МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра дискретной математики и алгоритмики

СВИДУНОВИЧ Денис Владимирович

СИСТЕМА ПОИСКА ОТВЕТОВ В ДОКУМЕНТЕ

Магистерская диссертация

специальность 1-31 81 09 «Алгоритмы и системы обработки больших объемов информации»

Научный руководитель
Соболевская Елена Павловна
кандидат физико-математических наук, доцент

Допущен к защите
«____» _____ 2020 г.
Зав. кафедрой дискретной математики и алгоритмики
_____ В.М. Котов
доктор физико-математических наук, профессор

Минск 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последнее время интерес к области обработки естественных языков (NLP) непрерывно растет. Гиганты рынка ИТ вкладывают огромное количество ресурсов для развития данного направления. Это в первую очередь связано с распространенностью и большой популярностью поисковых систем, личных голосовых ассистентов, голосовых интерфейсов и «умных» гаджетов. Благодаря развитию нейронных сетей, такие программные средства перешли на новый уровень качества, тем самым став полезными и удобными для пользователей. Производители, наблюдая за большим интересом пользователей, ведут жесткую борьбу на данном рынке, создавая программное средство все лучше «понимающее» человека, позволяя ему решать задачи быстрее и качественнее.

Автоматическая обработка естественных языков лежит в основе множества современных программных средств, создавая возможность человеку «общаться» с ЭВМ. Кроме приведенных выше массивных программных систем, с помощью алгоритмов и подходов NLP решают задачи:

– машинного перевода;
– автоматического реферирования и упрощения текста;
– извлечения информации из текста;
– анализа эмоциональной окраски текста;
– распознавания текста с изображений и аудио;
– генерирования текста;
– синтеза речи;
– и др.

Из-за такого глубокого проникновения данных технологий в жизнь людей, трудно себе представить современный мир без них, что еще раз подчеркивает их важность и необходимость развития.

Подходы к решению задач в области автоматической обработки естественных языков изучаются и развиваются уже несколько десятилетий, однако по-прежнему даже самые лучшие системы имеют множество недостатков и не могут корректно обработать все входные данные. В частности, одной из таких не решенных задач является обучение ЭВМ оперировать с текстом не только как с набором символов, но и как со смысловой единицей. Иначе говоря, задача состоит в том, чтобы научить компьютер читать.

Задача научить компьютер читать и понимать текст является в некотором смысле базовой для большинства систем автоматической обработки текстов, иногда можно увидеть, что ее называют святым Граалем искусственно-

го интеллекта, так как хорошее решение позволит сделать огромный скачок к решению множества задач компьютерной лингвистики и обработки естественных языков.

В данной магистерской диссертации проводится исследование в области понимания прочитанного (от англ. reading comprehension). Само по себе «понимание» - это тяжело формализуемое понятие, и как замерить, что алгоритм A понимает текст лучше, чем алгоритм B, никто не знает. На практике это понимают косвенно, используя более сложные задачи. Например, можно сравнить существующий алгоритм машинного перевода с его модификацией в части обработки и представления входного текста, при улучшении метрик перевода модифицированным алгоритмом относительно оригинального, можно сказать, что модель стала лучше учитывать контекст текста, улавливать синонимию языка, зависимости и характеристики объектов, то есть улучшилось «понимание» текста. Получается, что для разработки в области reading comprehension невозможны без работы над задачами более высокого уровня. В данной работе такой задачей выступает поиск ответов на вопросы (от англ. questing answering). Поставленная задача очень сильно отражает способность алгоритма читать: для правильно ответа нужно хорошо понимать о чем вопрос, учитывать его мельчайшие детали, правильно определять контекст, иметь некоторые знания о внешнем мире.

Цель данной работы - разработать систему поиска ответов на вопросы в тексте. В качестве входных данных система должна принимать текст и вопрос (или список вопросов), на выходе для каждого заданного вопроса система должна предоставить ответ из текста, либо вернуть значение, соответствующие отсутствию ответа в тексте.

Задача данной работы - разработать подходы машинного обучения для решения задачи Question answering, провести экспериментальное исследования и сравнительный анализ их качества, подобрать гиперпараметры для разработанных методов, оптимизировать лучший метод для уменьшения потребления вычислительных ресурсов.

1 ОСНОВНЫЕ ПОНЯТИЯ И ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В данном разделе рассмотрены основные теоретические сведения, которые используются на протяжении всей работы в качестве базовых конструкций.

1.1 Искуственная нейронная сеть

Искуственная нейронная сеть - это математическая модель, основанная на идее работы мозга живых существ. Высокоуровнево нейронная сеть представляет собой множество нейронов, соединенных специальным образом между собой. Когда раздражитель в достаточной мере воздействует на определенные нейроны, они активируются и передают сигнал всем связанным нейронам, если в какой-то нейрон поступает достаточное количество сигнала, то он в свою очередь тоже активируется и передает сигнал дальше. Каждая связь имеет свою силу, чем большая сила связи между нейронами, тем больший сигнал передается по ней при активации одного из нейронов. Эти основополагающие принципы были заложены в искуственных нейронных сетях (далее понятия нейронной сети и искуственной нейронной сети значат одно - математическую модель). Как правило нейроны организовываны в слои, во всех слоях кроме первого каждый нейрон связан с каждым нейроном предыдущего слоя, раздражитель воздействует на нейроны первого слоя, и сигнал проходит от него к последнему, такую нейронную сеть называют полносвязной. После прохождения сигнала по нейронной сети, ожидается что нейроны последнего слоя будут хранить уже обработанный сигнал. Например, если нейронная сеть на вход принимает значения цвета изображения в каждой его точке и цель состоит в том, чтобы определить цифру на изображении, то выходной слой может состоять из 10 нейронов соответствующих каждой цифре: если в результате активировался i-ый, то этот сигнал можно интерпретировать как распознавание i-ой цифры. Получается, что обучение нейронных сетей - это поиск таких весов связей между нейронами и их активаций, при которых выходной сигнал лучше всего решает поставленную задачу.

Таким образом по описанной выше модели, результат работы одного нейрона сети можно записать с помощью формулы 1.1:

$$o_j^{(i)} = activation(\sum_{k=1}^{|o^{(i-1)}|} w_{jk}^{(i)} o_k^{(i-1)} - bias_j^{(i)}) \tag{1.1}$$

где o^i — вектор результатов активаций каждого нейрона i-го слоя;
o^i_j — значение активации j-го нейрона i-го слоя;
$w^{(i)}_{jk}$ — сила связи между j-ым нейроном i-го слоя, и k-тым нейроном $i-1$-го слоя;
$bias^{(i)}_j$ — порог активации j-го нейрона i-го слоя;
$activation(x)$ — функция активации нейрона.

Назовем все веса связей $w^{(i)}_{jk}$ и пороги активации $bias^{(i)}_j$ обучаемыми параметрами нейронной сети, и обозначим их θ, а с помощью $f(x,\theta) = o^{(d)}$, где d-глубина нейронной сети, обозначим результат работы нейронной сети на объекте x. Тогда качество работы нейронной сети на обучающей последовательности $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ можно оценить по формуле 1.2:

$$L(S,\theta) = \sum_{x_i, y_i \in S} l(y_i, f(x_i,\theta)) \tag{1.2}$$

где x_i — оцениваемый объект;
y_i — истинная метка объекта x_i;
$l(y,\hat{y})$ — функция потерь на одном объекте, на вход принимает истинные целые значения, и рассчитанные целевые значения, результат отображает степень ошибки.

Используя введенные выше обозначения, обучение нейронной сети можно записать используя формулу 1.3:

$$\theta = \underset{\theta}{\operatorname{argmin}}\, L(S,\theta) \tag{1.3}$$

Пространство перебора $\theta \in \mathbb{R}^{|\theta|}$ слишком велико, поэтому задачу решают с помощью метода градиентного спуска или его модификации. Это накладывает ограничения: во-первых функция активации и функция потерь должны быть гладкими, во-вторых метод может найти локальный минимум. Тем не менее на данный момент такой вариант обучения чуть ли не единственный.

В данной работе используется два основных вида нейронных сетей(в том числе их комбинация):

а) Нейронная сеть с прямой связью (от англ. feed forward) - сеть, в которой не бывает циклов распространения сигналов, то есть он продвигается

строго от первого слоя к последнему. Иногда слои полносвязной сети называют Dense-слоями.

б) Рекурентная нейронная сеть - класс нейронных сетей, которые могут образовывать циклы распространения сигнала, а также обладать собственной памятью. Эффективны при обработке последовательностей и входных данных варьирующейся длины.

1.2 Представления текстовых данных

Как было показано в 1.1, нейронные сети работают только с вещественными данными, поэтому для того, чтобы применить их на текстах используют подход мешка слов (от англ. BOW - Bag Of Words). Алгоритм заключается в том, что входное текстовое поле каким-то образом разбивается на токены, в качестве токенов могут выступать n-грамы, слова, n-грамы уровня слов, предложения и др., в зависимости от задачи. На основании обучающей выборки данных строят словарь известных токенов V, и присваивают каждому токену его порядковый номер в словаре. Теперь для любого токена из словаря можно получить его индекс, однако подавать на вход нейронной сети индексы токенов не эффективно, потому что по принципу своей работы сеть оперирует значениями цифр, т.е. например три синонимичных токена могли получить значения $\{1, \frac{|V|}{2}, |V|\}$ в итоге скорее всего задача окажется слишком сложной и представленная синонимия не будет выявлена, так как индекс является случайной величиной и заставляет нейронную сеть искать связь там где ее нет, либо выучивать сложные конструкции только для того, чтобы определять где какой токен.

Для решения этой проблемы используют подход one-hot encoding: на основании индекса токена i формируется вектор размера $|V|$, где на i-ой позиции стоит 1, а на всех остальных 0. Таким образом в нейронную сеть не поступает никакой информации о фальшивой близости и при обучении не происходит «поиск суеверий».

После кодирования каждого токена входной последовательности токенов, к ним применяют так называемый слой представления, который для каждого токена формирует некоторый вектор в пространстве $\mathbb{R}^d$, обучаемый нейронной сетью, и хранящий информацию о слове. Такое преобразование необходимо для сжатия, очевидно что в one-hot векторах информация хранится не самым эффективным способом, кроме того в большинстве случаев из-за огромного размера словаря V без этого в принципе нельзя было бы рассчитывать на быструю работу сети. Полученные вектора обычно называют

представлениями или эмбеддингами(от англ. embeddings).

Дальше, если используется полносвязная нейронная сеть, то все представления входного поля агреггируют в вектор фиксированной длины, а если используется рекурентная нейронная сеть, то каждое представление последовательно подают на вход сети. В качестве агрегирующей функции обычно лучше всего себя показывает среднее арифметическое, либо максимум.

1.2.1 Byte pair encoding

Достаточно долгое время, да и по сей день, во многих задачах обработки текстовых полей с помощью нейронных сетей, использовался словарь токенов, состоящий из слов или символов, например, в таких известых предобученных представлениямх как Word2Vec, GloVe или FastText. Но у таких подходов есть существенные недостатки, у подхода на словах:

– модель вынуждена хранить словари на сотни тысяч слов, и соответственно представления к ним, занимая слишком много пространства в памяти;

– для модели тяжело понять, что перед ней две формы одного и того же слова, а не совершенно разные слова, если же применить лемматизацию, то теряется ценная информация о синтаксисе;

– из-за ограничений на размер словаря всегда встречаются слова, которых нет в словар, и вносят шум в предсказание.

У подходов на символах тоже есть недостатки:

– для модели еще более тяжело понимать такую сущность как слово, и пытаться строить какую-нибудь языковую модель или понимать смысл, так как ей нужно дополнительно выучить много информации о том, как строить слова из символов;

– сам по себе подход плохо работает с feed forward сетями, потому что теряется информация о порядке;

– здесь все еще более загадочно в понимании моделью форм одного и того же слова.

Так как при разработке нейронной сети, работающей с текстовыми полями, очень важно, чтобы она максимально хорошо понимала текст, то ни первый, ни второй подход для этого не подходит, а использование обоих методов сразу не позволяет представить входное текстовое поле в виде последовательности токенов. Нужно использовать нечто среднее, и таким средним оказался алгоритм BPE, который был разработан еще в 90-ые годы, но успешно применен в нейронных сетях совсем недавно. Ниже представлен алгоритм

работы BPE:

а) Словарь Инициализуется символами;

б) В тексте подсчитываются все пары символов;

в) Находится самая популярная пара символов;

г) Создается новый символ состоящий из самой популярной пары, в тексте пара заменяется новым символом;

д) Шаги б-г Повторяются до тех пор, пока словарь не приобретет желаемый размер.

Благодаря тому, что в основе такой словарь на символах проблема неизвестных слов решается, в тоже время достаточно популярные слова будут представлены одним символом, а формы одного и того же слова, скорее всего будут иметь корень как один символ в словаре. Данный метод создает баланс между рассмотренными ранее и использует плюсы обоих, лишаясь почти всех недостатков.

1.3 Долгая краткосрочная память

В [1] было показано, что хоть обыкновенные рекурентные сети в теории и могут выявить зависимости в длинных последовательность на практике обычно достичь такой спасобности не удается, по крайней мере с использованием актуальных методов обучения.

Для решения данной проблемы в [2] была разработана архитектура специальных рекурентных сетей под названием «Долгая краткосрочная память» или Long Short Term Memory (LSTM). Главная ее особенность в том, что у сети есть собственная память, которая обновляется специальными операциями на основе поступивших входных данных. Т.е. на архитектурном уровне в сеть вложена способность запоминать информацию на больших последовательностях, с чем она довольно успешно справляется на большом количестве задач.

Для изучения LSTM поставим задачу закодировать последовательность токенов x_t так, чтобы закодированное представление содержало информацию о всей последовательности.

На рисунке 1.1 представлена архитектура LSTM рекурентной нейронной сети.

Ключевая идея LSTM состоит в так называемом cell state - состоянии ячейки, на рисунке 1.1 это состояние отмечено с помощью символа C, перед обработкой токена t внутреннее состояние ячейки равно C_{t-1}, после обработки - C_t. Внутреннее состояние ячейки используется как собственная память

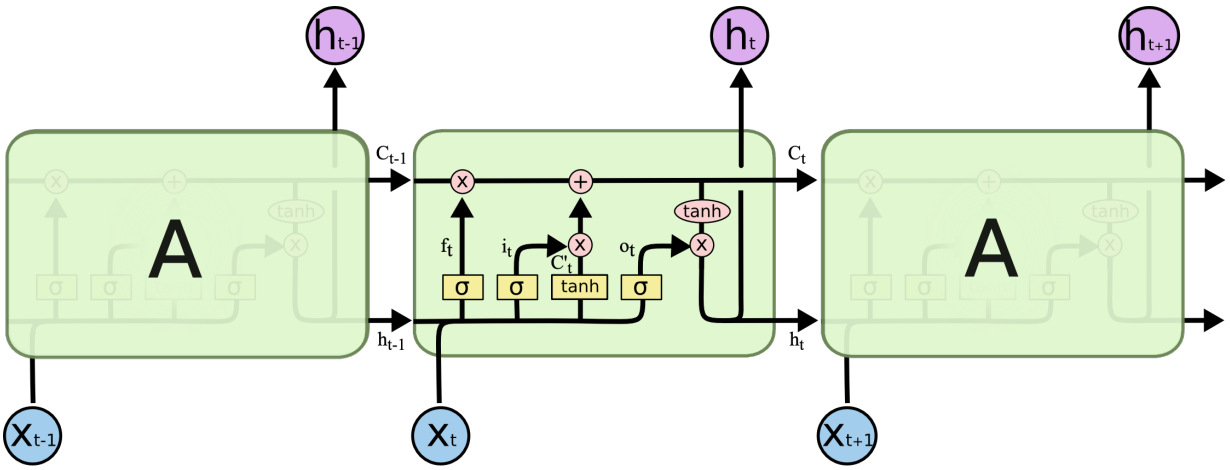

Рисунок 1.1 – Архитектура LSTM рекурентной нейронной сети

сети, а в качестве результата кодирования используется вектор h_t, в который кодируется необходимая информация о текущем токене и предыдущих.

