Содержание

[Введение 2](#_Toc102146672)

[Постановка задачи 4](#_Toc102146673)

[1. Обзор литературы 5](#_Toc102146674)

[1.1. Тестовые задания 5](#_Toc102146675)

[1.2. Модель seq2seq 9](#_Toc102146676)

[1.3. Генеративные модели 17](#_Toc102146677)

[1.4. Генерация вопросов 21](#_Toc102146678)

[1.5. Генерация дистракторов 24](#_Toc102146679)

[1.6. Выводы 33](#_Toc102146680)

[2. Разработка метода генерации тестовых заданий 34](#_Toc102146681)

[2.1. Набор данных 34](#_Toc102146682)

[2.2. Метод генерации тестовых заданий 36](#_Toc102146683)

[1. Обработка текста 36](#_Toc102146684)

[2.4. Выводы 39](#_Toc102146685)

[3. Реализация метода генерации тестовых заданий на основе извлечения именованных сущностей 40](#_Toc102146686)

[3.1. Обучение модели 40](#_Toc102146687)

[3.2. Полученные результаты 41](#_Toc102146688)

[3.3. Выводы 42](#_Toc102146689)

[Заключение 43](#_Toc102146690)

[Список литературы 44](#_Toc102146691)

# Введение

Методы автоматической обработки текстов (АОТ) на естественном языке изучаются применительно к различным предметным областям, включая образование. К примеру, на всемирно известных научных конференциях ACL и NAACL, посвященных вопросам АОТ и компьютерной лингвистики, регулярно проводятся специализированные семинары на тему разработки программного обеспечения для сферы образования с применением АОТ: Innovative Use of NLP for Building Educational Applications («Инновационное применение АОТ для построения образовательных приложений»). Одной из актуальных задач, рассматриваемых исследователями в этой отрасли, является генерация тестов контроля знаний.

Актуальность этой задачи обусловлена тем, что разработка тестов контроля знаний представляет собой трудоемкий процесс, требующий существенных временных затрат от преподавателя. В настоящее время доступно множество программных средств, которые автоматизируют проведение и оценку тестов, но обладают ограниченными возможностями композиции тестовых заданий. Например, популярная в российских вузах система дистанционного обучения Moodle предполагает наличие банка тестовых вопросов, который необходимо заполнять вручную. Система позволяет вносить элемент случайности в задания отдельного типа (т. н. вычисляемые вопросы), однако смысловое содержание заданий целиком и полностью определяется преподавателем. При этом очевидно, что содержание заданий во многих случаях отражает содержание учебных пособий и других текстовых источников, задействуемых в преподаваемом курсе. Отсюда вытекает предположение, что процесс извлечения информации из текста с целью композиции тестовых заданий по данному тексту можно автоматизировать.

Целью настоящей работы является разработка и исследование методов автоматической генерации тестовых заданий для контроля знаний путем извлечения фактов из текстов на русском языке.

Важно отметить разницу между двумя связанными терминами: тестовое задание представляет собой отдельное упражнение того или иного вида, входящее в состав теста, в то время как сам тест есть средство измерения знаний, готовое к применению в педагогической практике. В данной работе предполагается, что тест из отдельных тестовых заданий в конечном итоге компонуется преподавателем, который самостоятельно контролирует такие педагогические характеристики, как валидность получаемого теста, а рассматриваемый метод используется для генерации содержательной основы тестов, т. е. для наполнения банка вопросов некоторой тестирующей системы, такой как Moodle.

# Постановка задачи

Цели данной работы:

1. Исследовать типологию тестовых заданий и вопросы структуры и содержания тестовых заданий

2. Исследовать модели генерации тестовых заданий

3. Исследовать источники данных для обучения системы генерации тестовых заданий

Для достижения этих целей должны быть решены следующие задачи:

1. На основе проведенных исследований разработать метод автоматизированной генерации тестовых заданий на основе использования модели T5 адаптированной для русского языка.

2. Реализовать метод автоматизированной генерации тестовых заданий закрытого типа извлекаемых из системы повествовательных предложений (сентенций), представляющих собой определение или объяснение термина на русском языке.

# 1. Обзор литературы

## 1.1. Тестовые задания

Идея автоматизированной подготовки тестов контроля знаний довольно широко изучается в научной литературе. В русскоязычных источниках некоторые авторы описывают подходы, обеспечивающие создание множества тестовых заданий посредством методов генерации текста. В работах А. П. Сергушичевой и А. Н. Швецова описан метод, который использует формальное описание процессов генерации тестовых заданий средствами канонических исчислений Э. Поста и позволяет формировать структуру и содержание прикладной тестовой системы, определяя лингвистическое содержание конкретных тестов с помощью контекстносвободных грамматик. Одно из преимуществ метода состоит в том, что после составления грамматики он дает возможность легко порождать множество различных вариантов одного теста, используя в тесте более сложные структуры, чем упомянутые выше вычисляемые вопросы Moodle. Сложность метода заключается в том, что преподавателю требуется самостоятельно составлять грамматики, а это может оказаться весьма трудоемкой задачей.

Отчасти схожий метод генерации заданий для компьютерного тестирования разработан В. В. Кручининым . Метод основан на шаблонах для символьных преобразований текста. Под шаблоном здесь понимается заготовка текста, в которой некоторые элементы могут изменяться в соответствии с заданным алгоритмом. Шаблоны, как и грамматики, требуется разрабатывать вручную.

Л. Г. Алсынбаевой [3] разработан метод автоматизированной генерации тестов по программированию, в котором продукционные формализмы используются для генерации не только текстовых фрагментов, но и фрагментов программ.

В и А. Н. Швецовым и соавторами предложен метод, позволяющий подойти к задаче с другой стороны. Основная идея метода заключается в том, что посредством АОТ из составляющих текст утвердительных предложений можно автоматически генерировать вопросы, которые затем будут отбираться, корректироваться и редактироваться преподавателем. Эта идея взята за основу настоящей работы.

Аналогичные методы описаны, в основном, в зарубежной литературе. Например, известен проект LISTEN (Literacy Innovation that Speech Technology ENables), разрабатываемый в Университете Карнеги-Меллон. Цель проекта - разработка обучающей системы, способной выводить на экран компьютера тексты и распознавать речь обучаемых при чтении этих текстов. Одним из компонентов проекта является система разработки тестовых заданий на заполнение пропусков .

Р. Митков и др. предложили подход к созданию системы, которая с помощью технологий АОТ генерирует тестовые вопросы из электронных документов, задействуя ресурсы онтологий. Важной особенностью системы является то, что она позволяет генерировать не только вопросы, но и неправильные, отвлекающие варианты ответов (дистракторы), семантически соответствующие тому или иному вопросу.

М. Хайлмен разработал систему, в которой, помимо прочего, реализован алгоритм статистического ранжирования порождаемых вопросов. Система осуществляет генерацию фактологических вопросов из материалов, которые выдаются обучаемым на чтение. На вход системы подается некоторый источник текста. Система сначала генерирует крупный набор потенциально полезных вопросов, а затем ранжирует их при помощи статистической модели качества вопросов.

Следует отметить, что перечисленные зарубежные проекты ориентированы на обработку англоязычных текстов, а методы обработки текста сильно зависят от конкретного языка. Поэтому исследование подобных методов для русскоязычных текстов представляет существенную актуальность.

Этапы разработки теста:

* Замысел. В основу замысла теста ложится концепция знания предмета, представляющая собой структуру учебной дисциплины.
* Цель. Четко формулируется, для чего разработан тест, какие функции выполняет.
* Условия применения. Определяется контингент, подлежащий тестированию (курс, объем знаний, период обучения), время на тестирование, указывается безмашинный (анкетный) или автоматизированный тесты.
* Анализ разделов, тем. В зависимости от важности разделов делается примерная процентная раскладка знаний по разделам и составляется общий план теста с учетом содержания заданий.
* Определение вида тестовых заданий. Используются тестовые задания различных видов в зависимости от темы раздела, концепции разработки. Для одной группы вопросов нужно апробировать различные типы тестовых заданий. Критерии отбора – показатели трудности и показатели коррелируемости задания с суммарным баллом.
* Экспертиза тестовых заданий. Цель экспертизы: с привлечением квалифицированных специалистов определить, можно ли с помощью предлагаемых тестовых заданий оценить знания. Экспертиза должна быть формализована с применением количественной оценки мнений экспертов.
* Эмпирическая проверка теста. Проверке подлежат все тестовые задания (по 40-50 заданий на 100-200 испытуемых). Результаты тестирования подвергаются статистической обработке.

Требование к тестовым заданиям:

* Содержательность – обязательная принадлежность к теме, разделу дисциплины, для которой составляется тест.
* Краткость – четкая, короткая формулировка, ясное изложение, требующее однозначность ответа, невозможность двойного исполнения при формировании задания желательно придерживаться прямого порядка слов. Вместе с ответом задание должно составлять утверждение, а не вопрос-ответ. В начале предложения следует ставить ключевое слово.
* Калибровка по трудности – в ходе апробации теста определяется мера трудности каждого задания. В тесте должны быть задания различной трудности.
* Взаимосвязь – задания должны быть взаимосвязаны по содержанию (из любого раздела, темы) и коррелировать между собой.
* Дифференцирование – разделение студентов на сильных и слабых. Если на тестовое задание никто не отвечает или отвечают все, такие задания исключаются.

Наибольшее распространение в педагогической практике нашли четыре формы тестовых заданий.

* Наиболее часто в вузовской практике использовались и используются тестовые задания закрытой формы, когда предлагается несколько вариантов готовых утверждений (ответов), из которых нужно выбрать истинные. Как правило, дается 4-5 ответов. Наибольшая трудность в составлении тестовых заданий с выборочными ответами заключается в подборе правдоподобных, но все-таки неправильных ответов.
* Открытая форма тестовых заданий: она помогает студенту овладеть ключевыми словами учебных дисциплин. По этой форме формулируется предложение, в конце которого делается пробел, в который тестируемый записывает произвольный ответ (задание на дополнение). При составлении открытых заданий они должны быть корректными, не допускающими неоднозначного ответа, в задании должно быть не более 7-12 слов.
* Задание на соответствие. Необходимо установить соответствие элементов одного множества элементам другого. При таком методе тестирования оценивается либо каждый вопрос (что желательно), либо целиком полный правильный ответ.
* Тестовые задания на установление правильной последовательности тех или иных действий, процессов, операций.

