The Alternating Decision Tree Learning Algorithm // Proceedings of the Sixteenth International Conference on Machine Learning. — Vol. 99. — 1999. — P. 124–133.

[17] Harris Zellig. Distributional structure // Word. — 1954. — Vol. 10. — P. 146–162.

[18] Jones Karen Spärck. A statistical interpretation of term specificity and its application in retrieval // Journal of Documentation. — 1972. — Vol. 28. — P. 11–21.

[19] Mikolov Tomas, Chen Kai, Corrado Greg, Dean Jeffrey. Efficient Estimation of Word Representations in Vector Space // arXiv preprint arXiv:1301.3781. — 2013.

[20] Nair Vinod, Hinton Geoffrey E. Rectified Linear Units Improve Restricted Boltzmann Machines // Proceedings of the 27th International Conference on International Conference on Machine Learning. — 2010. — P. 807–814.

[21] Bojanowski Piotr, Grave Edouard, Joulin Armand, Mikolov Tomas. Enriching Word Vectors with Subword Information // arXiv preprint arXiv:1607.04606. — 2016.

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

[23] Schuster Mike, Paliwal Kuldip K. Bidirectional recurrent neural networks // IEEE Transactions on Signal Processing. — 1997. — Vol. 45, no. 11. — P. 2673–2681.

[24] Kim Yoon. Convolutional neural networks for sentence classification // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. — 2014. — P. 1746–1751.

[25] Server Fault // Stack Overflow. — 2018. — Режим доступа: https://serverfault.com (дата обращения: 01.03.2018).

[26] Overflow Stack. Stack Exchange Data Dump // Internet Archive. — 2018. — Режим доступа: https://archive.org/details/stackexchange (дата обращения: 13.03.2018).

[27] Wikipedia. Document Object Model // Wikipedia, the free encyclopedia. — 2016. — Режим доступа: https://ru.wikipedia.org/wiki/Document_Object_Model (дата обращения: 03.01.2016).

[28] Wikimedia. Wikimedia database dump of the English Wikipedia on October 01, 2017 // Internet Archive. — 2017. — Режим доступа: https://archive.org/details/enwiki-20171001 (дата обращения: 10.01.2017).

[29] Smith E. A., Senter R. J. Automated readability index // AMRL-TR-66-220. — 1967.

[30] Flesch Rudolf. A new readability yardstick // Journal of Applied Psychology. — 1948. — Vol. 32. — P. 221–233.

[31] McLaughlin G. Harry. SMOG grading: A new readability formula // Journal of Reading. — 1969. — Vol. 12. — P. 639–646.

[32] Kincaid J. Peter, Fishburne Robert P., Rogers Richard L., Chissom Brad S. Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy enlisted personnel // Research Branch Report. — 1975. — Vol. 8-75.

[33] Coleman Meri, Liau T. L. A computer readability formula designed for machine scoring // Journal of Applied Psychology. — 1975. — Vol. 60. — P. 283–284.

[34] Gunning Robert. The Technique of Clear Writing. — McGraw-Hill, 1952. — P. 36–37.

[35] Björnsson Carl-Hugo. Läsbarhet. — Stockholm: Liber, 1968.

[36] Jaccard Paul. Étude comparative de la distribution florale dans une portion des Alpes et des Jura // Bulletin de la Société Vaudoise des Sciences Naturelles. — 1901. — Vol. 37. — P. 547–579.

[37] Feng Minwei, Xiang Bing, Glass Michael R. et al. Applying Deep Learning to Answer Selection: A Study and An Open Task // arXiv preprint arXiv:1508.01585. — 2015.

[38] Kingma Diederik P., Ba Jimmy. Adam: A Method for Stochastic Optimization // arXiv preprint arXiv:1412.6980. — 2014.

[39] Keras. — 2018. — Режим доступа: https://keras.io/ (дата обращения: 01.03.2018).

[40] Gensim. — 2018. — Режим доступа: https://radimrehurek.com/gensim/ (дата обращения: 01.03.2018).

[41] NLTK. — 2018. — Режим доступа: https://www.nltk.org/ (дата обращения: 01.03.2018).

[42] textstat. — 2018. — Режим доступа: https://github.com/shivam5992/textstat (дата обращения: 01.03.2018).

[43] Podguzov Nikita. StackOverflow Correct Answer Detection // Github. — 2018. — Режим доступа: https://github.com/Nikitosh/StackOverflow-Correct-Answer-Detection (дата обращения: 01.03.2018).

[44] Scikit-Learn. — 2018. — Режим доступа: http://scikit-learn.org/stable/index.html (дата обращения: 01.03.2018).

[45] Lai Siwei, Xu Liheng, Liu Kang, Zhao Jun. Recurrent convolutional neural networks for text classification // Proceedings of the 2015 Conference of the Association for the Advancement of Artificial Intelligence. — 2015. — P. 2267–2273.

[46] Mou Lili, Li Ge, Liu Yuxuan et al. Building Program Vector Representations for Deep Learning // arXiv preprint arXiv:1409.3358. — 2014.