После поступления нового токена x_t сеть решает какую информацию нужно удалить из внутреннего состояния с помощью полносвязного слоя f_t представленного в формуле 1.4, так например ячейка может удалить информацию о том, что была открывающая скобка, если на вход пришла закрывающая.

$$f_t = \sigma(W_f[h_{t-1}; x_t] + b_f) \tag{1.4}$$

где σ — сигмойда;
W_f, b_f — тренеруемые параметры сети;
$[a; b]$ — конкатенация векторов a, b.

На следующем шаге формируется обновление внутреннего состояния $C_t^{'}$, и маска i_t, определяющая какие значения состояния нужно обновлять, далее полученное обновление по маске применяется к внутреннему состоянию. В формулах 1.5, 1.6 и 1.7 описан процесс обновления внутреннего состояния.

$$i_t = \sigma(W_i[h_{t-1}; x_t] + b_i) \tag{1.5}$$

$$C_t^{'} = tanh(W_C[h_{t-1}; x_t] + b_C) \tag{1.6}$$

$$C_t = f_t * C_{t-1} + i_t * C_t^{'} \tag{1.7}$$

Результат кодирования формируется на основе состояния ячейки. Для этого рассчитывает маска по прошлому выходу и новому токену в маску o_t записывается то, в какие ячейки внутренней памяти нужно смотреть. На ос-

нове маски считаетс выход сети по формулам 1.8 и 1.9:

$$o_t = \sigma(W_o[h_{t-1}; x_t] + b_o) \tag{1.8}$$

$$h_t = o_t \; tanh(C_t) \tag{1.9}$$

Рассмотренную выше архитектуру обычно применяют в обе стороны последовательности, т.е используется две независимых ячейки LSTM, результирующие закодированные представления которых конкатенируются. Хоть LSTM и работает лучше, чем стандартные RNN, эксперименты показывают, что метод приносит дополнительное качество на многих задачах.

2 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Question answering - это информационная система, которая способна отвечать на вопросы человека на естественном языке.

Первые вопросно-ответные системы основывались на поиске по базе знаний. База знаний (от англ. knowledge base) - это база данных, содержащая некоторую информацию о человеческом опыте в определенных областях. Так как один и тот же вопрос можно задать разными способами, то первоначальный вопрос обрабатывался. Если это было возможно, сложные вопросы разбивались на множество простых, в примитивных системах слова вопроса приводили в начальную форму и искали целиком, в более сложных из каждого вопроса извлекали нужные категории, например, (субъект, отношение, объект) и искали в базе по имеющимся категориям недостающие. При необходимости система просила уточнения, и если в базе знаний был ответ - выдавала его. Таким образом качество QA системы напрямую зависели от размера и качества базы знаний.

По алгоритму выше было довольно сложно создать хорошую вопросно-ответную систему, способную ответить на вопросы из различных областей, они были специализированными. QA системы делят на:

– Специализированные (от англ. Closed-domain) - QA системы, осуществляющие поиск в заранее установленных областях.

– Общие (от англ. Open-domain) - QA системы, осуществляющие поиск по всем доступным областям в базе знаний.

Для того, чтобы перейти от специализированных к общим QA системам использовали разные ухищрения: формировали ответ на вопрос в каждой из имеющихся областях знаний, и мета-алгоритм решал, какой из предложенных ответов подходит к вопросу больше всего.

В том или ином виде, почти все современные QA системы имеют базу знаний. Чаще всего она хранится в виде графа, где вершины представляют собой разные сущности, а ребра - разные виды отношений между сущностями. Такой подход называют графом знаний (от англ. knowledge graph), он позволяет с помощью моделей машинного обучения оценить отношения между сущностями, информации о которых нет в базе, на основе похожих вершин, смежных интересуемой вершине или имеющихся видов ребер. Также сейчас широко используются QA системы, основанные на информационном поиске: вся имеющаяся информация представляется в виде документов, которые ранжируют в соответствии с вопросом, дальше в зависимости от системы и выполняемых ей задач выдаются лучшие документы, или происходит уточ-

ненный поиск среди лучших документов, или формируется ответ совместно с алгоритмами, основанными на графах знаний.

Задача искать документы, в которых скорее всего есть ответ на поставленный вопрос, уже очень хорошо решена, сейчас из-за стремительного развития голосовых ассистентов наибольший интерес представляет задача поиска ответа в документе, т.к. новые системы имеют возможность дать только один ответ, то и стоимость ошибки гораздо больше. В данной магистерской диссертации разрабатывается именно такая система: поиска ответа в документе.

2.1 Исходные данные

2.1.1 SimpleQuestions

Представляет собой QA датасет общего типа из 100 тыс. вопросов [3]. Датасет состоит из вопросов, и соответствующей им базы знаний:

query, (subject, relationship, object)

где query — вопрос;
subject — субъект;
relationship — отношение между объектом и субъектом;
object — объект, или ответ на вопрос.

Например:
What American cartoonist is the creator of Andy Lippincott?
(andy lippincott, character created by, garry trudeau)

Which forest is Fires Creek in?
(fires creek, containedby, nantahala national forest)

Таким образом, авторы датасета SimpleQuestions предлагают создавать систему QA, основывающуюся на базе знаний. Задача состоит в улучшении выделение субъекта и отношения из вопроса.

2.1.2 News QA

QA датасет общего типа из 100 тысяч вопросов по 10 тысячам новостям CNN [4]. Включает в себя большое количество различных тем: политику, экономику, текущие события и др. Был собран с помощью краудсорсинга. Представляет собой набор кортежей:

(Story, Question, Answer)

где Story — новость;
Question — вопрос;
Answer — ответ.

Ответом является непрерывная последовательность символов из новости. Последовательность может быть пустой, если в новости не имеется ответа на поставленный вопрос.

Авторами был исследован разрыв между пониманием текста человеком и сильной рекуррентной нейронной моделью, он составил 0.198 F_1 меры.

2.1.3 MS Marco

MAchine Reading COmprehension dataset [5] - датасет, состоящий из больше 1 млн. анонимных вопросов, сэмлированных из поисковых логов поисковика Bing, с генерированными на основе документов ответами из которых 180 тыс. исправлены в ручную. В данном датасете для запроса может быть как множество ответов, так и ни одного. С помощью предоставленных данных авторы предлагают решать 3 задачи:

– Предсказать, правда ли на вопрос можно ответить по предоставленным документам.

– Синтезировать ответ на естественном языке в контесте предоставленных документов.

– Для запроса отранжировать предоставленные документы.

2.1.4 SQuAD v1.1

Stanford Question Answering Dataset [6] - краудсорсинговый датасет общего типа, состоящий из более 100 тыс. вопросов к статьям википедии, где ответ на каждый вопрос - это непрерывный сегмент текста в соответствующей статье.

(Passage, Question, Answer)

где Passage — статья из wikipedia;
Question — вопрос;
Answer — ответ (возможно пустое значение или множество ответов).

2.1.5 SQuAD v2.0

Представляет собой улучшенный датасет SQuAD v1.1 [7]. В новой версии появились 50 тыс. пар (вопрос, документ) для которых в документе не имеется ответа. Эти вопросы составлены таким образом, чтобы их было тяжело отличить от тех, на которые ответ есть. Кроме того, были существенно отфильтрованы вопросы, размеченные людьми, плохо справляющимися с заданием. Так же по сравнению с предыдущей версией для каждого вопроса в среднем использовалось перекрытие 4.8, вместо 1. Благодаря таким изменениям, сложность предлагаемой задачи очень сильно увеличилась:

– Теперь модель должна не просто находить ответ, а определять есть ли он в тексте.

– Негативные вопросы (не имеющие ответа) очень сильно похожи на позитивные, в частности некоторые из них были сгенерированы добавлением частиц не, заменой слова на антоним, изменением даты и других методами на основе позитивных вопросов. Такой механизм очень сильно усложняет задачу, так как модель должна еще лучше «понимать» текст.

Благодаря всем изменениям нейронная модель, которая на SQuAD v1.1 имела точность 86% F_1 меры, на SQuAD v2.0 дает 66%. Так же была исследована точность людей на этом датасете - 89.452% F_1 меры, а лучший машинный результат отстает на больше 6%.

Так как корпус оценен в среднем с перекрытием 4.8, он является одним из самых качественных на данный момент. Учитывая это и перечисленные выше его преимущества, именно он был выбран для экспериментов в данной магистерской работе.

2.1.6 SberQuAD

Под конец выполнения данной магистерской работы был опубликован новый QA корпус - Sberbank Question Answering Dataset [8]. Датасет состоит из 50 тысяч вопросов в тренировочной выборке и 15 тысяч вопросов в валидационной, тестовая выборка не была опубликована и использовалась для проведения соревнования.

По способу генерации датасет очень похож на SQuAD v1.1: в качестве исходных данных использовались параграфы страниц русскоязычной википедии, для генерирования вопросов эти параграфы отдавались на краудсорсинговую платформу Яндекс.Толока, где за плату люди придумывали различные вопросы к каждому параграфу и выписывали ответы на них.

Помимо русского языка из отличий от SQuAD v1.1 можно выделить следующие:

– Для каждого вопроса известен только параграф к которому он задается, и не известна вся статья.

– Для каждого вопроса нет набора позиций ответов в тексте параграфа, ответ задан строкой.

– Ничего не известно про уровень качества достигаемый человеком на датасете. Есть риск, что некоторая большая доля датасета представляет собой шум.

Для исследований русских вопросно-ответных систем авторы данных предоставляют два baseline решения, относительно которых предлагается улучшать качество решений.

Так как датасет появился уже после проведения большого количества работы по англоязычным вопросно-ответным системам, и на текущей момент еще является достаточно сырым, то в данной магистерской диссертации он почти не используется. Однако, его нельзя пропускать, в 3.3.3 был проведен эксперимент с применением одного из разработанных подходов с использованием этих данных.

2.2 Метрики оценки качества

Полное совпадение (от англ. Exact match - EM) - метрика, измеряющая процент предсказаний, полностью совпадающий с истинными ответами 2.1.

$$EM(X) = \frac{1}{n}\sum_{i=1}^{n}\max_{j}([a(x_i) = y_{ij}]) \tag{2.1}$$

где X^n — множество объектов (query; text);

$a(x_i)$ — ответ, предсказанный тестируемым алгоритмом $a(x)$ на объекте x_i;

y_i — множество правильных ответов для объекта x_i.

(Macro-averaged) F_1 score - метрика измеряющая максимальное пересечение между предсказанным и истинным ответом по формуле 2.2. Все ответы (как истинные, так и предсказанные) представляются в виде мешка токенов (от англ. bow - bag of words). Для каждого предсказанного ответа находится максимальный по F_1 мере истинный ответ, после чего такая характеристика

усредняется по всем вопросам тестируемого множества.

$$F_1(X) = \frac{1}{n}\sum_{i=1}^{n}\max_j(F_1(a(x_i), y_{ij})) \tag{2.2}$$

где X^n — множество объектов (query; text);
$a(x_i)$ — ответ, предсказанный тестируемым алгоритмом $a(x)$ на объекте x_i;
y_i — множество правильных ответов для объекта x_i;
$F_1(bow_{pred}, bow_{true})$ — функция F_1 меры 2.3 по предсказанному набору токенов bow_{pred} и истинному набору токенов bow_{true}.

$$F_1(bow_{pred}, bow_{true}) = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.3}$$

где $precision(bow_{pred}, bow_{true})$ — функция «точности» 2.4, характеризующая то, какая часть предсказанных токенов в bow_{pred} является правильной, то есть присутствует в bow_{true};
$recall(bow_{pred}, bow_{true})$ — функция «полноты» 2.5, характеризующая то, какая часть правильных токенов из bow_{true} присутствует в ответе модели bow_{pred}.

$$precision(bow_{pred}, bow_{true}) = \frac{|bow_{pred} \cap bow_{true}|}{|bow_{pred}|} \tag{2.4}$$

$$recall(bow_{pred}, bow_{true}) = \frac{|bow_{pred} \cap bow_{true}|}{|bow_{true}|} \tag{2.5}$$

Для разработки модели, решающей поставленную задачу необходимы обе метрики: exact match и f1 мера. В то время как exact match высокоуровнево отображает качество системы целиком, f1 мера более чувствительна и нужна для того, чтобы замечать малейшие улучшения алгоритма и небольшими изменениями двигаться в сторону улучшения exact match.

2.3 Сложности при решении задачи поиска ответов на вопросы

Задача поиска ответов на вопросы на данный момент не является решенной, для разработки качественной системы необходимо решить ряд проблем, связанных с обработкой и понимаением естественных языков, основные из них:

– Различная лексика запросов и документов. Используя естественный язык, одну и туже мысль можно выразить разными способами. Кроме того, что при обработке вопроса можно не найти ответ, из-за использования синонимов, проблема выражена в том, что язык запросов и язык документов - это разные языки. Хоть они и используют одни и теже элементы при построении предложений - слова, их распределение в языке запросов и документов разные. Это наиболее острая проблема в построении вопросно-ответных систем, она проявлялась как в более ранних разработках, так как и сейчас. Системы информационного поиска наиболее подвержены влиянию лексических различий, так как для решения поставленной задачи им нужно не просто «понимать» текст, но и сопоставлять его, статистические различия текстов усложняют сопоставление.

– Омонимия, или неоднозначность - это ситуация, когда вопросу, выражению или произвольной последовательности языковых знаков можно дать несколько толкований. При разработке вопросно-ответных систем чаще всего сталкиваются с лексической и синтаксической омонимией. Лексические омонимы - это такие слова, которые имеют одинаковую письменную форму, но могут иметь различные интерпретации в зависимости от контекста: коса, мир, клуб, некоторые названия городов и т.д.; омонимы такого типа встречаются чаще всего и сильно ухудшают качество лучших на данный момент нейросетевых подходов, так как в этом классе решений лексемы представляют вектором, характеризующим их в некотором пространстве, при этом не используя разные вектора для ортогональных интерпретаций. Синтаксическая омонимия проявляется когда невозможно однозначно установить синтаксические связи. Например, «положи тетрадь в клетку», можно интерпретировать «в клетку» и определением, и обстоятельством. Этот вид неопределенности чаще всего вызван лексической омонимией и встречается достаточно редко.