В литературе (например, [[3]](#bookmark126), [[4]](#bookmark127), [[5]](#bookmark128)) выделяют три основные проблемы, связанные с тестовыми заданиями:

* Генерирование вопросов (QG - question generation) - задача создания правильного вопроса на основе некоторого текста, изображения или базы знаний.
* Генерирование дистракторов (DG - distractor generation) - задача генерирования правильного дистрактора, при заданных вопросе и параграфе (необязательно).
* Системы ответов на вопросы (QA - question answeing) - задача дать правильный ответ на вопрос, учитывая сам вопрос и варианты (необязательно).

Это широко используемая структура, которая может применяться как для текстовой и визуальной модальностей, так и для их комбинации. Несмотря на различия в модальностях, методы, используемые для визуальных модальностей, могут быть применены к текстовым, и наоборот.

## 1.2. Модель seq2seq

Поскольку QG и DG являются задачами обработки естественного языка (NLP), очень важно понимать наиболее распространенные модели и методы NLP, которые также применимы в задачах QG и DG. Применяемые модели глубокого обучения можно классифицировать как seq2seq и генеративные модели. Схема обучения "nоследовательность-nоследовательность" (seq2seq) принимает входную последовательность переменной длины в качестве входа и производит последовательность переменной длины в качестве выхода. Обучение Seq2seq нашло свое применение в различных областях: Нейронный машинный перевод, распознавание речи, резюмирование текста, создание подписей к изображениям и многие другие.

RNN Encoder-decoder. Энкодер-декодер - одна из самых популярных схем преобразования последовательностей в последовательности. Нейронная сеть RNN Encoder- decoder была впервые представлена в [[19]](#bookmark129) автором Cho и др. RNN расшифровывается как Recurrent neural networks (рекуррентная нейронная сеть, РНС), RNN называются рекуррентными, потому что они выполняют одну и ту же задачу для каждого элемента последовательности таким образом, что выход модели зависит от предыдущих вычислений и предыдущих входов. Это означает, что выходная лексема зависит от предыдущих входов и выходов. Модель принимает исходную последовательность  в качестве входа и производит целевую последовательность  в качестве выхода. Модель состоит из двух основных компонентов (оба реализованы в виде RNN): энкодера, который обучается данным и создает скрытый вектор контекста, и декодера, который генерирует последовательность в целевой области на основе скрытого вектора. Архитектура модели представлена на Рис. 1.1.

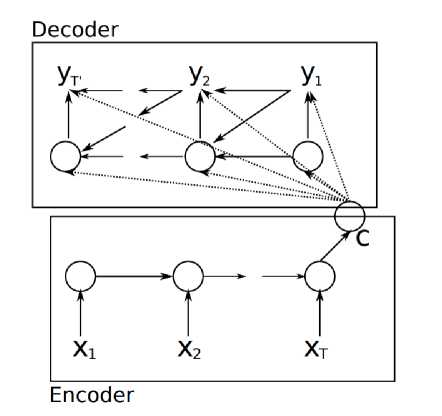


Рис. 1.1: RNN Архитектура энкодера-декодера из [[6]](#bookmark129)

Скрытое состояние кодирующей РНС обучается путем обработки каждого токена из входной последовательности следующим образом: , где  - текущее скрытое состояние,  - скрытое состояние от предыдущей метки времени,  - текущий входной токен, а f - некоторая нелинейная функция. На основе изученных скрытых состояний ht формируется контекстный вектор c: , где q - нелинейная функция. Контекстный вектор рассматривается как сжатие или обобщение входной последовательности x.

Целью декодера является предсказание следующего токена  с учетом скрытого состояния  . Скрытое состояние декодера  и условная вероятность генерации следующего токена вычисляются следующим образом:


где f , g - нелинейные функции активации.

Авторы показали, что модель способна улавливать лингвистические закономерности и предлагать хорошо сформированные целевые фразы в задаче машинного перевода.

LSTM (Long Short Term Memory) является одной из самых популярных РНС. Модель LSTM была впервые представлена в [[20]](#bookmark130) авторами Hochreiter и Schmidhuber, а затем модифицирована, расширена и применена многими исследователями. Авторы показали, что для длинных последовательностей у RNN есть две проблемы: градиенты либо раздуваются (explode), либо исчезают (vanish).

Рекуррентная нейронная сеть имеет форму повторяющейся цепочки ячеек. Повторяющаяся ячейка в обычной РНС имеет только один слой, в то время как LSTM имеет четыре слоя (см Рис. 1.2). LSTM состоит из трех компонентов, известных как ворота: входные ворота (обновляется ячейка), ворота забывания (память устанавливается в 0), выходные ворота (текущая информация отображается в будущем).

Слой Forget Gate (Ворота забывания) решает, какая информация может быть отброшена из состояния клетки. Он учитывает предыдущее скрытое состояние ht -1 и текущий вход xt и возвращает значение от 0 до 1 для каждого числа в состоянии клетки: 0 означает "полностью сохранить 1 означает "полностью отбросить".

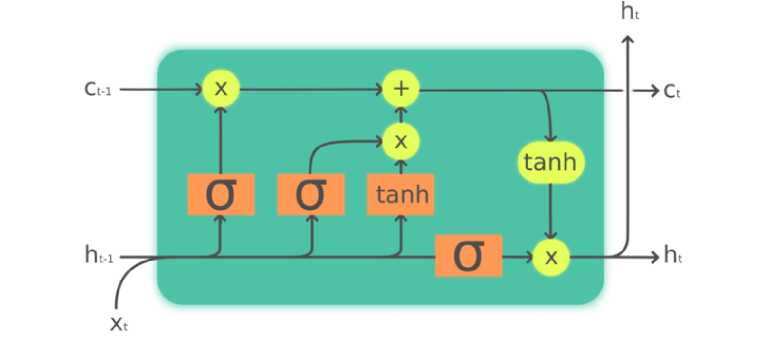


Рис. 1.2: A LSTM cell from [[21]](#bookmark131). Оранжевые прямоугольники представляют слои, желтые эллипсы - точечные операции, а стрелки, расходящиеся из одной точки - операции копирования



Следующий шаг - решить, какая информация будет храниться в состоянии клетки. Сначала сигмоидальный гейт входного слоя определяет, какие значения должны быть обновлены. Затем tanh-слой строит вектор новых значений-кандидатов , которые могут быть добавлены в состояние клетки.





После этого нужно заменить старое состояние клетки  на новое . Умножаем старое состояние  на , забывая то, что решили забыть. После этого мы добавляем , значения кандидатов, умноженные на it, балл обновления:

t

Наконец, мы должны решить, какую информацию мы хотим получить в качестве выходных данных. Выходные значения зависят от ячейки, к которой применяются некоторые фильтры. Сначала мы применяем сигмоидный слой, который решает, какая информация из состояния клетки будет выведена. Затем значения состояния клетки пропускаются через tanh-слой для получения значений в диапазоне [-1; 1] и умножаются на выход сигмоидного слоя, что позволяет получить на выходе только необходимую информацию.



Наиболее выдающиеся результаты среди РНС получены с помощью LSTM, для большинства задач они работают намного лучше, чем простые РНС. Возникает естественный вопрос: каков следующий большой шаг? Наиболее распространено мнение, что следующим прорывом станет механизм внимания.

Механизм внимания (attention mechanism) был представлен Vaswani и др. в [[22]](#bookmark132). Авторы утверждают, что стандартная архитектура ED (encoder- decoder) значительно хуже справляется с переводом длинных предложений. Они также отмечают, что модель LSTM работает последовательно (каждый узел нуждается в состоянии предыдущего узла для создания своего собственного состояния) и поэтому не использует возможности современных графических процессоров (предназначенных для параллельных вычислений).

Модель трансформера ([[23]](#bookmark133)) - это версия архитектуры энкодера- декодера, использующая механизм внимания. Он был разработан для имитации поведения человека при переводе предложений: разделить предложение на несколько значимых частей, перевести их по очереди и объединить перевод с имеющимся результатом. Идея механизма внимания основана на контексте, какой район считается важным для конкретного узла. Для этого вычисляются веса внимания, то есть сколько внимания мы уделяем i -му входному слову. В отличие от модели RNN, полная входная последовательность проходит через модель одновременно.

Архитектура модели трансформера показана на Рис. 1.3. Левая часть модели состоит из следующих частей:

* Входное встраивание (Input Embedding) отображает входное слово на вектор из пространства встраивания.
* Позиционное кодирование (Positional Encoding) - это вектор, который придает входному встраиванию контекст, основанный на положении слова в предложении.
* Энкодер:

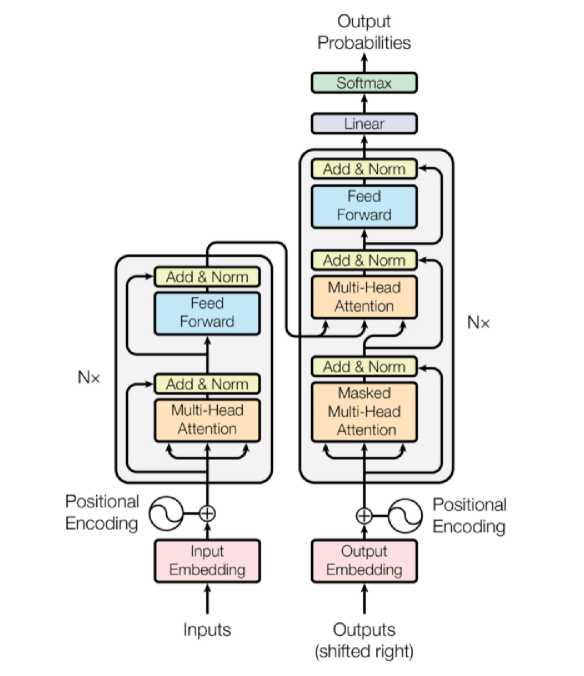


Рис. 1.3: The Transformer - модель архитектуры от [[23]](#bookmark133)

Многоголовое внимание (Multi-head attention) определяет, на какой части входного сигнала должна сосредоточиться модель, и генерирует вектор значений внимания для каждого входного слова.

Сумма & Нормализация (Sum & Norm) суммирует все значения внимания, присвоенные конкретному слову, а затем нормализует их (используется для решения проблемы, когда внимание фокусируется на самом слове, что является бесполезной информацией) .

Нейронная сеть с прямой связью (Feed-forward net) преобразует результаты внимания в форму, более пригодную для использования следующим компонентом.

Компоненты, расположенные в правой части рисунка представления модели, работают с выходными словами (описание для повторяющихся блоков опущено):

* Выходное встраивание, или Output Embedding (например, переведенное предложение из обучающей базы данных).
* Блок декодера: Маскированное многоголовое внимание (Masked Multi-Head Attention): какие слова определяют контекст для каких слов. Оно называется маскированным, потому что при генерировании следующего слова с целевого языка используются все переводы ранее увиденных целевых слов. Многоголовое внимание (внимание энкодера-декодера): сопоставляет векторы входного и выходного внимания.
* Линейный слой (feed-forward layer) расширяет размеры до количества слов на целевом языке.
* Слой Softmax преобразует вектор в распределение вероятности.
* Выходное слово - это слово, которое имеет максимальную вероятность.

Трансформеры способны изучать дальние связи между текстовыми лексемами. Основным недостатком трансформеров является скорость обучения и вывода, однако они поддаются распараллеливанию, что позволяет исследователям использовать возможности современных графических процессоров.

Механизм внимания стал популярен в NLP благодаря модели BERT (Bidirectional Encoder Representations from Transformers) [[24]](#bookmark134), которая использует кодировщики из трансформеров.

Модель BERT была обучена на большом наборе данных для понимания языка и контекста независимо от задачи. Предварительно обученная модель BERT может быть тонко настроена с помощью всего одного дополнительного выходного слоя для решения конкретной задачи (например, ответа на вопрос, автоматического перевода, анализа текста и т.д.). Общие процедуры предварительного обучения и тонкой настройки BERT показаны на рисунке Рис. [1.4.](#bookmark29)

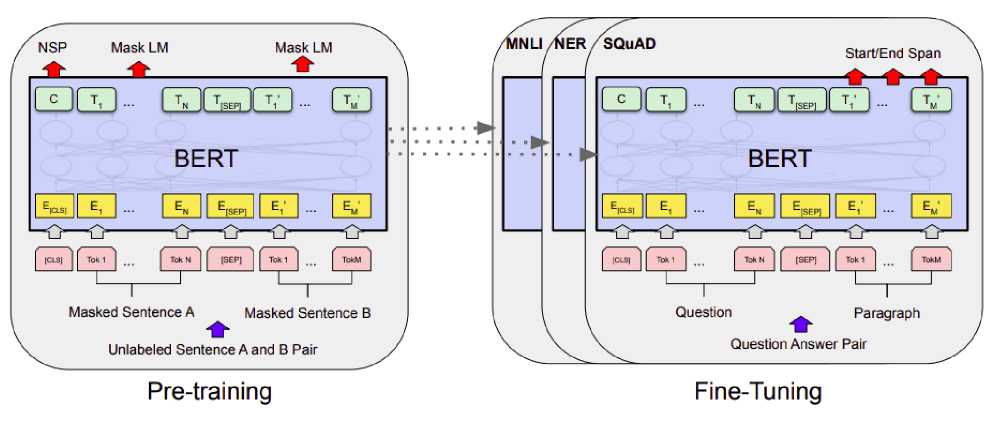


Рис. 1.4: Общие процедуры предварительной подготовки и тонкой настройки для BERT от [[25]](#bookmark134)

Модель предварительно обучена одновременно на двух задачах без надзора: Masked language modeling (MLM), Next sentence prediction (NSP). Задача MLM похожа на вопросы с заполнением пробелов, то есть некоторые слова в предложении заменяются специальным маркером [MASK ],

и модель обучается предсказывать замаскированные слова. Что касается задачи NSP, модель получает на вход два предложения и делает вывод о том, следует ли второе предложение за первым. Обучаясь на этих двух задачах вместе, BERT получает хорошее понимание языка. Задачи решаются одновременно, т.е. на вход модели подаются два предложения, разделенные лексемой [SEP ], в которых некоторые лексемы (слова) заменены на [MASK]. На первом этапе входные слова преобразуются в эмбеддинги с помощью предварительно обученной модели эмбеддинга. Поскольку одновременно решаются две задачи, то и выход модели состоит из двух частей: бинарная оценка классификации для задачи NSP и набор вкраплений для маскированных слов обоих предложений T1, T2,..., TN, T, T2,..., T'N. Адаптация BERT для решения конкретной задачи состоит из следующих частей: тонкая настройка предварительно обученных слоев и обучение дополнительного слоя для получения выходных данных, специфичных для конкретной задачи. Поскольку с нуля обучаются только параметры выходного слоя, а параметры других слоев лишь слегка изменяются, обучение проходит быстро. Devlin и др. описывает конкретное применение BERT для решения задачи ответа на вопрос на наборе данных SQuAD. Вопрос и абзац передаются через модель в качестве входных данных, и модель выдает положение начальных и конечных слов ответа (предполагается, что ответ содержится в абзаце).

Авторы утверждали, что BERT превосходит существующие решения в проблеме ответов на вопросы.

## 1.3. Генеративные модели

Второй тип моделей NN, которые в основном применяются в задачах QG и DG, - это генеративные модели. Генеративные модели, как следует из их названия, генерируют примеры входных данных путем изучения их статистики.

Автоэнкодер, Autoencoder ([[25]](#bookmark135)) - это особый тип глубокой нейронной сети, которая обучается для изучения скрытого представления данных путем проецирования входных данных в кодированное пространство и последующего восстановления (декодирования) кодированного представления на выходе. Более конкретно, автоэнкодер решает следующую задачу: при входных данных , кодер - это функция , декодер - функция , где  - восстановленная версия x. Це лью процесса обучения является минимизация потерь при реконструкции . Пример AE показан в Рис. [1.5](#bookmark34).

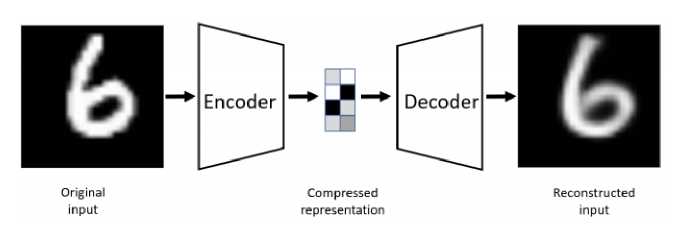


Рис. 2.5: Пример автоэнкодера из [[26]](#bookmark136)

Автоэнкодеры применимы для решения многих задач, не требующих контроля, включая, но не ограничиваясь, сжатие данных, уменьшение размерности, восстановление поврежденных данных и кластеризацию латентного пространства. Существует множество типов автоэнкодеров, наиболее популярными являются следующие ([[26]](#bookmark136)):

* Undercomplete Автокодировщики имеют латентную размерность, которая меньше входной размерности;
* Overcomplete Автокодировщики имеют латентную размерность, которая больше, чем входная размерность;
* Регуляризованные автоэнкодеры используют функцию потерь, которая заставляет модель обладать другими свойствами, кроме способности копировать (реконструировать) входные данные:
* Разреженные автокодировщики штрафуют активацию скрытых слоев так, что только несколько слоев активируются, когда образец подается в сеть;
* Деноизирующие автокодировщики учат наиболее надежные характеристики и отбрасывают шум;
* Вариационные автокодировщики обобщают данные через вероятностное распределение.

Сжатое представление в VAE - это распределение вероятностей. В VAE модель кодера иногда называют моделью распознавания, а модель декодера - генеративной моделью. Для любой выборки скрытых распределений мы ожидаем, что наша модель декодера сможет точно восстановить входные данные. Целью VAE является нахождение распределения  некоторых латентных переменных, которое мы можем выбрать из  для генерации новых выборок  из , где  и  в обозначают параметры кодера и декодера соответственно. Обзор модели показан на Рис. [1.6.](#bookmark36)

encode -> decode ->

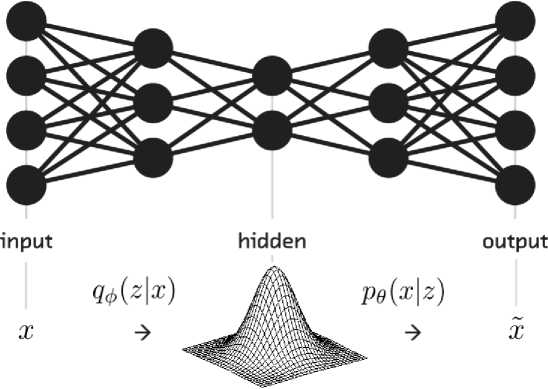


Рис. 1.6: Архитектура VAE от [[27]](#bookmark137)

Потери VAR определяются по следующей формуле:



где DKL означает дивергенцию Куллбэка-Лейблера. p(z) может быть любой функцией распределения вероятности, например, гауссовским распределением.

Модель VAE элегантно сжимает данные в простое распределение вероятности и дает очень хорошие результаты. С другой стороны, изображения, созданные с помощью VAE, как правило, размыты.