– Сложные вопросы. Современные системы ответов на вопросы могут правильно ответить на большинство простых вопросов, проблемы начинаются, когда для ответа на вопрос нужно найти несколько фактов и сформировать ответ скомбинировав их. Но вопросы могут быть сложными не только с

точки зрения поиска ответа, но и структурно. Когда вопрос не четко сформулирован, очень большой или в нем множество объектов, алгоритму может быть трудно понять относительно чего ищет пользователь.

– Многоязычность. Вопросно-ответным системам приходится обрабатывать как запросы пользователей на разных языках, так и документы на разных языках. Это серьезная проблема, так как качество и количество данных на разных языках разное, и ситуация, когда система может найти ответ на одном языке и не может на другом не редкая. Эта проблема не касается данной работы, так как будет использоваться корпус на английском языке.

2.4 Лучшие подходы построения вопросно-ответных систем

По открытому рейтингу SQuAD v.2 видно, что лучшие результаты достигаются при помощи ансамблей нейросетевых моделей. Далее будут рассмотрены некоторые из них.

2.4.1 Syntax-Guided Machine Reading Comprehension

В основе рассматриваемого подхода [9] использовался механизм так называемого внимания [10]. При чтении и понимании прочитанного, люди обычно уделяют основную часть своего внимания ключевым словам, в то время как остальные слова пробегают поверхностно. Авторы работы заметили, что существующие подходы к реализации внимания в равной степени оценивают внимание к каждому слову предложения, без предварительного фокуса на определенные слова, из-за чего часть «внимания» размывается по неважным словам, и затрудняет оценку полезности важных слов, что вызывает ухудшение точности reading comprehension моделей при работе с большими предложениями. Предлагается решить эту проблему введя ограничения, основанные на синтаксисе предложений, а именно, предварительно разобрав синтаксические зависимости предложения, при обработке лексемы использовать существующий механизм внимания только на тех лексемах, которые синтаксически зависимы от текущей 2.1.

Модель обучается на кортежах $(passage, query, answer)$,

где passage — это набор предложений, в котором ищется ответ;
query — это вопрос, на который ищется ответ;
answer — это смещение начала и конца ответа в passage.

На вход подается конкатенация $passage$ и $query$ разделенные специальным символом, а на выходе для каждой позиции $passage$ считается вероятность

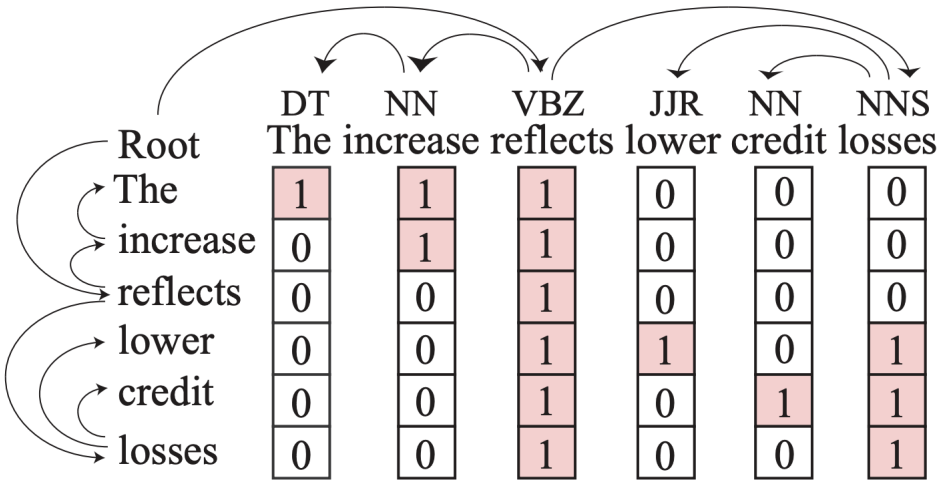

Рисунок 2.1 – Маска важности внимания, построенная по дереву зависимостей

того, что в этом месте начинается ответ и вероятность, что заканчивается.

Прежде чем построить представление каждого предложения, сначала их подают на вход синтаксическому парсеру, по результатам которого строится маска важности внимания 2.6:

$$M[i,j] = \begin{cases} 1 & \text{если } j \in P_i \vee j = i \\ 0 & \text{иначе} \end{cases} \tag{2.6}$$

где P_i — множество всех синтаксически зависимых лексем входного предложения для лексемы i.

Для кодирования входных данных в векторную форму используется классическая модель BERT [11]. Выход последнего слоя H которой проецируется в $key, value, query$ представления обозначаемые K_i, Q_i, V_i для каждой головы i внимания. Дальше для получения весов внимания A_i для каждой пары $(key, value)$ с учетом маски важности внимания, производят скалярное произведение 2.7:

$$A_i = Softmax\left(\frac{M \cdot (Q_i K_i^T)}{\sqrt{d_k}}\right) \tag{2.7}$$

После получения представления токенов на основе синтаксического внимания $W_i = A_i V_i$, их направляют в обычную feedforward нейронную сеть, которая для каждой позиции предсказывает вероятность начала и конца ответа в ней.

Для обучения нейронной сети в качестве функции потерь используется

перекрестная энтропия 2.8:

$$L = y_s \log s + y_e \log e \tag{2.8}$$

где y_s — вектор, полученный на основе входных данных, в котором позиция начала ответа в $passage$ имеет значение 1, а все остальные 0;

y_e — вектор, полученный на основе входных данных, в котором позиция конца ответа в $passage$ имеет значение 1, а все остальные 0;

s — посчитанный моделью вектор, в котором значение на позиции i хранит вероятность того, что в этой позиции начинается ответ на вопрос $query$ в тексте $passage$;

e — посчитанный моделью вектор, в котором значение на позиции i хранит вероятность того, что в этой позиции заканчивается ответ на вопрос $query$ в тексте $passage$;

Таким образом, если вероятности 2.9 начала и конца самого вероятного ответа в $passage$ достаточно малы (меньше подобранного порога), то разработанная система отвечает, что ответа нет, в противном случае использует выбранные начало и конец 2.10.

$$score_{has} = \max_{0 \le k \le l \le n} s_k + e_l \tag{2.9}$$

$$answer = \operatorname*{argmax}_{0 \le k \le l \le n} s_k + e_l \tag{2.10}$$

2.4.2 Retrospective Reader for Machine Reading Comprehension

Вдохновившись тем, что для ответа на вопрос по тексту люди сначала читают весь текст целиком, на основании прочитанного определяют положение возможного ответа, перечитывают и отвечают, авторы рассматриваемого подхода [12] попробовали реализовать нейросетевую систему, имитирующую подход человека на архитектурном уровне.

Основной вклад и успех данного подхода заключается в том, что авторы построили качественную систему валидации ответов, сам же поиск ответов осуществляется подобно тому, как это происходит в других системах.

Для реализации озвученной выше идеи, задачу разбили на два параллельных этапа:

а) sketchy reading - беглое, поверхностное чтение, которое изучает изложенный смысл и дает предварительное суждение правда ли в тексте есть ответ на вопрос;

б) intensive reading - интенсивное чтение, которое, как и беглое чтение проверяет наличие ответа на вопрос, на данном этапе дополнительно вычисляется положение ответа.

На рисунке 2.2 изображена архитектура нейронной сети, которую придумали авторы подхода Retrospective Reader для поиска и валидации ответов.

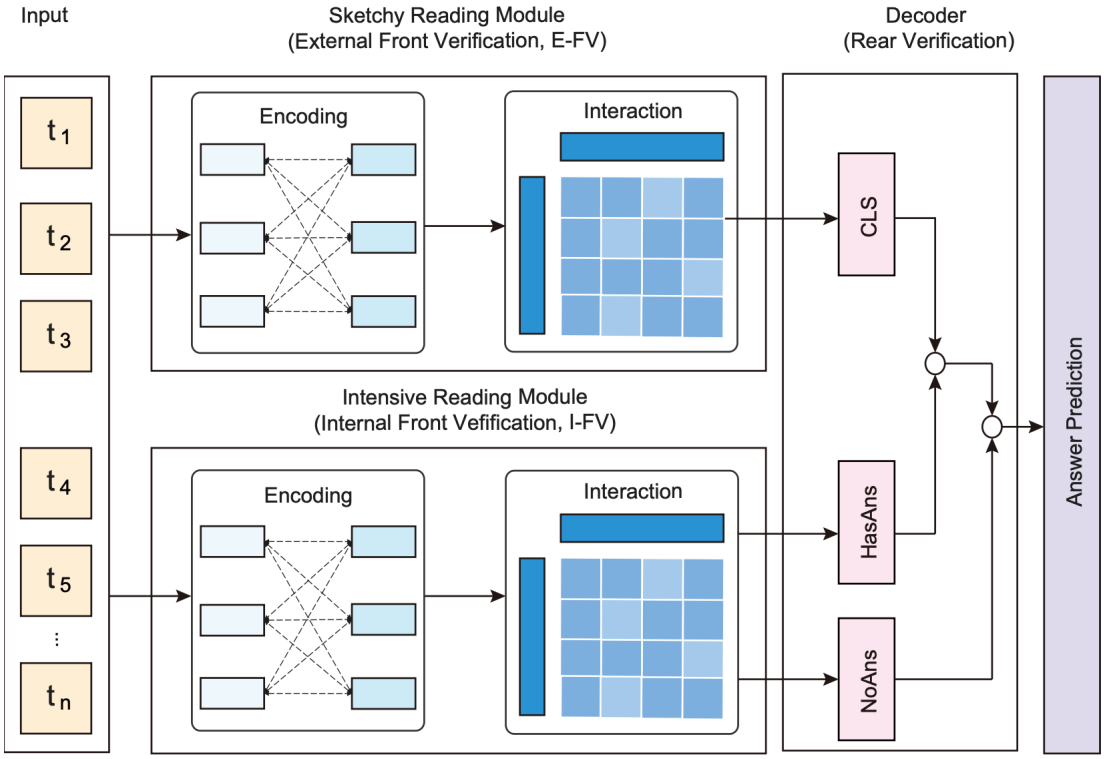

Рисунок 2.2 – Архитектура нейронной сети подхода Retrospective Reader

На вход модель принимает входной текст в виде последовательности токенов BPE и кодируется с помощью BERT архитектуры в представление входа $H = \{h_1, ..., h_n\}$, где n - это количество входных токенов t_i. Далее представление H подается на вход sketchy reading module и intensive reading module.

В модуле беглого чтения происходит так называемая верификация внешнего фронта(от англ. external front verification), для этого берется представление специального токена [CLS] и подается на вход полносвязного слоя нейронной сети, который оценивает правда ли в тексте имеется ответ, и в результате выдает число. При верификации внешнего фронта используется общепринятый подход: ожидается, что нейронная сеть научится собирать всю

необходимую информацию о тексте в представление специального токена [CLS].

В модуле интенсивного чтения сначала исходное представление H на основе эмбеддингов сегмента разбивается на представление вопроса H^Q и представление текста H^P. Дальше на основе механизма внимания рассчитывается новое представление текста $H^{'}$, которое учитывает вопрос. В формулах 2.11 и 2.12 представлено описание того, как рассчитывается представление текста с вниманием на запрос.

$$M = SoftMax(H(W_p H^Q + b_p \otimes e_q)^T) \tag{2.11}$$

$$H^{'} = MH^Q \tag{2.12}$$

где W_p, b_p — тренируемые во время обучения параметры;
e_q — вектор из единиц (нужен, чтобы применить смещение ко всей матрице).

Представление $H^{'}$ подается на вход двух полносвязных слоев: один для оценки начала ответа, второй для оценки конца ответа. Таким образом формируется оценка того, где находится ответ если он есть. В формуле 2.13 представлено описание того, как рассчитывается позиции начала и конца ответа:

$$p^s, p^e \propto Softmax(Linear(H^{'})) \tag{2.13}$$

где p^s, p^e — вектора «вероятности» того, что ответ начинается и заканчивается в каждом из токенов;
$Linear$ — полносвязный слой нейронной сети.

Для верификации в модуле интенсивного чтения используется такой же механизм, как и в модуле беглого чтения, только, обрабатывается представление токена не из H, а из $H^{'}$.

Результаты верификаций подаются на вход Rear verifiction стадии, которая учитывает вердикт внешнего и внутреннего фронта верификации, это происходит с помощью взвешенной суммы вердиктов по формуле 2.14:

$$v = \beta_1 \hat{y} + \beta_2 \bar{y} \tag{2.14}$$

где β_1, β_2 — гиперпараметры модели;
$\hat{y}$ — вердикт верификатора внешнего фронта;
$\bar{y}$ — вердикт верификатора внутреннего фронта.

На данном этапе нейронная модель заканчивается, а дальше используются ручные эвристики, чтобы получить итоговый вердикт. Для обучения такой архитектуры, для каждой из задач используется своя функция потерь, как для предсказания позиции, так и для задачи верификации - это перекрестная энтропия. Итоговая функция потерь определяется взвешенной суммой функций потерь подзадач и определена формулой 2.15:

$$L = \alpha_1 L^{span} + \alpha_2 L^{ans} \tag{2.15}$$

где α_1, α_2 — гиперпараметры модели;
L^{span} — функция потерь стадии определения позиции ответа;
L^{ans} — функция потерь Rear verifiction, т.е. валидатора наличия ответа в тексте.

Таким образом нейронная сеть учится сразу на две связанные между собой задачи. Для того, чтобы правильно интерпретировать масштаб результатов нейронной сети и определить правда ли в предсказанных значениях содержится ответ авторами был предложен эвристический подход, который взвешивает «уверенность» нейронной сети в оценке позиций начала и конца, результаты внешнего и внутреннего валидатора. Взвешенную сумму подают на вход пороговой функции, которая по предварительно подобранному порогу решает есть ли в тексте ответ на заданный вопрос. На момент написания данной магистерской диссертации, рассмотренный подход занимает лидирующие позиции и опережает показатели людей по выбранным метрикам на 4% $F1$ и 3% EM.

3 ПОСТРОЕНИЕ НЕЙРОСЕТЕВОЙ ВОПРОСНО-ОТВЕТНОЙ СИСТЕМЫ

Судя по изученным подходам, абсолютным лидером в задаче поиска ответа на вопрос является предварительно обученная компанией Google модель под названием BERT [11] и ее производные. Однако, у этой модели есть большой недостаток - она требует очень много вычислительных ресурсов как для обучения, так и для применения. Так, например, даже компания Google со всеми ее ресурсами еще не смогла внедрить модель (на момент написания работы) на всем потоке поисковых запросов, хотя с момента публикации статьи прошло уже два года. Выходит, что данная модель несет в себе огромную ценность для изучения, но может быть применена далеко не во всех практических задачах.