Генеративные состязательные сети (GAN) были впервые представлены Goodfellow, Pouget-Abadie, Mirza и др. в [[28]](#bookmark138). Обучение GAN представляет собой игру двух игроков: модель генератора G обманывает дискриминатор, генерируя образцы, похожие на настоящие, а модель дискриминатора D различает настоящие и поддельные образцы. В оригинале [[28]](#bookmark138) образцы - это изображения, но это может быть любой тип данных, который может быть сгенерирован G и дискриминирован D . Обзор фреймворка представлен в Рис. [1.7.](#bookmark38)

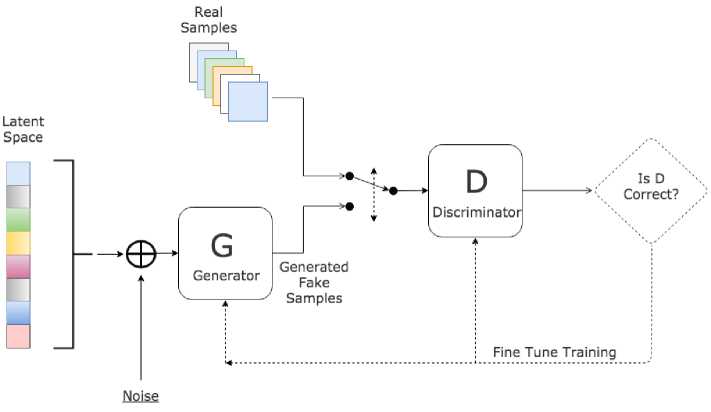


Рис. 1.7: Структура генеративной адверсариальной сети (GAN) от [[29]](#bookmark139)

Целью модели G является изучение распределения вероятностей pmodel(x), которое аппроксимирует распределение вероятностей реальных выборок pdata(x ). Модели G и D могут быть определены формулами:



D ( x ) представляет собой вероятность того, что x получено из реальных данных, а не из pmodel. z - случайный шум, используемый для генерации выборки. Локальный минимум обеих потерь можно определить как равновесие Нэша. Равновесие Нэша достигается, когда Генератор производит изображения, которые Дискриминатор не может отвергнуть (точность дискриминации составляет 0,5). Дискриминатор D обучается для максимизации вероятности присвоения правильной метки как обучающим примерам (т.е. x), так и образцам из G (т.е. G(z)). В то же время G обучается минимизировать log(1-D(G(z))). Тогда функция потерь определяется следующим образом:



Модель GAN нашла свое применение во многих областях и производит высококачественные современные образцы. Тем не менее, она довольно сложна, а обучение нестабильно.

## 1.4. Генерация вопросов и ответов

Поскольку генерация элементов теста требует как QG, так и DG рассмотрим существующие решения по генерации вопросов.

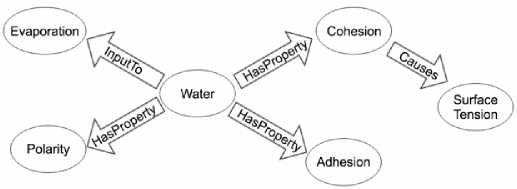


Рис. 1.8: Часть онтологии биологии [[17]](#bookmark127)

В литературе предлагается множество критериев хороших вопросов с точки зрения преподавания и обучения. Наиболее часто обсуждаются четыре критерия, приведенные ниже:

1. Фокусируется на ключевом моменте текста ([[30]](#bookmark140), [[31]](#bookmark141));
2. Сбалансированная сложность ([[30]](#bookmark140));
3. Апеллирует к пониманию и навыкам мышления высшего порядка учащихся, например, резюмирование, анализ отношения, оценка последствий [[17]](#bookmark127).

Применяемые в настоящее время методы делятся на две основные группы: подходы на основе правил и подходы на основе нейронных сетей. Подходы, основанные на правилах, используют структуру данных и созданные вручную правила для создания вопроса. Подходы на основе нейронных сетей являются более сложными и могут отражать более глубокие отношения между объектами.

Подходы, основанные на правилах Методы генерации вопросов на основе правил учитывают структуру данных (онтологию или структуру базы данных, или синтаксис предложения) и применяют некоторые предопределенные правила для создания вопроса.

Stasaski и Hearst [[17]](#bookmark127) применил онтологию Biology для генерации вопросов на основе правил. Пример объектов и отношений показан на рисунке 1.8. Вместо того чтобы генерировать вопрос из триплета узел-отношение- узел, авторы рассмотрели возможность генерации вопросов на основе графовой структуры онтологии. Узел рассматривается как ответ, а три случайно выбранных исходящих отношения - как основа для вопроса. Например, если взять узел "Вода" и его связи (DissolvesIn, "соль"), (HasProperty, "когезия") и (InputTo, "испарение"), то вопрос будет звучать так: "Что растворяет соль, имеет когезию и является исходным для испарения?". После первого эксперимента авторы также изменили алгоритм, чтобы генерировать вопросы на основе двух связей, а не трех, как было изначально, и это увеличило результаты количественной оценки. Они также добавили в систему правила грамматической коррекции, чтобы вопросы викторины звучали более человекоподобно. Ограничение описанного метода заключается в том, что он сильно зависит от онтологии, которая доступна не для всех доменов и тем.

В Heilman в [[32]](#bookmark142) была представлена трехэтапная схема генерации фактологических вопросов. На первом этапе сложные предложения преобразуются в простые фактологические предложения путем извлечения фактологических утверждений и замены местоимений соответствующими существительными. На втором этапе входные предложения преобразуются в набор вопросов-кандидатов путем изменения порядка слов в предложении. Третий этап включал ранжирование набора с помощью статистической модели. Основным недостатком описанной схемы является то, что она требует высокоструктурированных данных с предварительными знаниями о синтаксисе. Таким образом, она вряд ли применима из-за отсутствия соответствующих данных и языков со сложным синтаксисом.

Подходы на основе нейронных сетей Методы, основанные на NN, используют модели глубокого обучения для генерации вопроса.

Первая нейронная сеть QG была представлена Serban и др. в [[33]](#bookmark143). Авторы рассматривали задачу QG как перевод из структурированной базы знаний в вопросы на человеческом языке. Они использовали архитектуру кодер-декодер с механизмами GRU и внимания. Однако модель по- прежнему сильно зависит от базы знаний и поэтому может быть использована только в узких областях с существующими KB.

Jain, Zhang и Schwing [[34]](#bookmark144) попытались решить проблему генерации визуальных вопросов. Они использовали вариационные автоэнкодеры и LSTM для создания набора возможных вопросов на основе входного изображения. Авторы предложили вариационный автоэнкодер. LSTM применяется для кодера (q-распределение) и декодера (р-распределение). Входное изображение и предложение кодируются в латентное представление с помощью q-распределения. Учитывая представление изображения и случайную выборку z, р-распределение реконструирует вопрос. В процессе вывода выборка z генерируется либо из нормального, либо из равномерного распределения. Авторы утверждают, что парадигма вариационного автоэнкодера была выбрана потому, что они более устойчивы при обучении по сравнению с адверсарными сетями. Jain и др. сообщают, что сгенерированные вопросы можно считать либо простыми (на них можно ответить, просто взглянув на изображение), либо сложными (требующими глубокого человеческого понимания объектов и их взаимодействия). Они также выделили два основных типа проблем. Первая называется неудачи распознавания и вызвана неспособностью модели правильно распознать объекты на входном изображении. Второй тип неудач называется co-occurrence based failure, и он приводит к вопросам об объектах, которые не присутствуют на изображении, но часто встречаются вместе с представленными (например, вопросы о птицах на нептичьих изображениях деревьев).

## 1.5. Генерация дистракторов

В некоторых работах [[35]](#bookmark145), [[31]](#bookmark141) говорится, что генерация дистракторов является наиболее сложной задачей при создании элементов теста.

Критерии для хорошего дистрактора, которые были указаны в предыдущих работах:

1. Правдоподобный, трудно отличить от ответа ([[36]](#bookmark146), [[18]](#bookmark128), [[14]](#bookmark124), [[15]](#bookmark125), [[17]](#bookmark127))
2. Semantically Consistent with Question and Answer ([[37]](#bookmark147), [[38]](#bookmark148), [[30]](#bookmark140), [[15]](#bookmark125))
3. Грамматически согласуется с вопросами и ответами ([[37]](#bookmark147))
4. Не является правильным вариантом ([[36]](#bookmark146), [[14]](#bookmark124), [[17]](#bookmark127), [[15]](#bookmark125))
5. Не является модифицированной версией ответа или дистрактора [[37]](#bookmark147)

Самый простой подход - генерировать дистракторы человеком (как в [[39]](#bookmark149)). Однако такой подход требует больших затрат времени и средств, а наша цель - сократить и то, и другое, поэтому мы заинтересованы в автоматической генерации тестовых заданий.

Еще одно важное соображение связано с идеальным количеством дистракторов. Graesser и Wisher [[40]](#bookmark150) доказали, что идеальное количество дистракторов равно трем плюс правильный ответ, следовательно, наша работа подразумевает необходимость генерировать три дистрактора.

Как и в задаче QG, автоматические методы можно разделить на две большие группы: основанные на правилах и основанные на нейронных сетях, обе из которых способны генерировать три дистрактора. Первый метод более интерпретируемый и в основном подходит для однословных дистракторов, в то время как второй может генерировать более сложные дистракторы.

Подходы, основанные на правилах, обычно используют некоторый датасет или базу знаний и метрики сходства для создания дистракторов, которые похожи на ответ.

В [[36]](#bookmark146), [[39]](#bookmark149) авторы предложили использовать частотный подход. В этом методе генерируемый дистрактор - это просто слово в том же POS- теге, что и ключевое слово, которое имеет наибольшее количество вхождений с ним в большом корпусе. Идея основана на интуиции, что слова с близкой частотой встречаемости, вероятно, известны изучающим язык с одинаковым уровнем знаний. Существует огромное ограничение подхода: процедура не подходит для ключей с несколькими словами (метод, применяемый к каждому слову в ключе независимо, будет генерировать неправдоподобную и сомнительную последовательность).

Метод, основанный на коллокациях ([[36]](#bookmark146)) похож на предыдущий, и он производит дистрактор, который имеет наиболее частую коллокацию либо с левым соседним словом ключа, либо с правым соседним словом. Как и в подходе, основанном на частоте, для ключей с более чем одним словом подход, основанный на коллокации, генерирует маловероятный дистрактор.

Lee и др. в [[36]](#bookmark146) применил метод, основанный на неродных английских корпорациях. Метод генерирует отвлекающий предлог, который чаще всего по ошибке ставился между левым и правым соседними словами ключа неродными носителями языка. Авторы показали, что этот подход является наиболее привлекательным для студентов (по сравнению с методами, основанными на частоте и коллокации). Тем не менее, представленный подход вряд ли применим к открытым POS-тега[м[[1]](#footnote-1)](#bookmark2) из-за отсутствия доступных аналогичных данных.

В нескольких исследовательских проектах ([[41]](#bookmark151), [[39]](#bookmark149)) использовались различные измерения лингвистического сходства. К ним относятся следующие методы, но не ограничиваются ими: морфологически[й[[2]](#footnote-2)](#bookmark3) (слова, принадлежащие к различным частям речи, но имеющие одну и ту же базовую форму, bored - boring), орфографический (слова с двумя или тремя измененными буквами, например, bread - beard), фонетические (слова с 2-3 измененными фонемами или с фонемами, замененными на похожие, например, file - fly), и графемный (дистрактор с наименьшим расстоянием Левенштейна от целевого слова). Слова-кандидаты сканируются по большому корпусу, и наиболее близкое слово с точки зрения некоторого измерения сходства считается дистрактором.

Pino и др. в [[41]](#bookmark151) попытался объединить различные подходы, основанные на сходстве, чтобы получить дистракторы, расположенные дальше, чем целевое слово. Однако дистракторы, созданные с помощью смешанных стратегий, таких как OrthMorph (сочетание орфографического и морфологического методов) или PhonMorph (сочетание фонетического и морфологического методов), менее правдоподобны, и эксперименты показали, что испытуемые выбирают такие дистракторы реже.

Liang и др. в [[35]](#bookmark145) расширил меры сходства для ключей с 2-5 словами и сформулировал задачу генерации дистракторов как задачу Learning to rank. После этого авторы свели задачу к задаче бинарной классификации, применив в качестве моделей классификации логистическую регрессию, Random Forest ([[42]](#bookmark152)) и LambdaMART ([[43]](#bookmark153)). Авторы обучили модель, используя дистракторы из входного образца в качестве положительных примеров и подмножество дистракторов из других образцов в качестве отрицательных примеров. Характеристики для классификации были получены путем комбинирования различных мер сходства для вопроса q, ответа a и дистрактора d :

* Emb Sim (сходство встраивания),
* POS Sim (сходство по Жаккарду между POS-тегами а и d),
* ED (расстояние редактирования, или расстояние Левенштейна),
* Token Sim (парное сходство по Жаккарду между токенами q, a и d),
* Length (длина токенов a и d и разница между ними),
* Суффикс (длина самого длинного общего суффикса a и d,
* Freq (средняя частота слов в a и d),
* Single (единственное и множественное число последовательности a и d),
* Num (есть ли числа в a и d),
* Wiki Sim (сходство встраивания сущностей Википедии).

Авторы [[35]](#bookmark145) обнаружили, что каскадное обучение (первая модель выбирает небольшое подмножество кандидатов, а вторая модель производит окончательное ранжирование), работает лучше, чем обучение с использованием одной модели. Liang и др. использовал две модели последовательно, одну за другой, однако параллельное пакетирование с двумя, тремя или более моделями потенциально может работать гораздо лучше.

Zesch и др. в [[15]](#bookmark125) предложил метод, который использует контекстночувствительные и контекстно-нечувствительные правила вывода. Этот метод генерирует дистракторы, которые семантически похожи на целевое слово, но не похожи в конкретном смысле, порождаемом контекстом цели. Например, "сильный" и "мощный" семантически похожи, но контекст, добавленный словом "чай", делает их разными ("сильный чай" против "мощный чай" ). Авторы сначала получили список возможных дистракторов, используя контекстно-зависимые правила (например, для целевого слова "приобрести" дистракторами могут быть "купить", "иметь", "приобрести" ). Затем они применяли контекстно-зависимые правила для создания черного списка дистракторов. Такие слова являются правильными ответами и не могут быть использованы в качестве дистракторов (например, черный список для ключевого слова "приобрести" в предложении "Microsoft приобретает Skype" может быть "купить", "продать", "приобрести" ). Окончательный набор дистракторов получается путем вычитания дистракторов из черного списка из первоначального списка. Недостатком предложенного метода является то, что он не применим ни для длинных ответов и дистрак- торов (из-за недостатка данных), ни для вопросов, заданных в естественной форме (из-за природы метода).

В [[17]](#bookmark127) К. Стасаки и М. Херст используют онтологическую структуру для задач QG и DG. Узлы свойств n1, n2 , n3 связаны с узлом правильного ответа n отношениями r1, r2, r3, соответственно. Чтобы убедиться, что узел-дистрактор m не является правильным ответом на вопрос, авторы проверили, что максимум два узла из n1, n2, n3 связаны с m каким-либо отношением. Они изучили пять подходов к генерации дистракторов:

* Два совпадающих отношения (используются два отношения из r1, r2, r3),
* Одно совпадение и одно новое отношение (используется одно отношение из r1, r2, r3 и новое r4),
* Два новых отношения (ни одно из r1 , r2 , r3 и новые r5 , r6 не используются),
* Одно совпадающее отношение (используется одно отношение из r1 , r2, r3),
* Структура узла (измерение сходства для пары узлов). Сходство структуры узлов рассчитывается по формуле



где cn - множество узлов, которые соединены с узлом n, ln,r - количество связей, которые имеет узел n типа r .

Stasaski и Hearst [[4]](#bookmark127) также предложил подход, основанный на онтологии. Каждый узел n из графа онтологии G состоит из серии слов, и он встраивается как en. Дистрактор выбирается путем сравнения узла- кандидата дистрактора n с узлом правильного ответа n по формуле



где  обозначает косинусное сходство между векторами e n и e m , а m - узел графа G. В другой методике авторы применили приведенную выше формулу для нахождения наиболее похожего узла на каждый из компонентов вопроса n1, n2 и n3. Ограничения подходов, основанных на онтологиях, обусловлены их природой: онтологии обычно специфичны для конкретной области и доступны не для всех доменов.

Подходы на основе нейронных сетей Основываясь на IR-GAN (впервые представленной в [[44]](#bookmark154)), модифицированной версии генеративной состязательной сети для информационного поиска, Liang и др. в [[35]](#bookmark145) предложил состязательную схему обучения, состоящую из двух компонентов: генератора G и дискриминатора D . Генеративная модель G выдает условную вероятность генерации дистракторов с учетом вопроса и ключа - P (d | q, a). Дискриминантная модель D пытается угадать, происходит ли данный дис- трактор из G или из реальных обучающих данных. Несмотря на то, что данный подход основан на нейронных сетях, в ходе экспериментов он показал себя хуже, чем Random Forest. Предположительно, модель может быть улучшена путем обновления архитектуры, тонкой настройки предварительно обученных моделей или оптимизации гиперпараметров.

Gao и др. в [[37]](#bookmark147) предложил использовать сеть обучения по принципу "последовательность-последовательность"и брать в качестве входных данных для NN интервал текста и вопрос для генерации списка дистракторов. Они описали иерархическую кодирующую-декодирующую структуру с динамическими и статическими механизмами внимания. Сначала данные проходят через кодер, который изучает контекстуализированное представление всей статьи на уровне слов и предложений. Затем механизм динамического внимания изучает важность каждого предложения. На третьем этапе статическое внимание используется для того, чтобы заставить динамическое внимание не фокусироваться на нерелевантных предложениях или предложениях, которые способствуют правильному ответу. И наконец, информация о вопросе сжимается, и снова задействуется механизм динамического внимания. Авторы заявили, что явное добавление сигналов наблюдения к обучению статического внимания может улучшить качество генерации дистракторов.

Lu, Ye, Ren и др. [[18]](#bookmark128) сообщают о результатах, связанных с визуальным DG. В этой работе они, во-первых, представили новую задачу генерации текстовых дистракторов для VQA (DG-VQA). Их целью было создание наиболее запутанных дистракторов для задачи VQA. Авторы применили решение reinforcement learning, в котором модель DG получает вознаграждение от хорошо обученной модели VQA в зависимости от правильности выбранного моделью VQA варианта. В терминах RL, модель DG становится агентом политики, сгенерированный дистрактор становится действием, а модель VQA - средой. Применение моделей VQA имеет два основных преимущества. Во-первых, этот подход пытается оценить запутывающую способность дистрактора. Во-вторых, модель является несамостоятельной и не требует маркировки образцов дистракторов. Предложенное соревнование показало, что модели VQA могут быть использованы для обучения более точных и эффективных визуальных моделей QG и DG.

В работе [[45]](#bookmark155) от Zhou, Luo и Wu предлагается использовать иерархическую сеть Co-Attention. Авторы утверждают, что существующие модели Seq2Seq не имитируют взаимодействие между статьей и вопросом, а также отношения между дистрактором и статьей. Предлагаемая модель способна отразить отношения между статьей и вопросом благодаря иерархической архитектуре с усиленным совместным вниманием. Связь между дистрактором и ответом обусловлена дополнительной потерей семантического сходства, что заставляет генерируемый дистрактор быть более релевантным статье. Кодер вычисляет представление предложения, комбинируя следующие вкрапления: представление слов внутри предложения на уровне предложения и представление вектора признаков предложения с учетом вопроса. Состояние декодера инициализируется сжатым вопросом, чтобы обеспечить грамматическую согласованность между дистрактором и вопросом. Иерархическая оценка внимания рассчитывается итеративно с учетом представлений на уровне слов и конечных представлений предложений. В качестве направления будущей работы авторы рассматривают возможность учета правильного ответа. Эта идея потенциально может генерировать более запутанные дистракторы, но при этом может генерировать правильные (недостоверные) варианты.

Chung, Chan и Fan в [[46]](#bookmark156) предлагают новую модель под названием BDG (BERT-based Distractor Generation). Модель итеративно делает вывод, т.е. предсказывает i -ую лексему длинного дистрактора. Изначально на вход модели подается C (комбинация текстового отрезка P , вопроса Q и ответа A ). После этого модель предсказывает следующую лексему t i в дистракторе на основе последовательности C и ранее предсказанных лексем дистрактора t1, t2, ..., ti -1. Чтобы избежать генерации некогерентного вывода, результаты предыдущего декодирования учитываются при декодировании следующего дистрактора. Кроме того, авторы объединили модель BDG с параллельным MLM для повышения производительности модели DG. Chung и др. столкнулись с тем, что модель BGD имеет тенденцию генерировать дистракторы, похожие на ответ A. Чтобы отбить желание предсказывать лексемы из A при предсказании следующей лексемы ti , они ввели отрицательную потерю ответа. Применив мультимодальный подход и введя регуляризацию Answer negative, авторы показали, что их модель является самой современной моделью для задачи DG для набора данных RACE.