В данной работе было решено разработать более универсальное решение, которое можно было бы исполнять на порядки дешевле, при этом жертвуя относительно небольшой долей качества.

3.1 Архитектура нейронной сети

На рисунке 3.1 изображена архитектура разработанной в данной работе нейронной сети. Для выбора именно такой архитектуры был проведен ряд небольших экспериментов, например:

– Поиск решения для кодировки входных context и query. Пробовались RNN, GRU, LSTM.

– Поиск решения для связывания информации из context и query представлений. Пробовалась внимание токенов входного поля в это же поле, представление каждого токен контекста со всеми представлениями токенов запроса конкатенировались и подавались на вход dense сети, записывая в каждый токен контекста нужную информацию о запросе. Однако такой подход работал плохо, и в дальнейшем эволюционировал во внимание с учетом другого поля, это позволило сети формировать представление токена, учитывая его важность для ответа на запрос, с самых первых слоев.

– Поиск решения для формирования ответа нейронной сети. В этом пункте исследовались различные варианты, оказалось, что очень важно на уровне архитектуры подсказывать сети, в каком месте была выбрана граница начала ответа, иначе в ряде случаев может произойти совершенно неожиданное поведение, кроме того, сеть будет тратить дополнительные силы во время обучения впустую. Так же оказалось, что просто дать на вход dense слоя рас-

пределение левой границы ответа недостаточно, важно чтобы в зависимости от выбранной точки старта сеть обновила состояния предыдущих и последующих токенов.

Так как эти эксперименты не являются целью данной работы, а лишь подбирают минимальный необходимый уровень качества модели, то их результаты публиковаться не будут. Лучшая из исследуемых архитектур представлена на рисунке 3.1, рассмотрим ее подробнее.

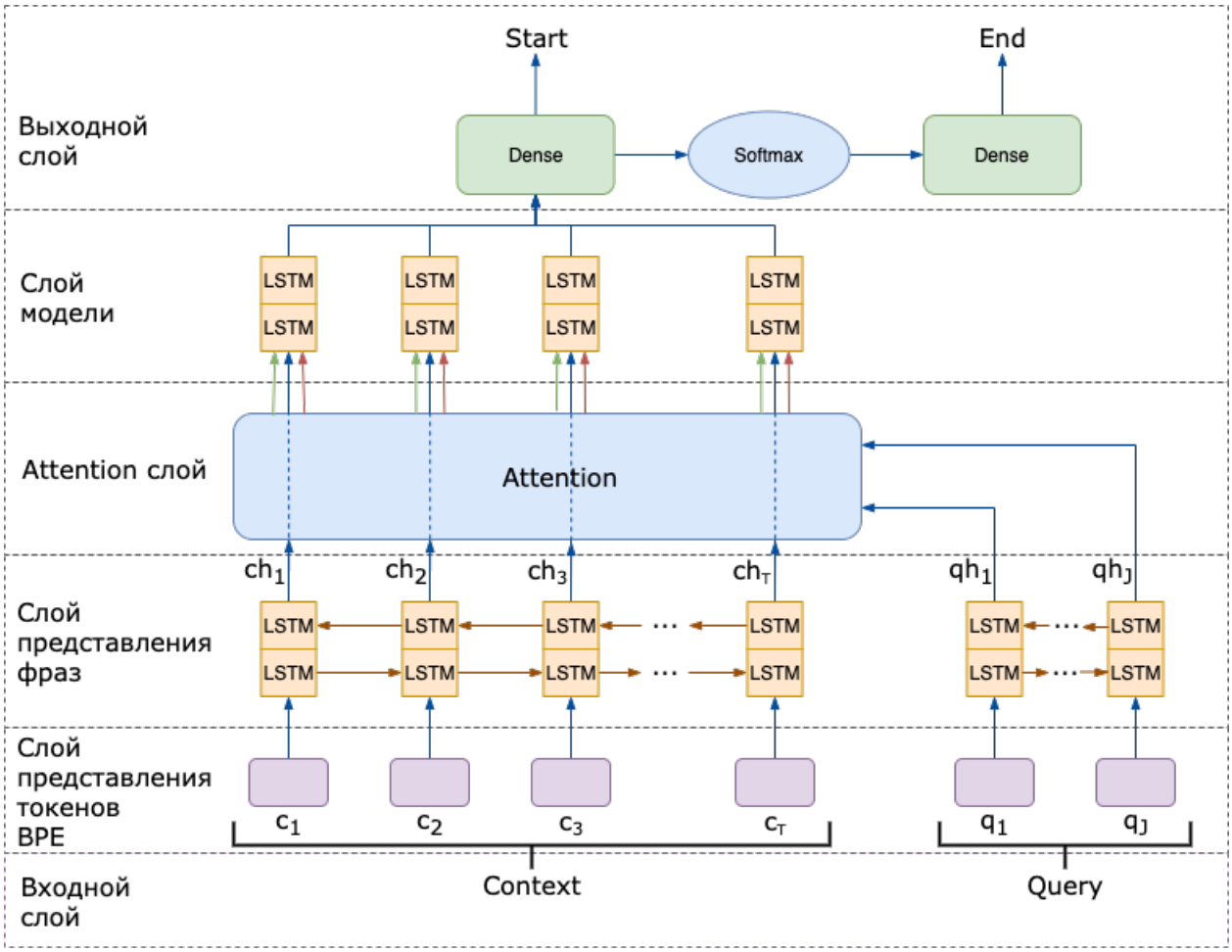

Рисунок 3.1 – Схематичное представление архитектуры нейронной сети

На вход сеть принимает два текстовых поля в виде целочисленных векторов, полученных в результате кодировке по bpe-словарю:

– context - текст документа, обозначим через c, где $|c| = T$

– query - вопрос по тексту, обозначим через q, где $|q| = J$.

После того, как в сеть поступили входные данные, они обрабатываются по схеме на визуальном представлении 3.1 снизу вверх. Рассмотрим подробнее, что происходит в каждом слое:

а) Слой представления токенов BPE. Для каждого токена выучивает вектор из e элементов, представляющий токен в пространстве модели. Для

исследуемой модели был подобран размер $e = 100$ как один из оптимальных поо соотношению скорость обучения / качество.

б) Слой представления фраз. С помощью Biderectional LSTM для каждого слова из контекста или вопроса составляет два представления: учитывающее все предыдущие и учитывающее все следующие слова последовательности, и конкатенирует их в представление qh_t токена запроса q_t или в представление ch_t токена контекста c_t.

в) Attention слой. На вход принимает все контекстные представления ch_t и запросные представления qh_j предыдущего слоя нейронной сети, и «связывает» информацию из context и query частей модели. В основе проводимых в слое вычислений лежит матрица похожести токенов контекста и вопроса $S \in \mathbb{R}^{T \times J}$, где элемент $S_{i,j}$ соответствует похожести i-го токена контекста j-ому токену вопроса, и рассчитывается по формуле 3.1:

$$S_{i,j} = W_{(S)}[ch_i; qh_j; ch_i \circ qh_j] \tag{3.1}$$

где $W_{(S)}$ — обучаемый вектор весов;
ch_t — t-ое контекстное представление из слоя представления фраз;
qh_j — j-ое вопросное представление из слоя представления фраз;
[;] — конкатенация векторов по строкам;
$\circ$ — поэлементное умножение векторов.

На основе матрицы S считаются контекстно-запросный и запросно-контекстный attention:

– Контекстно-запросный attention для каждого токена контекста ch_t взвешивает все токены вопроса qh_t в вектор внимания a_t по формуле 3.2, и на его основе собирает наиболее полезную информацию из запросной части для обработки токена ch_t по формуле 3.3:

$$a_t = softmax(S_{t:}) \tag{3.2}$$

$$\hat{cq}_{:t} = \sum_j a_{tj} ch_{:j} \tag{3.3}$$

На рисунке 3.2 изображена схема расчета контекстно-запросного attention с последующей агрегацией запросных представлений.

– Запросно-контекстный attention рассчитывает какие слова контекста важны для ответа на вопрос. Для этого для каждого токена вопроса на позиции j находится наиболее важное для него токен из контекста на позиции t по матрице S. В результате применения $Softmax$ к похожестям получается век-

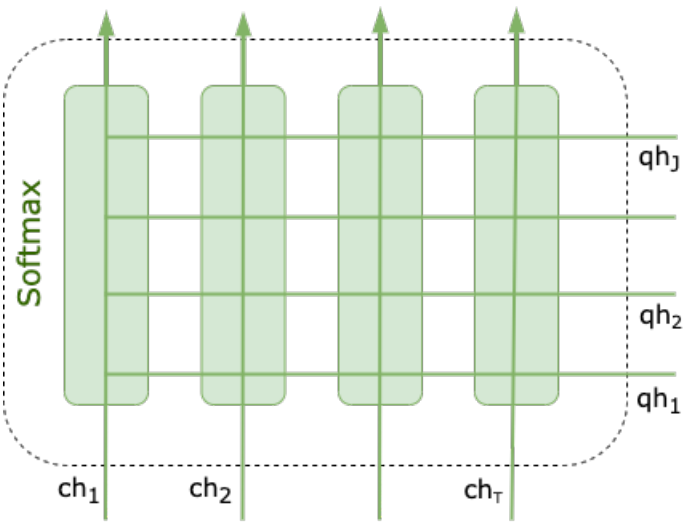

Рисунок 3.2 – Схема расчета контекстно-запросного attention

тор внимания b (см. формулу 3.4), на основе которого собирается наиболее полезная информация о контексте для данной запросной части (см. формулу 3.5):

$$b = softmax(\max_{col} S) \tag{3.4}$$

$$\hat{ch} = \sum_{j} a_{tj} ch_{:j} \tag{3.5}$$

На рисунке 3.3 изображена схема расчета запросно-контекстного attention с последующей агрегацией контекстных представлений.

г) Слой кодировки контекста с учетом вопроса. Кодирует каждое слово документа с учетом вопроса при помощи LSTM. Именно учетом вопроса при кодировании и отличается от слоя представления фраз, который в первую очередь определял отношение между словами в тексте. На этом уровне ожидается, что сеть уже понимает какие токены документа связаны с вопросом, определяет вероятные положения и размеры ответов, кодирует эту информацию. Для каждого токена на позиции t на вход слой принимает конкатенацию $[ch_{:t}, \hat{ch}_{:t}, \hat{qh}_{:t}]$.

д) Выходной уровень состоит из двух последовательных слоев: слой оценки позиции левой границы ответа в контексте, и слой оценки правой границы ответа в контексте. Оценка позиции начала ответа производится полносвязным слоем по конкатенации всех представлений предыдущего слоя, ожидается, что данный слой просто раскодирует информацию о начале ответа. После того, как вектор вероятностей начала ответа был найден, он и

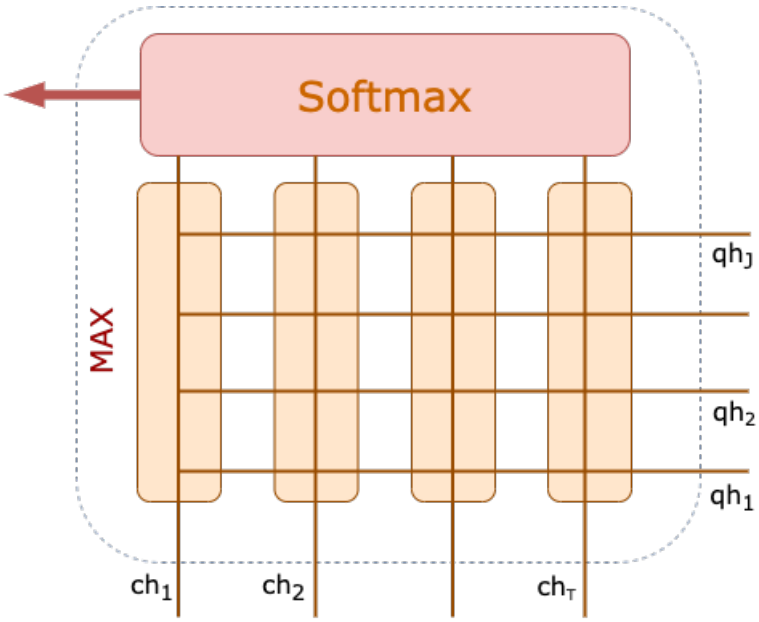

Рисунок 3.3 – Схема расчета запросно-контекстного attention

закодированная данные о положении ответа передаются в слой оценки позиции конца ответа в контексте. Данный слой с помощью LSTM дополняет закодированные данные прошлого уровня информацией о выбранном начале ответа и сразу же декодирует из нее вероятность конца ответа в каждом токене контекста. Левая граница ответа выбирается как $argmax$ первого слоя выходного уровня, а правая граница как $argmax$ второго соответственно.

В результате подбора гиперпараметров для описанной выше архитектуры нейронной сети, лучший достигнутый результат составляет 0.4121 EM и 0.4482 $F1$.

3.2 Обучение представлений текстовых данных

Скорее всего в предыдущем подходе был получен посредственный результат потому, что 100 тыс. пар контекст/вопрос, где на каждый контекст приходится несколько вопросов, - это очень маленькое количество текста для того, чтобы нейронная сеть хорошо понимала используемый язык. Это предположение дает основание провести эксперимент с предварительным обучением первого слоя (представления токенов) нейронной сети на других данных.

Исходные данные представляют собой вопросы по статьям википедии, поэтому для предварительного обучения векторов представлений были использованы статьи википедии, это позволит модели лучше учитывать распределение текста. Таким образом для предварительного обучения были исполь-

зован сэмпл из 5 млн. статей последнего дампа англоязычной википедии [13]. Словарь для обучения представлений токенов D был составлен алгоритмом BPE и использовал все 5 млн. статей.

Для обучения представлений использовалась техника CBOW [14]. Она заключается в том, что ставится следующая задача: для контекста C рассчитать вероятность того, что каждое возможное слово w используемого словаря встретится с этим контекстом. Наглядное изображение задачи представлено на рисунке 3.4:

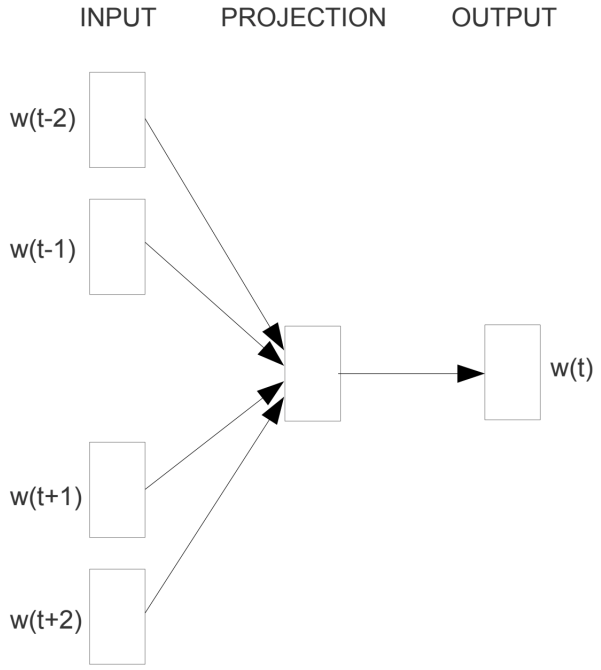

Рисунок 3.4 – Схема обучения текстовых представлений подхода CBOW

Формально: пусть имеется последовательность текстов $T = \{t_1,...,t_n\}$, а каждый текст представлен последовательность токенов $t_i = \{w_1, ..., w_k\}$. Тогда для текста t_i и ширины окна C задача состоит в том, чтобы для каждой подпоследовательности слов $W = \{w_{j-C/2}, ..., w_j, ..., w_{j+C/2}\}$,
$W \in t_i$ по входу $I = W \setminus \{w_j\}$ максимизировать вероятность пропуска слова w_j в контексте I.

Таким образом для решения задачи получаем очень простую нейронную сеть, которая имеет архитектуру представленную на рисунке 3.4:

– Слой представления слов. Для каждого из слов контекста по индексу берется обученный вектор представления.

– Слой представления контекста. Представления всех слов складываются, тем самым формируя эмбеддинг контекста.

– Слой предсказания. Представляет из себя dense слой, который на вход принимает представление контекста и рассчитывает вектор размера $\|D\|$,

где j-ый элемент хранит в себе некоторую меру того, что слово $w_j \in D$ было пропущено в контексте I.

Для обучения такой нейронной сети использовалась Softmax Cross Entropy функция потерь:

$$L(w_j, I) = -log(p_j) - \sum_{k!=j} log(1 - p_k)$$

где $L(w_j, I)$ — ошибка нейронной сети на токене w_j при контексте I;
p_i — значение i-го элемента softmax преобразования выхода нейронной сети $p_i = softmax(logits)_i$.

Обучив описанную выше нейронную сеть на дампе википедии, можно предположить, что полученные вектора представлений токенов словаря лучше, чем в предыдущем подходе потому, что нейронная сеть увидела гораздо больше данных, а также решала более простую задачу, в которой преуспела, а значит скорее всего научилась понимать текст. Для того, чтобы переиспользовать эти эмбеддинги токенов достаточно извлечь их из полученной нейронной сети и при инициализации представлений токенов другой интересующей нейронной сети использовать не псевдослучайные числа, а полученные вектора.

В данной работе были обучены вектора представлений размера 300, этот размер оказался оптимальным с точки зрения времени обучений и качества. Кроме того, размер эмбеддингов 300 позволит корректно сравнивать этот метод с некоторыми следующими.

В результате «теплого старта» исследуемой архитектуры нейронной сети удалось достичь качества 0.5998 EM и 0.6277 $F1$.

3.3 Использование предварительно обученных представлений текстовых данных

3.3.1 FastText

Полученный результат оказался в полтора раза лучше, чем обучение с нуля, но возможно можно было бы получить лучше, если:

– использовать большее количество данных для предварительного обучения эмбеддингов;

– потратить большее количество усилий и ресурсов на подбор гиперпараметров;

– подобрать более подходящую архитектуру для предварительного обучения;

– использовать другой вид токенизации.

По этим пунктам уже есть достаточное количество научных работ, чтобы не изобретать качественные статические эмбеддинги еще один раз, в этом эксперименте проверяется возможность использования одной из последних и лучших работ на эту тему.

FastText [15] - это библиотека, разработанная компанией Facebook, для эффективного обучения представлений слов и предложений. Подход в библиотеке идейно очень схож с тем, что использовалось в 3.2, однако имеет свои особенности, рассмотрим их:

В FastText вместо CBOW используется Skip-gram подход [14]. Суть Skip-gram подхода в том, что нейронную сеть учат предсказывать контекст для конкретного слова, т.е. подход является полной противоположностью CBOW. На рисунке 3.5 изображена схема работы подхода skip-gram.

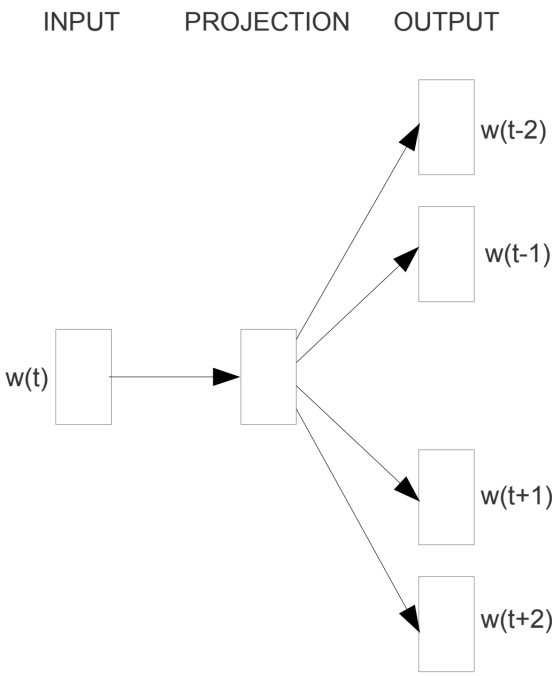

Рисунок 3.5 – Схема обучения текстовых представлений подхода Skip-gram

Как и в подходе CBOW все документы $t_i \in T$ нарезаются на непрерывные подпоследовательности токенов $\{w_{j-C/2}, ..., w_j, ..., w_{j+C/2}\}$, только теперь вход формируется на основе j-го слова $I = w_j$, а ожидаемый выход - на основе конкатенации one-hot encoded векторов по словам контекста $O = \{w_{j-C/2}, ..., w_{j-1}, w_{j+1}, ..., w_{j+C/2}\}$. Обучение такой нейронной сети происходит точно также, но при этом имеет свои преимущества и недостатки.

Недостатки:

– Поставленная задача является более сложной: контринтуитивно как по одному слову предсказать контекст.

– По архитектурным причинам одна шаг обучения такой сети требует большее количество вычислений.

Преимущества:

– Нет необходимости суммировать, а значит не теряется часть сигнала и потенциально модель может быть более хорошей.

– Так как чем ближе слово контекста к входному, тем больше они на друг друга влияют, то подход позволяет реализовать эту зависимость на уровне архитектуры. При суммировании функции потерь каждого слова контекста предлагается взвешивать значение обратно пропорционально расстоянию до входного слова. Эта техника позволяет обучать нейронную сеть эффективнее, так как ей не требуется в процессе самой выявлять эту зависимость.

По результатам исходной работы [14], где были предложены CBOW и Skip-gram, оба подхода конкурируют друг с другом на разных задачах, и каждый из них где-то лучше другого.

Второе важное отличие FastText от представлений, обученных в 3.2, состоит в способе токенизации. В данном методе каждое слово представлялось мешком из буквенных n-грам, перед токенизацией каждому слову в начало и конец дописывали символы «"и » что позволяет отличать n-граму являющуюся префиксом или суффиксом от других n-грам. Кроме того, и само слово включалось в мешок из n-грам. Например, при $n = 3$ слово «привет» разбивается на следующие токены:

«<пр», «при», «рив», «иве», «вет», «ет>», «<привет>»

Таким образом, используя n-грамы в подходе библиотеки FastText борются с редкими словами, и даже для тех слов, которых нет в словаре скорее всего будет составлено представление с помощью суммирования эмбеддингов по знакомым n-грамам.

Помимо теоретической части подхода компания Facebook предоставила и предварительно обученные их командой вектора представлений для разных языков [16]. Для каждого языка в качестве исходных данных использовалось поддерево википедии соответствующей локали.

Для проведения текущего эксперимента были использованы предварительно обученные FastText вектора на английских данных, которые позволили реализовать «теплый» старт обучения исходной нейронной модели. В результате ее обучения удалось достичь качества 0.6241 EM и 0.6615 $F1$.

3.3.2 BERT-Base

Во всех предыдущих экспериментах использовались так называемые статические эмбеддинги. Следующей ступенькой эволюции текстовых представлений являются контекстуальные представления, не зря лидирующие 50+ позиций исследуемой задачи занимают модели построенные с использованием именно такой техники.

Рассмотри эти подходы детальнее:

– Статические эмбеддинги - подход представления слов, основанный на ассоциативном массиве, в котором каждому токену ставится в соответствие некоторый вектор, представляющий его в используемом нейронной сетью пространстве. Данный метод очень чувствителен к омонимии, так как для каждого слова используется лишь один вектор, а значения у омонимов зачастую ортогональные. Но в тоже время метод является очень быстрым, потому что для отображения слова в d-мерное пространство достаточно найти его в ассоциативном массиве, который скорее всего вмещается в оперативную память компьютера.

– Контекстные эмбеддинги - подход представления слов, основанный на расчете эмбеддинга слова по его контексту. Т.е. для представления слова в d-мерном пространстве используется специальная нейронная сеть, которая на основе зависимости слова и его окружения формирует результирующий вектор. При качественном обучении данный подход успешно решает проблему омонимии, однако обладает недостатками. Первый недостаток таких представлений в том, что при использовании на коротких фразах или отдельных словах представления могут не обладать теми же свойствами, как и полноценные контекстные эмбеддинги. Второй, самый важный, недостаток заключается в применении таких эмбеддингов, если в статическом подходе почти всегда можно за константное время найти вектор в ассоциативном массиве, то в данном случае нужно применять нейронную сеть, которая скорее всего будет довольно тяжелой, так как для формирования успешного представления слова требуется решить много сложных задач: разметить последовательность по частям речи, произвести синтаксический парсинг входной последовательности, выучить языковую модель и т.д. Таким образом для получения хороших результатов в понимании текста компьютером техника контекстным эмбеддингов жертвует очень много вычислительных ресурсов.

Один из лучших алгоритмов контекстного представления - Transformer [17]. В основании алгоритма лежит механизм внимания в самого себя (от англ. self-attention). На рисунке 3.6 представлена схема работы механизма внима-

ния в себя на примере кодирования последовательности токенов, разберем ее подробнее.

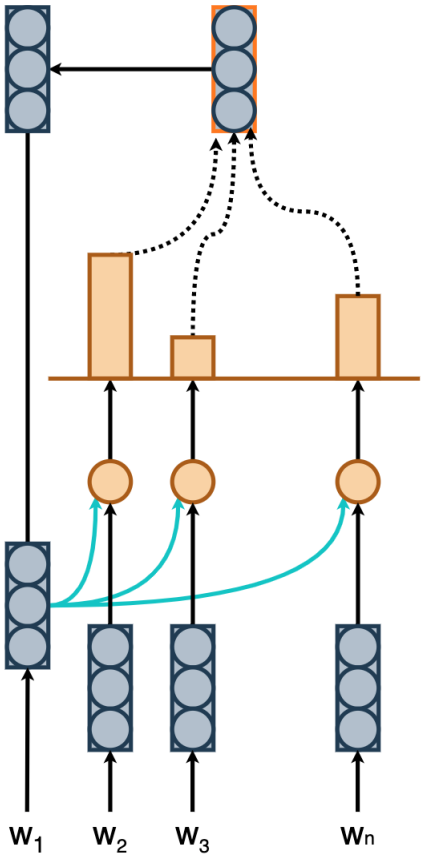

Рисунок 3.6 – Схема работы механизма внимания в себя на примере кодирования последовательности токенов

Пусть $\{w_1, w_2, ..., w_n\}$ входная последовательность токенов. Тогда применение механизма внимания в себя производится для каждого токена w_j. Основная цель операции - обновить представление каждого токена так, чтобы оно лучше учитывало контекст исходной последовательности. Расчет новых значений производится с помощью следующих операций:

а) вектор токена w_j скалярно перемножается с векторами всех других токенов последовательности;

б) результаты перемножения конкатенируются в вектор и масштабируются;

в) отмасштабированный вектор проходит через softmax преобразование, в результате хранится степень внимания к тому или иному токену контекста;

г) вектор внимания перемножается с матрицей представления исходной последовательности, в результате чего получается новый вектор представления токена w_j.

В формуле 3.6 представлено описание операции SelfAttention.

$$SelfAttention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.6}$$

где Q — обновляемое представление;
K, V — одинаковые последовательности представлений, соответствующие токенам контекста;
d_k — размерность векторов Q, K, V.

По сравнению с архитектурой RNN, SelfAttention теряет очень важную информацию - позицию исходных токенов. В RNN сигнал о позиции передается на архитектурном уровне, потому что токены подаются на вход сети последовательно, в SelfAttention они могут подаваться в произвольном порядке, и если на вход поступит последовательность с двумя одинаковыми токенами, то сеть не сможет понять, чем они отличаются, а если будет два подлежащих и сказуемых, то сети будет сложно установить правильные зависимости подлежащих и сказуемых. Для решения этой проблемы к исходному статическому представлению каждого токена добавляют представление позиции этого токена той же размерности. Вектор представления позиции формируют с помощью тригонометрических функций с разными периодами так, чтобы каждую позицию входной последовательности можно было однозначно определить. Суммирование эмбеддингов токенов и эмбеддингов позиций позволяет не потерять важную информацию, как это могло бы происходить при использовании других агрегирующих функций.

В архитектуре Transformer для получения максимальной эффективности механизм внимания используется многократно и параллельно. Идея заключается в том, чтобы рассмотреть контекст сразу в нескольких параллельных проекциях, в каждой из проекций построить независимое обновление представления, ксоответственно онкатенировать обновленные представления каждой проекции, и получить разностороннее представление для токена на позиции j. Такое использование внимания на себя получило название Multi-Head Attention. Считается, что благодаря данной технике, в каждой из голов внимания в себя нейронная сеть способна решать разные задачи и соответственно извлекать разные сигналы, например, частеречный разбор, синтаксический разбор, лексический разбор, пунктуация и т.д. Схема работы Multi-Head Attention изображена на рисунке 3.7:

В формулах 3.7 и 3.8 представлено формальное описание применения

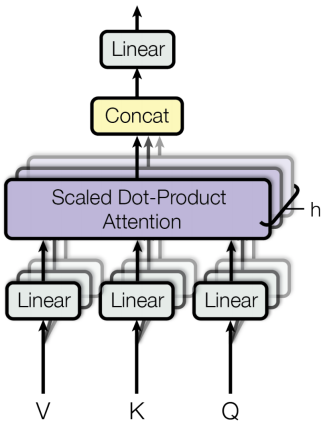

Рисунок 3.7 – Схема работы Multi-Head Attention

Multi-Head Attention:

$$MultiHead(Q, K, V) = [head_1; head_2; ...; head_h]W^O \tag{3.7}$$

$$head_i = SelfAttention(QW_i^Q, KW_i^K, VW_I^V) \tag{3.8}$$

где Q — обновляемое представление;
K, V — одинаковые последовательности представлений, соответствующие токенам контекста;
$[;]$ — операция конкатенации;
h — гиперпараметр количества параллельных проекций;
W_i^Q, W_i^K, W_i^V, W^O — обучаемые матрицы проекций.