Offerijns и др. [[47]](#bookmark157) работали над всеми тремя проблемами: QG, DG, QA. Тем не менее, они сосредоточились на генерации дистракторов. Для этого они использовали модель GPT-2. Как и в исследовании [[46]](#bookmark156), модель принимает контекст C в качестве входных данных, кроме того, она использует последовательность истинных дистракторов в качестве входных данных. Авторы сравнили модели GPT-2-small и GPT-2-medium. Удивительно, но маленькая версия показала лучшие результаты по показателям BLEU и ROUGE. Offerijns и др. использовал несколько примечательных методов. Во-первых, они применили штраф за повторение (предложенный в [[48]](#bookmark158)), который наказывает модель за генерацию синтаксически похожих дистракторов. Во-вторых, заметив, что модель генерирует менее трех дис-

тракторов, они повторяют шаги генерации до тех пор, пока не будет сгенерировано три дистрактора. Наконец, они ввели QA-фильтрацию, чтобы отсеять вопросы и дистракторы, которые в той или иной степени являются неправильными. Эксперименты показали, что автоматические оценки для метода лучше по сравнению с моделями, предложенными в [[37]](#bookmark147), [[45]](#bookmark155).

Другие подходы Альдабе и Маритксалар в [[49]](#bookmark159) предложили различные методы, основанные на латентном семантическом анализе (LSA, [[50]](#bookmark160)), который встраивает слова в векторное пространство на основе их встречаемости в наборе документов. Авторы извлекли из векторного пространства слова-кандидаты на отвлечение, которые лучше всего соответствуют запросу и предложению-носителю. Базовый метод LSA просто возвращает слова, которые не входят в предложение-носитель и имеют одинаковый POS с ключом. LSA & Semantic & Morphology генерирует отвлекающие слова, если они имеют хотя бы один семантический признак, например, animate, human, учитывая формы слов. Подход LSA & Specialised Dictionary отдает предпочтение словам, которые относятся к той же теме в энциклопедическом словаре в качестве ключа. В методе LSA & Knowledge-based авторы предложили использовать PageRank для сортировки отвлекающих узлов в графе LKB (Lexical Knowledge Base) на основе их сходства с узлами, представляющими слова входного предложения.

Существенным критерием для сгенерированного дистрактора является неправильный выбор, поэтому дистрактор не должен семантически совпадать с ответом. Сюда относятся синонимы, переформулировки ключей, а также другие правильные ответы, которые определенно удовлетворяют постановке вопроса и правильно отвечают на него. В тестах, где пункты правил имеют только один правильный ответ, несколько правильных вариантов могут запутать тестируемых. Таким образом, отказ от ненадежных дистракторов крайне важен.

Некоторые работы (например, [[49]](#bookmark159)) проверяют надежность вручную, с участием человека.

Как мы видели в литературе, огромное количество работ основано на идее, что сгенерированный дистрактор, не появляющийся в некоторой коллекции данных или базе знаний, вместе со своими соседними словами может считаться надежным.

Например, Зенч и Меламуд в работе [[15]](#bookmark125) использовали контекстночувствительные правила вывода, принимая во внимание контекст основного предложения. Если сгенерированный дистрактор попадает в черный список, созданный контекстно-чувствительными правилами, он может быть отвергнут.

Работа от Sumita и др. [[51]](#bookmark161) проверяют надежность путем поиска в интернете стволового предложения, где пустота заполняется сгенерированным дистрактором. Они предположили, что если есть хотя бы одно совпадение, дистрактор может быть отвергнут. Несмотря на широкое языковое разнообразие и огромное количество данных в Интернете, нулевое совпадение не обязательно означает, что дистрактор надежен.

Correia, Baptista, Mamede и др. [[39]](#bookmark149) предложили использовать лексический ресурс PAPEL, который поддерживает отношения синоним, гипоним и гипероним и исключить дистракторы с таким типом отношений с целевым словом. Они также использовали онтологическую модель MWN.PT, которая каталогизирует каждое слово в конкретной области. Исследователи использовали это свойство для исключения слов в одной и той же области (например, слова "юрист" и "врач" представляют профессию и могут быть исключены).

## 1.6. Выводы

Таким образом описаны подходы к генерации вопросов в рамках технологии NLP. Подходы связанные с генеративными моделями имеют сложности в реализации и требовательны к исходным данным.

В данной работе будет использоваться модель seq2seq для генерации вопросов и ответов.

# 2. Разработка метода генерации тестовых заданий

## 2.1. Набор данных

T5 – нейросетевая модель для понимания и генерации текста. Изобрели её [в работе от Google](https://arxiv.org/abs/1910.10683) два года назад, и расшифровывается это название как text-to-text transfer transformer. [Трансформер](https://arxiv.org/abs/1706.03762) – это архитектура нейросетей, позволяющая извлекать из текста довольно объёмную информацию. Благодаря этой архитектуре модели типа BERT понимают тексты, а модели типа GPT весьма правдоподобно их генерируют. Text-to-text означает, что модель T5 принимает на вход тексты и "читает" их энкодером (как BERT), а потом "пишет" декодером новые тексты и отдаёт на выход. Слово transfer говорит о цели этой модели: она предобучалась восстанавливать пропущенные фрагменты текста, но при желании её можно дообучить на новые, более полезные задачи: перевод, перефразирование, суммаризация текстов, генерация диалоговых ответов, и т.п.

Модель T5 была представлена в книге Колина Раффела, Ноама Шазира, Адама Робертса, Кэтрин Ли, Шаран Наранг, Майкла Матена, Янци Чжоу, Вей Ли, Питера Дж. Лю в книге «Изучение пределов трансферного обучения с помощью унифицированного преобразователя текста в текст».

Трансферное обучение, при котором модель сначала предварительно обучается задаче с большим объемом данных, а затем настраивается для последующей задачи, стало мощной техникой обработки естественного языка (NLP). Эффективность трансферного обучения привела к разнообразию подходов, методологии и практики. В этой статье мы исследуем ландшафт методов трансферного обучения для НЛП, представляя унифицированную структуру, которая преобразует каждую языковую проблему в формат преобразования текста в текст. В нашем систематическом исследовании сравниваются цели предварительной подготовки, архитектуры, немаркированные наборы данных, подходы к переносу и другие факторы в десятках задач понимания языка. Объединив результаты нашего исследования с масштабированием и нашим новым «Колоссальным чистым просканированным корпусом», мы достигаем самых современных результатов по многим тестам, охватывающим обобщение, ответы на вопросы, классификацию текста и многое другое. Чтобы облегчить будущую работу по переносу обучения для НЛП, мы выпускаем наш набор данных, предварительно обученные модели и код.

T5 — это модель кодировщика-декодера, предварительно обученная на многозадачной смеси неконтролируемых и контролируемых задач, для которой каждая задача преобразуется в формат преобразования текста в текст. T5 хорошо работает с множеством готовых задач, добавляя к входным данным разные префиксы, соответствующие каждой задаче, например, для перевода: перевести с английского на немецкий: …, для обобщения: резюмировать: ….

T5 использует относительные скалярные вложения. Заполнение ввода энкодера может быть выполнено слева и справа.

Набор данных Сбербанка для ответов на вопросы (SberQuAD) — набор данных для понимания прочитанного в стиле SQuAD, созданный в рамках конкурса Сбербанка в 2017 году. Данные состоят примерно из 50 тысяч вопросов в Википедии.

Иходный обучающий набор SberQuAD был разделен на обучающий (45 328) и тестовый (5036).

Data Fields

* id: a int32 feature
* title: a string feature
* context: a string feature
* question: a string feature
* answers: a dictionary feature containing:
  + text: a string feature
  + answer\_start: a int32 feature

Пример данных набора SberQUAD.

{

"context": "Первые упоминания о строении человеческого тела встречаются в Древнем Египте...",

"id": 14754,

"qas": [

{

"id": 60544,

"question": "Где встречаются первые упоминания о строении человеческого тела?",

"answers": [{"answer\_start": 60, "text": "в Древнем Египте"}],

}

]

}

Для демонстрации генерации вопросов использовался набор определений по теме «программирование» из книги В.А. Камаев Технологии программирования https://it.wikireading.ru/58436.

Абстракция — мысленное отвлечение, обособление от тех или иных сторон, свойств или связей предметов и явлений для выявления существенных их признаков.

## 2.2. Метод генерации тестовых заданий

Основная идея предлагаемого метода заключается в создании системы, которая автоматически генерирует тестовые задания по текстам на русском языке (сентенционного характера) и представляет их преподавателю, который, в свою очередь, может использовать их для компоновки теста и последующей оценки знаний.

Система принимает текст на русском на вход и формирует тестовые задания на выходе. Затем тестовые задания представляются преподавателю, которые выбирает и редактирует те задания, которые сочтет полезными.

1. Обработка текста

Метод основан на применении лингвистических процессоров, последовательно обрабатывающих текст учебного пособия, статьи или иного источника информации, на знание которого требуется проверить обучаемых.

Обработка текста выполняется в три основных этапа:

1. Выделение предложений. К тексту применяется графематический алгоритм, разбивающий его на отдельные предложения с учетом таких особенностей, как общепринятые сокращения и сокращения имен собственных.
2. Фильтрация предложений. Набор предложений фильтруется таким образом, чтобы по возможности сохранить в нем наиболее значимые (то есть соответствующие теме текста) предложения.
3. Генерация вопросов. Преобразование предложений либо в вопросы, либо в такие тестовые задания, в которых тестируемый должен заполнить пропуски.

Выделение предложений

В первом приближении предложение есть последовательность символов, заканчивающаяся на символы «.», «!» или «?», однако на практике следует учитывать возможность использования точки в качестве символа сокращения и другие нюансы [9]. В применяемом графематическом алгоритме используются предопределенные наборы общепринятых («г.», «гг.», «и т. д.») и распространенных («т. к.», «т. е.», «т. н.») сокращений, а также учитываются сокращения инициалов в именах собственных («А. С. Пушкин» и т. п.). В результате работы алгоритма получается массив предложений, которые в дальнейшем могут обрабатываться алгоритмами морфологического и синтаксического анализа.

Фильтрация предложений

Очевидно, что не каждое предложение исходного текста будет пригодно для генерации вопросов. Отсюда возникает идея фильтрации набора предложений. Процедура сокращения текстового документа с целью сохранения наиболее важных его частей называется автоматическим реферированием текста. В отрасли АОТ существует два основных подхода к автоматическому реферированию: составление выдержек (выделение характерных фрагментов - как правило, предложений) и формирование краткого изложения (аннотирование; порождение связной аннотации исходного текста). Второй подход реализуется труднее и требует применения более сложных методов лингвистического анализа. В данной задаче главной целью является удаление несущественных предложений (предложения в результате процедуры не обязательно должны составлять связный текст), поэтому целесообразно использовать первый подход.