После конкатенации эмбеддингов всех голов результат нормализируется и направляется в полносвязный слой. Этот слой позволяет смешать информацию из разных голов внимания, что помогает извлекать сигнал еще лучше. Например если учесть частеречную разметку, то провести синтаксический разбор становится легче. Для того, чтобы усовершенствовать представление каждого токена на основании информации, полученной в параллельных головах, трансформер применяют несколько раз, т.е. после первого слоя трансформера сразу же применяется второй, третий и т.д., количество используемых слоев выбирают исходя из решаемой задачи и имеющихся ресурсов. Полная схема архитектуры представлена на рисунке 3.8

После исследования архитектуры нейронных сетей типа трансформер, в компании Google сформировали несколько задач обучения, наподобие тех, что использовались в FastText, обучили несколько тяжелых языковых моделей на основе последовательности трансформеров, и выложили их в откры-

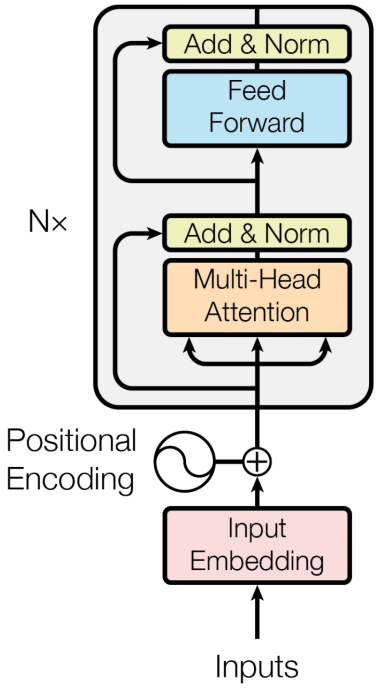

Рисунок 3.8 – Схема работы Transformer

тый доступ. Данный подход, на момент выполнения работы является самым сильным и называется BERT - Biderectional Encoder Representations from Transformer [11].

Важность предварительно обученных моделей BERT не только в том, что они хорошо решают поставленные задачи, но и в том, что их можно адаптировать почти под любую задачу для работы с естественным языком. Более того, из-за того, что в прикладных задачах зачастую очень мало данных, подход с использованием предварительно обученной моделью очень часто оказывается лучшим. В результате адаптации предварительно обученной языковой модели BERT, в данной работе была разработана архитектура вопросно-ответной модели, ее схема представлена на рисунке 3.9.

Для того, чтобы разделять токены вопросов и текстов, представление каждого токена формируется суммированием не только эмбеддинга токена и позиции, но и эмбеддинга сегмента, который обозначает источник токена: вопрос или текст, и вычисляется по такому же принципу, как и эмбеддинг позиции. После применения предварительно обученной BERT модели к обучающему примеру, для каждого входного токена формируется контекстное представление. Каждое контекстное представление поступает на вход двум полносвязным слоям: первый полносвязный слой «start dense» оценивает правдоподобие того, что в этом токене начинается ответ, а второй полносвязный слой «end dense» оценивает правдоподоие конца ответа в этом то-

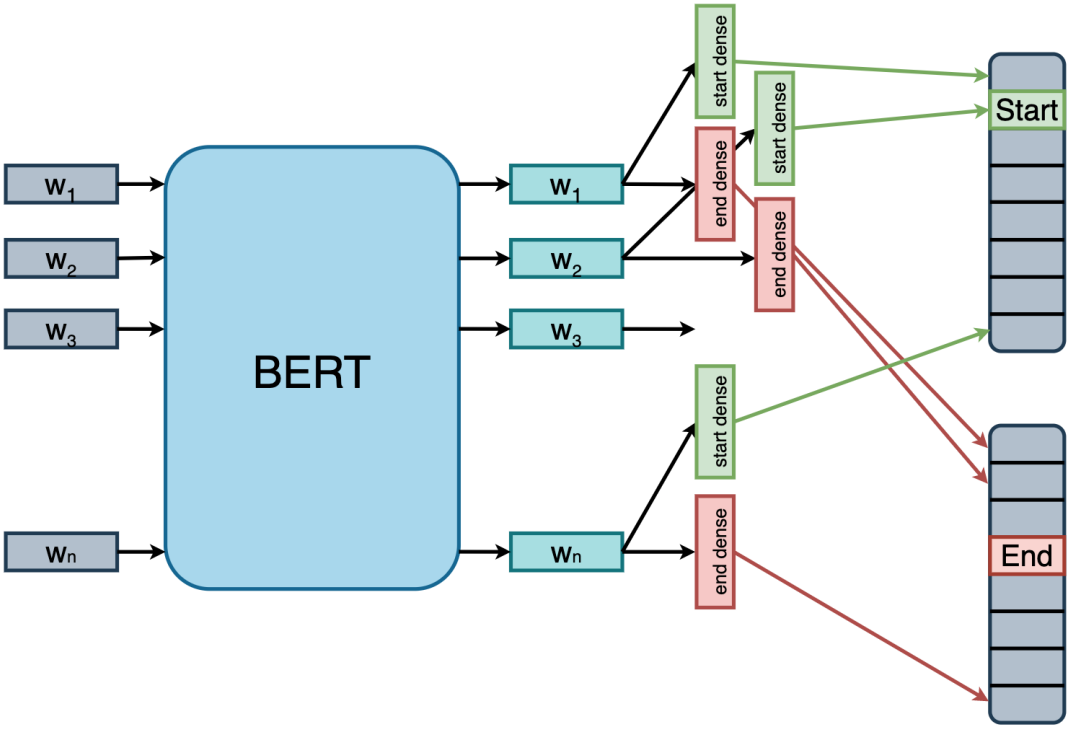

Рисунок 3.9 – Архитектура вопросно-ответной модели на основе BERT

кене. Правдоподобия начала ответа всех токенов конкатенируется в вектор и проходят softmax преобразование, тоже самое происходит и с правдоподобиями конца ответа, таким образом формируется предположение нейронной сети о том, где находится ответ. Для решения поставленной задачи, берется исходная предварительно обученная BERT-Base модель, ее архитектура модифицируется так, чтобы соответствовать описанию выше и максимально переиспользовать уже существующие, обученные веса, в итоге слои «start dense» и «end dense» обучаются с нуля.

В результате использований контекстных представлений BERT-Base и дообучения предложенной выше архитектуры удалось достичь качества 0.7085 EM и 0.7509 $F1$.

3.3.3 FastText на SberQuAD

Во всех предыдущих экспериментах исследовалось построение вопросно-ответной системы в терминах решения задачи SQuAD 2.0, рассмотренной в 2.1.5, однако под конец выполнения данной магистерской работы был опубликован русский QA датасет SberQuAD 2.1.6. Несмотря на схожесть этих датасетов, построение русскоязычной вопросно-ответной системы представляет научный интерес, потому что:

– Даже при относительно хороших результатах на английском языке,

точно такие же решение не факт, что будут успешно работать при переходе на русский язык. Построив похожее решение для русского языка с одной стороны можно понять то, насколько сильно подход является языкоспецифичным, а с другой стороны, есть вероятность неплохое решение для дополнительного языка.

– Область построения вопросно-ответных систем на русском языке почти не изучена, открытых исследований и решений мало.

– На ряду с английским и китайскими языками русский - один из самых распространенных в мире.

Данных в русскоязычном датасете в два раза меньше англоязычного, кроме того, качественно предварительно обученная BERT-Base модель была опубликована только для английского языка, поэтому для проведения эксперимента с русскоязычной моделью использовался лучший подход на статических предварительно обученных эмбеддингах FastText, разработанный в 3.3.1. Благодаря скорости обучения, вектора представлений FastText были предварительно обучены исследователями компании Facebook на 294 языках мира, в том числе и на русском.

Так как задачи SQuAD 2.0 и SberQuAD немного отличаются, а модель была разработана именно для того, чтобы искать ответ в виде непрерывной подпоследовательности символов в тексте, то для реализации данного эксперимента данные были предварительно обработаны так, чтобы удовлетворять этим условиям.

В результате исследования выше обозначенной модели на валидационной выборке SberQuAD удалось достичь качества 0.5197 $F1$ и 0.7253 EM.

В таблице 3.1 представлены полученные результаты в сравнении с двумя предложенными baseline. Дополнительно были добавлены результаты BERT решения, которое так же было предложено в работе по формированию задачи SberQuAD. Подход исследованный в данной магистерской работе обозначен с помощью имени «BiLSTM QA Attention + FastText pretrained embeddings»

Подход	EM	F1
Simple baseline	0.3	25
ML baseline	3.7	31.5
BiLSTM QA Attention + FastText pretrained embeddings	51.97	72.53
BERT-Base	66.6	84.8

Таблица 3.1 – Результаты исследования русскоязычной вопросно-ответной системы на данных SberQuAD в сравнении с предложенными baseline решениями

По результатам видно, что разработанное решение существенно превосходит предложенные опорные решения, однако как и в задаче SQuAD проигрывает BERT-Base подходам. В таблице 3.2 представлены результаты исследуемой архитектуры на задачах SQuAD 2.0 и SberQuAD.

Задача	EM	F1
SQuAD 2.0	62.41	66.15
SberQuAD	51.97	72.53

Таблица 3.2 – Сравнение исследуемой архитектуры нейронной модели на задачах SQuAD 2.0 и SberQuAD

При сравнении решения на разных задачах видно, что на русском языке модель существенно лучше по $F1$ метрике, однако достоверно так утверждать нельзя. Предположительно, такая разница в $F1$ связана с тем, что в русскоязычных данных на все вопросы существует ответ, в то время как в англоязычных более чем для 30% вопросов ответа не существует, это различие очень сильно упрощает задачу: если на вопрос гарантированно есть ответ, то это позволяет модели находить что-то отдаленно похожее на вопрос в тексте в случаях, когда она не знает ответ, тем самым накручивая метрику $F1$. С другой стороны видно, что на русских данных модель сильно проигрывает в $Exact\ Match$ метрике, скорее всего это связанно с относительной свежестью русского набора данных, в то время как в англоязычном наборе каждый ответ валидировался 5-ью независимыми людьми, в русском наборе данных такого не происходило. Соответственно по конкретному вопросу из русскоязычного датасета ответ является более смещенными в точку зрения человека, который придумал этот вопрос, что вносит шум в предсказания нейронной сети и оценку ее качества по полному совпадению.

В таблице 3.3 представлен выигрыш в результате метрик при переходе от лучшей исследованной модели статических представлений, к контекстным моделям, основанным на предварительно обученных BERT языковых моделях по двум языкам: задача SQuAD для английского и задача SberQuAD для русского.

Задача	EM	F1
SQuAD 2.0	+8.44%	+8.96%
SberQuAD	+14.63%	+12.27%

Таблица 3.3 – Сравнение улучшения от перехода к использованию контекстных представлений на задачах SQuAD 2.0 и SberQuAD

Видно, что для русского языка выигрыш от использования контекстных представлений гораздо больше, чем для английского. С чем это связано не известно, возможно дело в особенности языка, например, обилие различных форм одного и того же слова в русском языке может затруднять понимание модели статических представлений, в то время как в контекстной модели на уровне архитектуры заложено более эффективное использование разных источников информации (пунктуацию, синтаксис, и др.), что позволяет ей решать поставленную задачу лучше.

В результате проведенного эксперимента по задаче SberQuAD была построена вопросно-ответная система хорошего качества, и проведен сравнительный анализ исследуемых архитектур на русском и английском языках.

3.4 Анализ результатов экспериментов

В данном разделе были описаны 4 проведенных экспериментов на SQuAD 2.0 датасете. В текущем подразделе будут совместно рассмотрены их результаты Для анализа и подведения итогов по экспериментам обозначим их именами. Пусть:

– модель с холодным стартом в 3.1 несет название «BiLSTM QA Attention»;

– модель с теплым стартом из предварительно обученных текстовых представлений в 3.2 несет название «BiLSTM QA Attention + Wiki pretrained embeddings»;

– модель с теплым стартом из предварительно обученных эмбеддингов сторонней библиотеки FastText в 3.3.1 несет название «BiLSTM QA Attention + FastText pretrained embeddings»;

– однослойная модель с использованием предварительно обученных эмбеддингов BERT-Base в 3.3.2 несет название «BERT-Base».

Тогда в таблице 3.4 представлены лучшие достигнутые результаты по исследованным подходам построения вопросно-ответной системы.

Подход	EM	F1
BiLSTM QA Attention	41.21	44.82
BiLSTM QA Attention + Wiki pretrained embeddings	59.98	62.77
BiLSTM QA Attention + FastText pretrained embeddings	62.41	66.15
BERT-Base	70.85	75.09

Таблица 3.4 – Лучшие результаты исследованных подходов построения вопросно-ответной системы

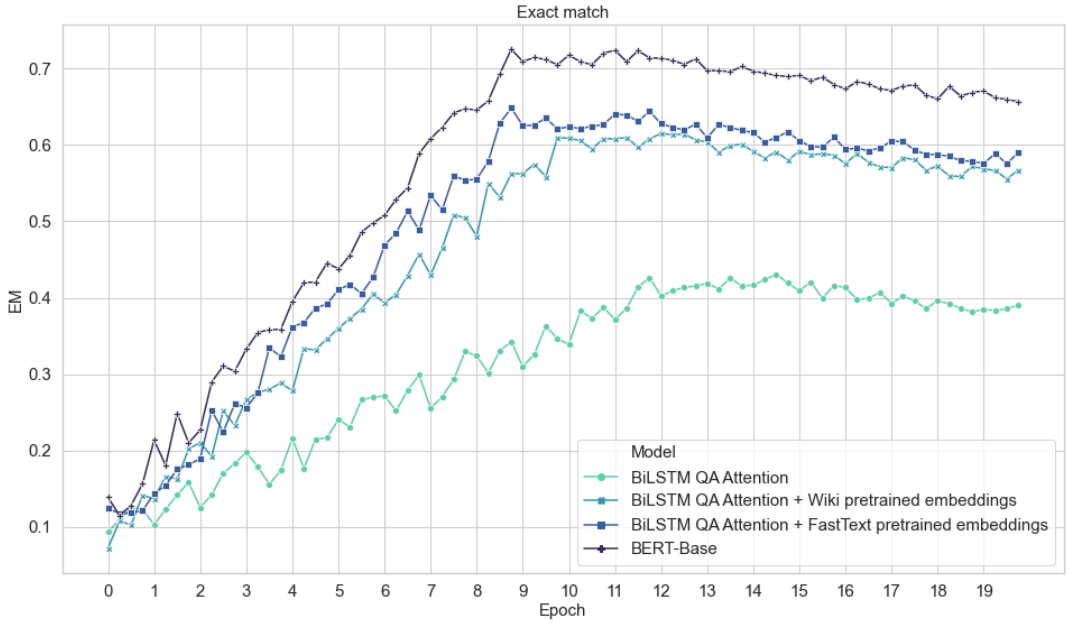

Рисунок 3.10 – Кривая изменения метрики Exact Match во время обучения на валидационной выборке

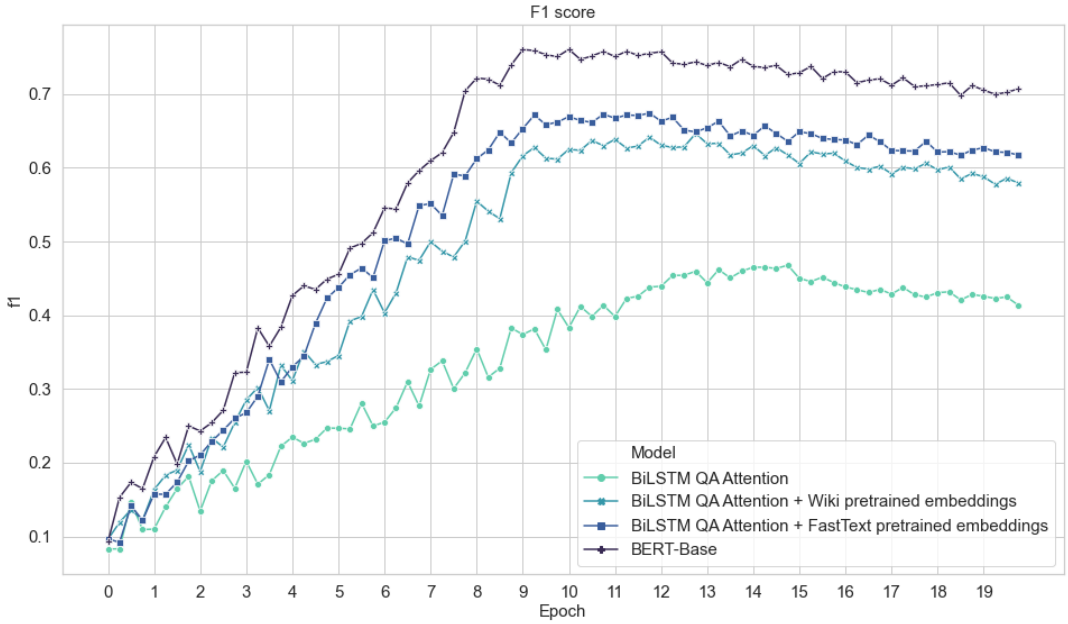

Рисунок 3.11 – Кривая изменения метрики F1 во время обучения на валидационной выборке

А на рисунках 3.10 и 3.11 представлены кривые метрик Exact Match и F1 на отложенной валидационной выборке в процессе обучения.

По представленным выше результатам видно, что после 10-11 эпох модели во всех подходах начинают переобучаться, это говорит о том, что каждый из исследуемых подходов был обучен в полной мере, и сравнение производится корректно: по полностью обученным моделям.

Данные визуализации демонстрируют превосходство качества контекстных методов представления, при решении задачи поиска ответа на вопрос, почти на любом шаге обучения.

4 ДИСТИЛЛЯЦИЯ ЗНАНИЙ «ТЯЖЕЛЫХ» НЕЙРОННЫХ СЕТЕЙ

В разделе 3 было показано, что подходы, основанные на динамических представлениях текста, существенно выигрывают у статических подходов по качеству. Однако, формирование контекстного представления осуществляется с помощью большой нейронной сети, что негативно влияет на производительность системы, и в большинстве случаев возможность использования таких подходов для решения прикладных задач исключена из-за дороговизны.

В такой ситуации сразу возникает вопрос: а можно ли как-нибудь сжать тяжелую модель, потеряв минимум качества? Так сделать можно, более того, подходы к сжатию моделей машинного обучения появились даже раньше нейронных сетей. До скачка развития в области нейронных сетей зачастую выходило так, что наилучший способ решения поставленной задачи - это ансамбль классических моделей машинного обучения, и уже тогда люди интересовались как сжать этот ансамбль, чтобы снизить стоимость модели. Понятно, что большинство подходов завязано на конкретный тип моделей и не может быть переиспользовано в других условиях, но некоторые из них были достаточно хорошо изучены, доработаны, и сейчас успешно используются в разработке нейросетевых решений.

Здесь и далее под сжатием подразумевается некоторая операция над моделью, в результате которой получается другая модель, решающая такую же задачу, но требующая меньшее количество вычислительных ресурсов. Так как узким местом в текущей системе является время, затраченное процессором или видеокартой на обработку задачи, то уменьшение потребления именно этих ресурсов считается успешным сжатием, возможно даже в ущерб другим.

В этом разделе будет исследована возможность сжать разработанную в 3.3.2 BERT-Base нейронную сеть в менее ресурсоемкую архитектуру BiLSTM QA Attention, разработанную в 3.1. В 4.1 рассмотрены основные подходы сжатия и мотивация использования дистилляции знаний для решения поставленной задачи, в 4.2 рассмотрена теоретическая часть дистилляции, в 4.3 описан исследованный в работе подход дистилляции, а в 4.4 проведен анализ полученных результатов.

4.1 Подходы сжатия нейронных сетей

Почти всегда сжатие модели машинного обучения - это процесс обреченный на успех, с другой же стороны методов сжатия очень много, и важно понимать в каком направлении исследовать, чтобы потенциально была возможность достигнуть нужного уровня сжатия и качества.

Для сжатия нейронных сетей выделяют три основных подхода:

а) Квантование (от англ. quantization). Идея подхода состоит в том, чтобы избавиться от избыточности выученной нейронной сетью информации, это достигается засчет уменьшения битовой ширины используемых чисел. Например, вместо 64-битного числа с плавающей точкой используют 16-битное. Ускорение происходит засчет использования SIMD операций. При обучении хранят сеть с полной точностью, перед применением над очередным примером веса сети квантуют в предварительно рассчитанные числа меньшей точности, применяют квантованную сеть, вычисляют ее ошибку и градиент, но обновление проводят для весов полной точности. Так в [18] было показано, что можно достаточно успешно использовать квантование в $\{-1, +1\}$, а в [19] показали, что можно проводить не линейное квантование, например в степени двойки, и на разном множестве весов для каждой итерации обучения. Используя квантование, можно рассчитывать на увеличение скорости нейронной сети в несколько раз, но не более.

б) Оптимальное прореживание нейронных сетей (от англ. optimal brain damage). Несмотря на использование мощных инструментов регуляризации таких как dropout, некоторые веса нейронной сети приносят больший вклад в формирование результата, чем другие. Идея подхода заключается в том, чтобы удалить такие веса из сети. После удаления выбранных элементов сети, обычно дополнительно проводят стадию донастройки, чтобы сеть приспособилась к новой архитектуре. Один из вариантов применения такого подхода был описан в работе [20], там было показано, что при решении задач обработки изображений можно удалить от 10% до 60% весов без потери качества модели, кроме того, авторы показали, что метод успешно работает вместе с квантованием. Хоть метод и уменьшает размерности внутренних матриц нейронной сети почти без потерь, но вместе с тем является очень ресурсоемким, по сути, чтобы понять какие связи в нейронной сети не нужны используется перебор.

в) Дистилляция знаний (от англ. knowledge distillation). Идея данного подхода была предложена в [21], и заключается в том, чтобы заставить одну модель обучать другую. Такой вариант сжатия является наиболее общим: его

можно применить к большинству моделей машинного обучения, более того помимо сжатия модели в более скромную архитектуру этот подход позволяет проводить более «смелые» эксперименты и сжимать модель в совершенно другие подходы машинного обучения. Это свойство вызывает огромный интерес в научном обществе в последнее время, и интенсивно исследуется.

На момент написания данной работы не известно почему именно работают данные подходы, однако все сходятся в интуитивном суждении, что каждый из них по-своему избавляется от лишней информации. Под лишней информацией понимают то переобучение на обучающую выборку, которое случилось при формировании обобщающей способности модели в процессе обучения.

Из описания возможных подходов сжатия модели становится понятно, что наибольший потенциал сжатия и ускорения BERT-Base вопросно-ответной модели находится в подходе дистилляции знаний. Исходная BERT-Base модель имеет 12 последовательных слоев transformer, каждый из которых имеет размер выхода 768, что вырождается в 110 миллионов обучаемых весов. Даже если с помощью квантования и удастся уменьшить модель в несколько раз, то этого будет по-прежнему недостаточно: модели со статическими представлениями имеют количество параметров на порядок меньше, т.е. десятки миллионов, кроме того, если в статических моделях основная масса параметров сосредоточена на представлениях токенов, то в BERT-Base архитектуре большая их часть находится в слоях сети, т.е. основное замедление происходит из-за перемножения сравнительно огромных матриц для прохождения сигнала по сети. Проредить такую нейронную сеть тоже не представляется возможным, дообучение BERT-Base занимает длительный промежуток времени, и перебрать удаление достаточного количества связей очень дорого.

На основании приведенных выше рассуждений, для сжатия и ускорения лучшей модели данной магистерской диссертации был выбран и исследован метод дистилляции знаний.

4.2 Теоретические сведения о дистилляции знаний

Изначально метод был разработан для того, чтобы научить одну модель предсказывать значения множества других моделей, каждая из которых являлась экспертом в своей области. Например, если имелся набор вопросно-ответных моделей по разным темам: садоводство, строительство, рыбалка, и др., а именно так и строились первоначальные вопросно-ответные системы 2,

то с помощью дистилляции предпринимались попытки создать одну модель, разбирающуюся во всех сферах. Сейчас же подход эволюционировал, и решает более общий круг задач, его можно описать как процесс передачи знаний одной модели другой.

Опишем формально процесс дистилляции нейронных сетей. Пусть $S^m = \{(x_1, y_1), ..., (x_m, y_m)\}$ тренировочная выборка размера m по поставленной задаче, где x_i - объекты тренировочной выборки, а y_i - истинные метки соответствующих объектов. Если обозначить параметры нейронной сети с помощью θ, тогда ее обучение по ERM парадигме представлено в формуле 4.1:

$$\theta = \underset{\theta}{\operatorname{argmin}} L(y, f(x, \theta)) \tag{4.1}$$

где L — функция потерь, принимающая на вход все истинные метки объектов обучающей выборки, и все предсказанные метки, на основании чего оценивающая ошибку модели;
x — объекты обучающей выборки;
y — истинные метки объектов обучающей выборки;
$f(x, \theta)$ — результат применения нейронной сети с параметрами θ к обучающей выборке x.

Тогда дистиляция знаний нейронной модели с параметрами θ в другую нейронную модель с параметрами $\theta^{'}$ происходит при помощи минимизации функционала, представленного в формуле 4.2:

$$\theta^{'} = \underset{\theta'}{\operatorname{argmin}} L(f(x, \theta), f(x, \theta^{'})) \tag{4.2}$$

где L — функция потерь, принимающая на вход все истинные метки объектов обучающей выборки, и все предсказанные метки, на основании чего оценивающая ошибку модели;
x — объекты обучающей выборки;
$f(x, \theta)$ — истинные метки обучающей выборки, в качестве которых выступают предсказание нейронной сети с параметрами θ;
$f(x, \theta^{'})$ — результат применения нейронной сети с параметрами $\theta^{'}$ к обучающей выборке x.

В контексте дистилляции знаний, полученную нейронную сеть с параметрами θ называют учителем, а сеть с параметрами $\theta^{'}$ называют учеником. Для учителя истинными метками объектов выступают исходные значения y_i,

а для ученика в роли истинных меток выступает результат работы учителя. Таким образом можно сказать, что ученик учится подражать учителю.

Используя метод дистилляции знаний для сжатия модели, априори известно, что:

– В роли ученика выступает более слабая модель, с меньшим количеством параметров. При обучении на исходную задачу ученик достигает меньшего качества, чем учитель.

– Учитель ошибается. Так как в роли учителя выступает другая модель, для сложных задач почти наверняка можно сказать, что она не идеальная.

Учитывая эти факты, возникает вопрос: какой смысл обучать ученика у не очень хорошего учителя, если можно использовать идеального (исходные данные)? Оказывается, смысл есть: эксперименты ведущих в мире ученых показывают, что при использовании подхода дистилляции знаний для многих задач итоговая модель выходит лучше, чем если бы она училась на исходных данных.

После публикации BERT-Base моделей в открытый доступ [11], данная область стала одной из самых исследуемых в области нейронных сетей. Однако, до сих пор никто не может доказать почему это работает, основные предположения, в которых сходится большинство исследований:

– Во время обучения ученик перенимает так называемые темные знания (от англ. dark knowledge). Если рассмотреть вопросно-ответную модель, то для исходного текста такая модель предсказывает два вектора: вектор вероятностей начала ответа, и вектор вероятности конца ответа. В исходной задаче, каждый из этих векторов имеет ровно одну единицу, а все остальные значения нули. Однако в результате применения учителя выходят не такие разреженные вектора, помимо информации о начале и конце ответа, такие вектора содержат дополнительные сигналы о словах и их отношению к вопросу. Именно такие сигналы и называют темными знаниями, считается, что таким образом ученик перенимает представление о «мире» у учителя, и соответственно если учитель - это более сильная модель, то учится эффективнее.

– Модель учителя более сильная, а значит имеет больше шансов заметить какие-то сложные связи между объектами и их метками. Если происходит такое, что при самостоятельном обучении ученик не способен обнаружить эти связи, а учитель их находит, то есть шанс, что это знание передастся через темные знания.

Хоть это и не является основной темой данного исследования, но нельзя не упомянуть, что метод дистилляции знаний иногда настолько успешный,

что даже если ученик и учитель - это модели одинаковой архитектуры, то может случится так, что в результате обучения ученик превосходит учителя [22].

4.3 Дистилляция BERT-Base модели в BiLSTM QA Attention

Для получения еще более хорошего результата при дистилляции BERT-Base QA модели в BiLSTM QA Attention в данной работе использовались дополнительные данные, а именно корпус MS Marco 2.1.3. Данный корпус использовался в качестве способа обучить ученика на большем количестве данных, таким образом результаты ученика можно корректно сравнивать с другими результатами, так как он не использует дополнительной человеческой разметки извне. Проще говоря, корпус MS Marco можно было бы без потерь качества заменить на любые другие вопрос и тексты, например, сгенерированные по википедии.

На рисунке 4.1 представлена схема обучения дистилляции вопросно-ответной модели реализованная в данной работе.