Общий принцип составления выдержек состоит в том, что каждому предложению назначается оценка, характеризующая его важность, и в результирующий набор включаются предложения с наиболее высокими оценками. Начиная с 1950-х гг. были исследованы различные методы оценивания предложений, которые обычно применяются в сочетании: исключение коротких предложений; учет вводных конструкций (включаются предложения, содержащие слова «в заключение», «в результате» и т. п.); учет положения предложений в документе и абзаце; учет вхождений часто встречающихся терминов (путем взвешивания терминов методом TF-IDF); учет вхождений слов из заголовка текста [10].

Токенизация — это процесс, при котором большое количество текста делится на более мелкие части, называемые токенами. Эти токены очень полезны для поиска шаблонов и считаются базовым шагом для определения стемминга и лемматизации. Токенизация также помогает заменить конфиденциальные элементы данных неконфиденциальными элементами данных.

Обработка естественного языка используется для создания таких приложений, как классификация текста, интеллектуальный чат-бот, сентиментальный анализ, языковой перевод и т. д. Для достижения вышеуказанной цели становится жизненно важным понять шаблон в тексте.

Токенизация предложений: Для выполнения такой задачи нам понадобится как токенизатор предложений NLTK, так и токенизатор слов NLTK для расчета соотношения. Такой вывод служит важной функцией машинного обучения, поскольку ответ будет числовым.

Генерация вопросов

В современных тестирующих системах применяются вопросы и упражнения различных типов: вопросы с множественным выбором, задания на сопоставление и т. п. В качестве отправной точки в настоящей работе исследуется тип вопросов: задания с множественным выбором. Затем метод можно будет расширить для генерации вопросов других типов.

Алгоритмы генерации выполняют построение вопроса по результатам морфосинтаксического анализа предложения. Сюда относятся следующие алгоритмы: вопросы к подлежащему (что?, кто?); вопросы к прилагательным (какой? и т. п.); вопросы к обстоятельству места (где?); вопросы к обстоятельству времени (когда?). Эти алгоритмы требуют наличия развитого морфологического словаря и синтаксического анализатора.

В данной работе будет проведено обучение модели mt5-base от Google для решения задач генерации вопросов и генерации ответов на вопросы.

## 2.4. Выводы

В рамках разработанного метода будет построена реализация системы построения по текстам на русском языке пар вопрос-ответ. Генерация дистракторов, как было показано в Главе 1 является отдельной сложной задачей и требует специального рассмотрения, поэтому мы приведем пример построения дистракторов на основе случайного выбора из «мешка» дистракторов для сентенций имеющих сходны контекст.

# 3. Реализация метода генерации тестовых заданий на основе извлечения именованных сущностей

## 3.1. Обучение модели

Мы проведем обучение модели mt5-base от Google для решения задачи генерации вопросов по тексту и ответов на вопросы по тексту, используя фреймворки transformers и pytorch.

Инициализировать модель можно многозадачной версией. Тут, как и везде в библиотеке transformers, model – это сама нейросеть, а tokenizer – это часть модели, ответственная за сопоставление текстов со словарём: разбиение текстов на числовые токены, и сбор текстов из токенов обратно. optimizer – это объект, отвечающий за градиентный спуск; он нужен только на время обучения.

Пример инициализации обучения модели для оптимизатора Адамаса.

import torch

from transformers import T5ForConditionalGeneration, T5Tokenizer

raw\_model = 'mt5-base'

model = T5ForConditionalGeneration.from\_pretrained(raw\_model).cuda();

tokenizer = T5Tokenizer.from\_pretrained(raw\_model)

optimizer = torch.optim.Adam(model.parameters(), lr=1e-5)

Ниже – полный код обучения модели. Переменная pairs – это список из пар, состоящих из вопроса и ответа из набора SberQUAD.

batch\_size = 16 # сколько примеров показываем модели за один шаг

report\_steps = 200 # раз в сколько шагов печатаем результат

epochs = 4 # сколько раз мы обучаем данные модели

model.train()

losses = []

for epoch in range(epochs):

print('EPOCH', epoch)

random.shuffle(pairs)

for i in trange(0, int(len(pairs) / batch\_size)):

batch = pairs[i \* batch\_size: (i + 1) \* batch\_size]

# кодируем вопрос и ответ

x = tokenizer([p[0] for p in batch], return\_tensors='pt', padding=True).to(model.device)

y = tokenizer([p[1] for p in batch], return\_tensors='pt', padding=True).to(model.device)

# -100 - специальное значение, позволяющее не учитывать токены

y.input\_ids[y.input\_ids == 0] = -100

# вычисляем функцию потерь

loss = model(

input\_ids=x.input\_ids,

attention\_mask=x.attention\_mask,

labels=y.input\_ids,

decoder\_attention\_mask=y.attention\_mask,

return\_dict=True

).loss

# делаем шаг градиентного спуска

loss.backward()

optimizer.step()

optimizer.zero\_grad()

# печатаем скользящее среднее значение функции потерь

losses.append(loss.item())

if i % report\_steps == 0:

print('step', i, 'loss', np.mean(losses[-report\_steps:]))

В результате получим модель для русского языка обученную на генерацию вопросов и ответов.

new\_model\_name = 'qa-model' # название папки

model.save\_pretrained(new\_model\_name)

tokenizer.save\_pretrained(new\_model\_name)

## 3.2. Полученные результаты

Результаты ошибки обучения модели в зависимости от количества эпох и алгоритма оптимизации:

|  |  |  |  |
| --- | --- | --- | --- |
| Эпохи | 2 | 3 | 4 |
| Adagrad | 5,611 | 5,425 | 5,345 |
| Adam | 4,261 | 4,112 | 3,890 |
| ASGD | 4,389 | 4,290 | 4,009 |
| RMSprop | 6,090 | 5,980 | 5,765 |

Большее количество эпох не было исследовано так как обучение модели в рамках одной эпохи на бюджетном ноутбуке без GPU занимало около 50 минут.

Таким образом оптимальное качество (с учетом выполнения на компьютере без GPU) дает оптимизатор Adam с количеством эпох 4.

Сформируем набор данных, состоящий из сентенций на русском языке, размеченных по категориям:

class,phrase

1,"Абстрагирование от проблемы — игнорирование ряда подробностей с тем, чтобы свести задачу к более простой задаче."

1,"Абстрактная машина Дейкстры — применяется в проектировании архитектуры системы, самый нижний уровень абстракции — это уровень аппаратуры. Каждый уровень реализует абстрактную машину с все большими возможностями."

1,"Абстрактный родительский класс — родительский класс, не имеющий экземпляров объектов."

1,"Абстракция — мысленное отвлечение, обособление от тех или иных сторон, свойств или связей предметов и явлений для выявления существенных их признаков."

1,"Абстракция сущности — произвольная абстракция. Объект представляет собой полезную модель некой сущности в предметной области."

Функция генерации ответа или вопроса по тексту:

def generate(text, \*\*kwargs):

inputs = tokenizer(text, return\_tensors='pt')

with torch.no\_grad():

hypotheses = model.generate(\*\*inputs, num\_beams=5, \*\*kwargs)

return tokenizer.decode(hypotheses[0], skip\_special\_tokens=True)

Пример сгенерированных тестовых заданий:

Исходная сентенция: "Экземпляр класса — объект."

Тест:

Что такое экземпляр класса?

+ объект

система

функция системы

Исходная сентенция: "В более узком смысле, эвристика — это догадки, основанные на опыте решения родственных задач."

Тест:

Какие догадки основаны на опыте решения родственных задач?

+ эвристика

Технология структурного программирования

Рассмотрение объекта как комплекса выполняемых им функций

Технология структурного программирования

Исходная сентенция: "Система — множество элементов, находящихся в отношениях и связях друг с другом, которое образует определенную целостность, единство."

Тест:

Какое множество элементов, находящихся в отношениях и связях друг с другом, образует определенную целостность, единство?

+ система

Эвроритм

Рассмотрение объекта как комплекса выполняемых им функций

Технология структурного программирования

## 3.3. Выводы

Мы провели обучение предобученной нейросетевой модели mt5-base от Google для решения задачи генерации вопросов и ответов из текстов на русском языке.

Показано применение данной модели к автоматической генерации тестовых заданий.

Из-за недостатка времени и ресурсов задача генерации качественных дистракторов была решена частично, но это может стать направлением для дальнейших исследований.

# Заключение

В данной работе была рассмотрена задача автоматической генерации тестов из текстов на русском языке.

Был проведен анализ существующих методов генерации вопросов и ответов и выбран метод дообучения предобученной нейросетевой модели для генерации вопросов и ответов из текстов на русском языке.

Рассмотренный метод представляется довольно перспективным в плане облегчения работы преподавателя при составлении тестов. На его основе могут быть построены программные средства для автоматизированной генерации тестовых заданий. В данной работе рассмотрен экспериментальный прототип, генерирующий задания закрытого типа с множественным выбором.

Согласно предварительным экспериментам, разработанная система демонстрирует достаточно хорошие результаты в определенных случаях.

# Список литературы

Сергушичева А. П., Швецов А. Н. Синтез интеллектуальных тестов средствами формальной продукционной системы / Математика, компьютер, образование. Вып. 10. Ч. 1 / Под ред. Г. Ю. Ризниченко. Москва-Ижевск: Научн.-изд. центр «Регулярная и хаотическая динамика», 2003. - С. 310-320.

Кручинин В. В. Генераторы в компьютерных учебных программах. - Томск: изд-во Томск. ун-та, 2003. - 200 с.

Алсынбаева Л. Г. Система автоматизированной генерации тестовых заданий // Программные продукты и системы. 2009. № 4. 2.

Воронец, И. В., Швецов А. Н., Алешин В. С. Универсальная автоматизированная система тестирования знаний и самообучения, основанная на анализе естественно-языковых текстов учебных пособий / Пилотируемые полеты в космос. Сб. докл. Пятой международ. научн.-практ. Конф. 9-10 апреля 2003 г. - Звездный городок Моск. обл.: РГНИИЦПК, 2003. - С. 65-67.