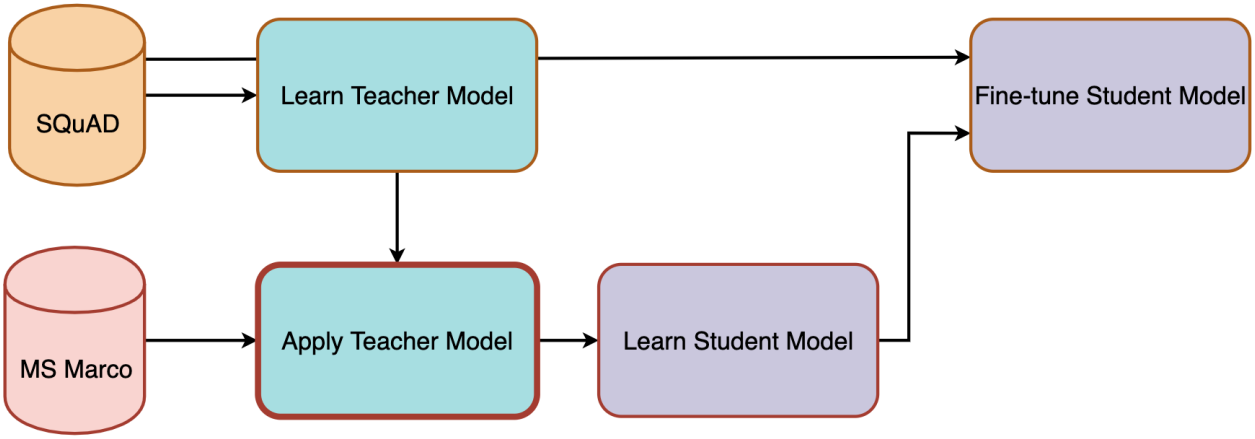

Рисунок 4.1 – Схема дистилляции BERT-Base QA модели в BiLSTM QA Attention с использованием MS Marco для решения задачи SQuAD 2.0

Детальное описание дистилляции, проведенной в этой работе:

а) Для задачи SQuAD 2.0 была дообучена BERT-Base модель 3.3.2.

б) Модель из предыдущего пункта была применена на MS Marco датасете вопросов и текстов.

в) Модель по архитектуре BiLSTM QA Attention 3.3.1 была обучена на смеси MS Marco и SQuAD 2.0 датасетов, состоящей на 70% из данных первого, и на 30% из данных второго соответственно. На данном шаге BiLSTM QA Attention выступала в роли ученика, которого учит BERT-Base модель, примененная на предыдущем шаге.

г) Ученик из предыдущего шага был дообучен на исходную задачу SQuAD 2.0. Данный шаг использовался для того, чтобы не ограничивать ученика возможностями учителя, а также настроить на распределение данных в исходной задаче.

В результате проведения описанной выше дистилляции знаний удалось достичь качества на SQuAD 2.0 равного 0.6789 EM и 0.7084 $F1$.

4.4 Анализ результатов экспериментов

В данном разделе описан эксперимент с дистилляцией лучшей модели, полученной в разделе 3, для решения SQuAD 2.0 задачи. В текущем подразделе будут сведены результаты всех экспериментов и сформулирован вывод по ним. Обозначения используется такие же как и в 3.4. Дистилляция, обученная в 4.3, получила название «BERT-Base Distillation with BiLSTM QA Attention».

В таблице 4.1 представлены собранные вместе результаты проведенных экспериментов данной работы:

Подход	EM	F1
BiLSTM QA Attention	41.21	44.82
BiLSTM QA Attention + Wiki pretrained embeddings	59.98	62.77
BiLSTM QA Attention + FastText pretrained embeddings	62.41	66.15
BERT-Base Distillation with BiLSTM QA Attention	67.89	70.84
BERT-Base	70.85	75.09

Таблица 4.1 – Результаты всех экспериментов текущей магистерской диссертации

На рисунках 4.2 и 4.3 представлены кривые метрик Exact Match и F1 всех экспериментов данной работы на отложенной валидационной выборке в процессе обучения. Первые 7 эпох обучения дистилляции соответствуют обучению ученика на смеси, состоящей на 70% из случайных примеров учителя MS Marco корпуса, и на 30% из случайных примеров учителя корпусе исходной задачи. Все следующие эпохи после 7 соответствуют обучению на исходных данных SQuAD 2.0 без использования учителя. Размер одной эпохи всегда равен размеру тренировочных данных SQuAD 2.0 (это сделано для возможности сравнения скорости обучения).

По метрикам в таблице 4.1 и их визуализациям 3.10,3.11 видно, что качество итоговой модели статических представлений увеличилось на 4.4% - EM и на 4.7% $F1$ в сравнении с таким же подходом, обучающимся на ис-

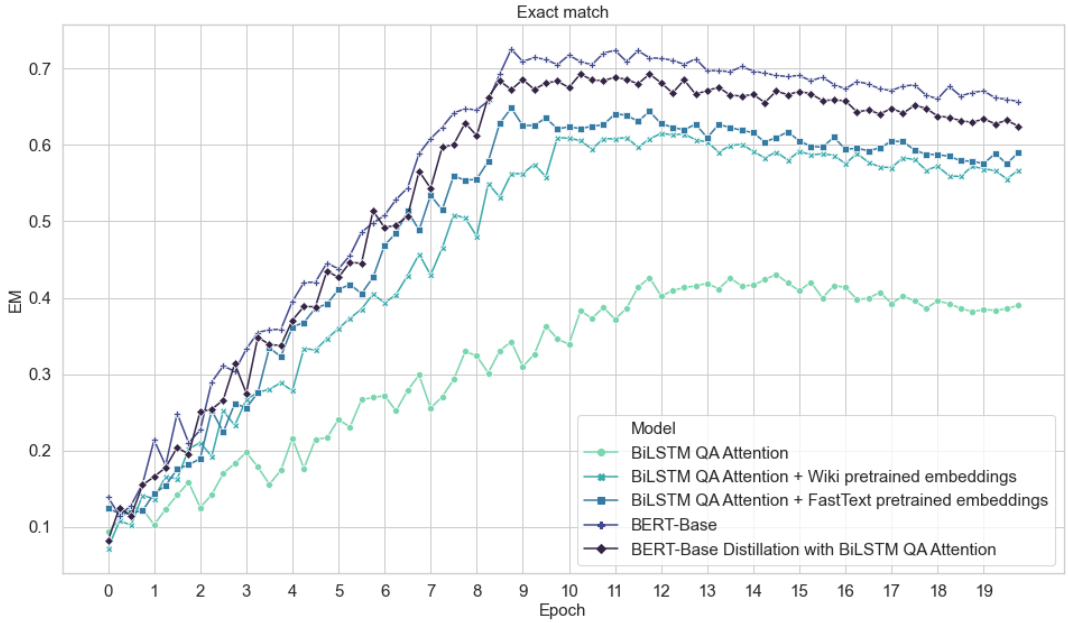

Рисунок 4.2 – Кривая изменения метрики Exact Match дистилляции во время обучения на валидационной выборке в сравнении с другими подходами

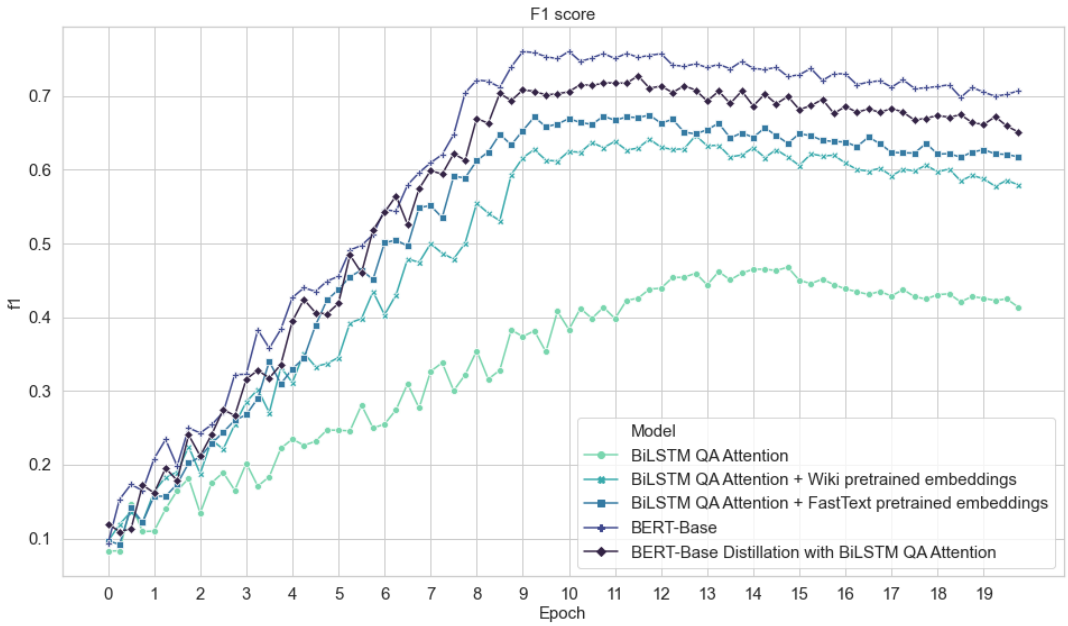

Рисунок 4.3 – Кривая изменения метрики F1 во время обучения дистилляции на валидационной выборке в сравнении в другими подходами

тинные метки объектов. Кроме того, по визуализации видно возможное влияние темных знаний: на первых итерациях дистилляция иногда обгоняет по качеству исходную модель, предположительно так происходит из-за того, что при дистилляции модели поступает дополнительная информация от учителя, и обучение происходит эффективнее. Скорее всего итерации, в которых ученик превосходит учителя, являются погрешностью, однако точно можно сказать, что на первых эпохах ученик(более плохая, маленькая модель) не отстает по качеству от учителя(сильной, на два порядка более тяжелой модели), т.е. улучшился не только итоговый результат, но и скорость обучения.

Таким образом, в данном разделе была исследована возможность уменьшить разрыв в качестве между «легкими» моделями статических представлений и «тяжелыми» моделями контекстных представлений с помощью метода дистилляции знаний. В результате исследования удалось добиться существенных улучшений в «легких» моделях статических представлений токенов. На момент проведения исследования полученные значения метрик входили в лучшие 50 результатов по данной задаче при сравнении метрик на валидационной выборке с публичным тестом данной задачи.

ЗАКЛЮЧЕНИЕ

В рамках работы над настоящей магистерской диссертацией были достигнуты следующие результаты.

– Исследованы основные подходы построения систем поиска ответа на вопрос, в частности, системы основанные на базах знаний, на классических алгоритмах машинного обучения и нейронных сетях; была произведена оценка перспективности новых исследования по каждому из подходов.

– Проанализированы корпуса данных, используемые в мировом научном сообществе для построения вопросно-ответных систем и метрики оценки качества на них. В результате проведенного анализа задача работы была уточнена в терминах качества системы на выбранных данных.

– Изучены сильные и слабые стороны лучших открытых подходов по исследуемой задаче. На основании полученных знаний были сформированы конкретные эксперименты и задачи для выполнения данной работы.

– Разработана опорная, архитектура нейронной сети. Проведены эксперименты с ее обучением и оптимизацией.

– Проведены эксперименты с предварительным обучением разработанной архитектуры на другой задаче. Что дало скачок качества результирующей системы без усложнения архитектуры.

– Проведены эксперименты с использованием внешних предварительно обученных языковых моделей для «теплого» старта обучения разработанной нейронной сети. Гипотеза экспериментов подтвердилось, и подход принес дополнительное качество разрабатываемой системе.

– Исследованы и применены подходы контекстного представления текстов в нейронных сетях. Экспериментально доказано, что контекстные нейронные сети превосходят статические аналоги.

– Изучены подходы сжатия нейронных сетей, проведен успешный эксперимент по сжатию тяжелой контекстной нейронной сети в разработанный опорный вариант, с уменьшением времени донастройки и применения сети до 100 раз.

– Проведен сравнительный анализ разработанных подходов.

– В результате проведенных исследований и экспериментов первоначальный вариант решения поставленной задачи был улучшен на $26.68\ \%EM$ и на $26.02\ \%F1$ без усложнения архитектуры сети.

– Проведен эксперимент с обучением одного из разработанных решений на русскоязычных данных. По результатам проведен сравнительный анализ подхода по двум языкам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Bengio, Y. Learning long-term dependencies with gradient descent is difficult / Y. Bengio, Patrice Simard, Paolo Frasconi // IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council. — 1994. — 02. — Vol. 5. — Pp. 157–66.

[2] Hochreiter, Sepp. Long Short-Term Memory / Sepp Hochreiter, Jürgen Schmidhuber // Neural Comput. — 1997. — Nov.. — Vol. 9, no. 8. — 1735–1780 P. https://doi.org/10.1162/neco.1997.9.8.1735.

[3] Large-scale Simple Question Answering with Memory Networks / A. Bordes, N. Usunier, S. Chopra, J. Weston // ArXiv e-prints. — 2015. — Jun..

[4] NewsQA: A Machine Comprehension Dataset / A. Trischler [et al.] // ArXiv e-prints. — 2016. — Nov..

[5] MS MARCO: A Human Generated MAchine Reading COmprehension Dataset / P. Bajaj [et al.] // ArXiv e-prints. — 2016. — Nov..

[6] SQuAD: 100,000+ Questions for Machine Comprehension of Text / P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang // ArXiv e-prints. — 2016. — Jun..

[7] Rajpurkar, P. Know What You Don't Know: Unanswerable Questions for SQuAD / P. Rajpurkar, R. Jia, P. Liang // ArXiv e-prints. — 2018. — Jun..

[8] SberQuAD – Russian Reading Comprehension Dataset: Description and Analysis / Pavel Efimov, Andrey Chertok, Leonid Boytsov, Pavel Braslavski // arXiv e-prints.

[9] SG-Net: Syntax-Guided Machine Reading Comprehension / Zhuosheng Zhang [et al.] // arXiv e-prints. — 2019. — Aug. — arXiv:1908.05147 P.

[10] Bahdanau, Dzmitry. Neural Machine Translation by Jointly Learning to Align and Translate / Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio // arXiv e-prints. — 2014. — Sep. — arXiv:1409.0473 P.

[11] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova // arXiv e-prints. — 2018. — Oct. — arXiv:1810.04805 P.

[12] Zhang, Zhuosheng. Retrospective Reader for Machine Reading Comprehension / Zhuosheng Zhang, Junjie Yang, Hai Zhao // arXiv e-prints. — 2020. — Jan.. — arXiv:2001.09694 P.

[13] Index of /enwiki/latest. — https://dumps.wikimedia.org/enwiki/latest/.

[14] Efficient Estimation of Word Representations in Vector Space / Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean // arXiv e-prints. — 2013.

— Jan.. — arXiv:1301.3781 P.

[15] Enriching Word Vectors with Subword Information / Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov // arXiv e-prints. — 2016. — Jul.. — arXiv:1607.04606 P.

[16] Crawl models. — https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md.

[17] Attention Is All You Need / Ashish Vaswani [et al.] // arXiv e-prints. — 2017. — Jun.. — arXiv:1706.03762 P.

[18] Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations / Itay Hubara [et al.] // arXiv e-prints. — 2016. — Sep.. — arXiv:1609.07061 P.

[19] Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights / Aojun Zhou [et al.] // arXiv e-prints. — 2017. — Feb.. — arXiv:1702.03044 P.

[20] Han, Song. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding / Song Han, Huizi Mao, William J. Dally // arXiv e-prints. — 2015. — Oct.. — arXiv:1510.00149 P.

[21] Ba, Lei Jimmy. Do Deep Nets Really Need to be Deep? / Lei Jimmy Ba, Rich Caruana // arXiv e-prints. — 2013. — Dec.. — arXiv:1312.6184 P.

[22] Born Again Neural Networks / Tommaso Furlanello [et al.] // arXiv e-prints. — 2018. — May. — arXiv:1805.04770 P.