Сергушичева А. П., Швецов А. Н. Гибридный подход к синтезу тестовых заданий в тестирующих системах / Математика, Компьютер, Образование: Сборник научных трудов. Выпуск 13. Том 1 / Под ред. Г. Ю. Ризниченко. Москва-Ижевск, R&C Dynamics, 2006. - С. 215-228.

Project LISTEN - School of Computer Science - Carnegie Mellon University. URL: http://www.cs.cmu.edu/~listen/.

Mitkov R. et al. A computer-aided environment for generating multiple-choice test items // Natural Language Engineering. 2006. 12(2): 1-18.

Heilman M. Automatic Factual Question Generation from Text. - Carnegie Mellon University, 2011.

Riley M. D. Some applications of tree-based modeling to speech and language indexing. -

Proceedings of the DARPA Speech and Natural Language Workshop. - Stroudsburg, PA, USA, 1989. - P. 339-352.

Hynek J., Jezek K. A practical approach to automatic text summarization // Proceedings of the ELPUB 2003 conference. - Guimaraes, Portugal, 2003.

Куртасов А. М. Возможности применения автоматического реферирования для фильтрации текста в автоматизированной системе генерации тестов из учебных пособий // Информатизация процессов формирования открытых систем на основе САПР, АСНИ, СУБД и систем искусственного интеллекта: Материалы 7-й межд. научно-техн. конф. - Вологда: ВоГТУ, 2013.

Методология создания агентно-ориентированных учебных комплексов для подготовки специалистов технического профиля: отчет о НИР (промежуточ.) / Вологодский государственный технический университет; рук. Швецов А. Н. ; исполн.: Горбунов В. А. [и др.]. - М., 2011. - 175 с. - Библиогр.: с. 170-175. - № ГР 01201056386.

Kurtasov A. M. A System for Generating Cloze Test Items from Russian-Language Text // Proceedings of the Student Research Workshop associated with The 9th International Conference on Recent Advances in Natural Language Processing (RANLP 2013). - Hissar, Bulgaria, 2013.

T. Zesch и O. Melamud, «Automatic generation of challenging distractors using context-sensitive inference rules,» в Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications, 2014, с. 143—148.

S. Rakangor и Y. Ghodasara, «Literature review of automatic question generation systems,» International journal of scientific and research publications, т. 5, № 1, с. 1—5, 2015.

K. Stasaski и M. A. Hearst, «Multiple Choice Question Generation Utilizing An Ontology,» в Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, Copenhagen, Denmark: Association for Computational Linguistics, сент. 2017, с. 303—312. dOi: 10.18653/v1/W17-5034. url: https://www.aclweb. org/anthology/W17-5034.

J. Lu, X. Ye, Y. Ren и Y. Yang, Good, Better, Best: Textual Distractors Generation forMulti-Choice VQA via Policy Gradient, 2019. arXiv: 1910. 09134 [cs.CV].

K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk и Y. Bengio, «Learning phrase representations using RNN encoder-decoder for statistical machine translation,» arXiv preprint

arXiv:1406.1078, 2014.

S. Hochreiter и J. Schmidhuber, «Long Short-Term Memory,» Neural Computation, т. 9, № 8, с. 1735—1780, 1997. dOi: 10.1162/neco.1997. 9.8.1735. eprint: https: //doi.org/10.1162/neco.1997.9.8.1735. url: https://doi.org/10.1162/neco.1997.9.8.1735.

G. Chevalier. (2018). «The LSTM cell.» Accessed on 18.10.2020, url: https://commons.wikimedia.org/wiki/File:The\_LSTM\_cell.png.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser и I. Polosukhin, Attention Is All You Need, 2017. arXiv: 1706. 03762 [cs.CL].

J. Uszkoreit, «Transformer: A novel neural network architecture for language understanding,» Google AI Blog, т. 31, 2017.

J. Devlin, M.-W. Chang, K. Lee и K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019. arXiv: 1810.04805 [cs.CL].

D. E. Rumelhart, G. E. Hinton и R. J. Williams, «Learning Internal Representations by Error Propagation,» в Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. Cambridge, MA, USA: MIT Press, 1986, с. 318—362, iSbn: 026268053X.

D. Bank, N. Koenigstein и R. Giryes, «Autoencoders,» arXiv preprint arXiv:2003.05991, 2020.

(2016). «Under the Hood of the Variational Autoencoder (in Prose and Code).» Accessed on 23.10.2020, url: https://blog.fastforwardlabs.com/ 2016/08/22/under-the-hood-of-the-variational-autoencoder-in-prose- and-code.html.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville и Y. Bengio, Generative Adversarial Networks, 2014. arXiv: 1406.2661 [stat.ML].

A. Gharakhanian. (2016). «Generative Adversarial Networks - Hot Topic in Machine Learning.» Accessed on 22.10.2020, url: https://www.linkedin. com/pulse/gans-one-hottest-topics-machine-learning-al-gharakhanian.

R. Mitkov и H. Le An, «Computer-aided generation of multiple-choice tests,» в Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing, 2003, с. 17— 22.

R. Mitkov, H. Le An и N. Karamanis, «A computer-aided environment for generating multiple-choice test items,» Natural language engineering, т. 12, № 2, с. 177, 2006.

M. Heilman, «Automatic factual question generation from text,» дис. ... док., Ph. D. thesis, Carnegie Mellon University, 2011.

I. V. Serban, A. Garcia-Duran, C. Gulcehre, S. Ahn, S. Chandar, A. Courville и Y. Bengio, «Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus,» arXiv preprint arXiv:1603.06807, 2016.

U. Jain, Z. Zhang и A. G. Schwing, «Creativity: Generating Diverse Questions Using Variational Autoencoders,» в Proceedings of the IEEE

Conference on Computer Vision and Pattern Recognition (CVPR), июль 2017.

C. Liang, X. Yang, N. Dave, D. Wham, B. Pursel и C. L. Giles,

«Distractor Generation for Multiple Choice Questions Using Learning to Rank,» в Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications, BEA@NAACL, ACL, 2018, с. 284— 290.

J. Lee и S. Seneff, «Automatic generation of cloze items for prepositions,» в Eighth Annual Conference of the International Speech Communication Association, 2007.

Y. Gao, L. Bing, P. Li, I. King и M. R. Lyu, «Generating Distractors for Reading Comprehension Questions from Real Examinations,» CoRR, т. abs/1809.02768, 2018. arXiv: 1809.02768. url: http://arxiv.org/abs/ 1809.02768.

H. C. Goodrich, «Distractor efficiency in foreign language testing,» Tesol Quarterly, с. 69—78, 1977.

R. P. d. S. Correia, J. Baptista, N. Mamede, I. Trancoso и M. Eskenazi, «Automatic generation of cloze question distractors,» в Second language studies: acquisition, learning, education and technology, 2010.

A. Graesser и R. Wisher, «Question generation as a learning multiplier in distributed learning environments. United States Army Research Institute for the Behavioral and Social Sciences,» Technical Report 1121, тех. отч., 2001.

J. Pino и M. Eskenazi, «Semi-automatic generation of cloze question distractors effect of students' 11,» в International Workshop on Speech and Language Technology in Education, 2009.

L. Breiman, «Random forests,» Machine learning, т. 45, № 1, с. 5—32, 2001.

C. J. Burges, «From RankNet to LambdaRank to LambdaMART: An Overview,» Learning, т. 11, № 23-581, с. 81, 2010.

J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang и D. Zhang, «IRGAN,» Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, авг. 2017. doi: 10.1145/3077136.3080786. url: http://dx.doi.org/10.1145/ 3077136.3080786.

X. Zhou, S. Luo и Y. Wu, «Co-Attention Hierarchical Network: Generating Coherent Long Distractors for Reading Comprehension,» arXiv preprint arXiv:1911.08648, 2019.

H.-L. Chung, Y.-H. Chan и Y.-C. Fan, A BERT-based Distractor Generation Scheme with Multi-tasking and Negative Answer Training Strategies, 2020. arXiv: 2010.05384 [cs.CL].

J. Offerijns, S. Verberne и T. Verhoef, Better Distractions: Transformer- based Distractor Generation and Multiple Choice Question Filtering, 2020. arXiv: 2010.09598 [cs.CL].

N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong и R. Socher, CTRL: A Conditional Transformer Language Model for Controllable Generation, 2019. arXiv: 1909.05858 [cs.CL].

I. Aldabe и M. Maritxalar, «Automatic distractor generation for domain specific texts,» в International Conference on Natural Language Processing, Springer, 2010, с. 27—38.

S. C. Deerwester, S. T. Dumais, G. W. Furnas, R. A. Harshman, T. K. Landauer, K. E. Lochbaum и L. A. Streeter, Computer information retrieval using latentsemantic structure, US Patent 4,839,853, июнь 1989.

E. Sumita, F. Sugaya и S. Yamamoto, «Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in- the-blank questions,» в Proceedings of the second workshop on Building Educational Applications Using NLP, 2005, с. 61—68. url: https://www. aclweb.org/anthology/W05-0210.pdf.

K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman и P. Blunsom, «Teaching machines to read and comprehend,» в Advances in neural information processing systems, 2015, с. 1693—1701.

F. Hill, A. Bordes, S. Chopra и J. Weston, «The goldilocks principle: Reading children's books with explicit memory representations,» arXiv preprint arXiv:1511.02301, 2015.

O. Bajgar, R. Kadlec и J. Kleindienst, «Embracing data abundance: Booktest dataset for reading comprehension,» arXiv preprint arXiv:1610.00956, 2016.

T. Onishi, H. Wang, M. Bansal, K. Gimpel и D. McAllester, «Who did what: A large-scale person-centered cloze dataset,» arXiv preprint arXiv:1608.05457, 2016.

P. Rajpurkar, J. Zhang, K. Lopyrev и P. Liang, «SQuAD: 100,000+ questions for machine comprehension of text,» arXiv preprint

arXiv:1606.05250, 2016.

P. Rajpurkar, R. Jia и P. Liang, Know What You Don't Know: Unanswerable Questions for SQuAD, 2018. arXiv: 1806.03822 [cs.CL].

P. Efimov, A. Chertok, L. Boytsov и P. Braslavski, «SberQuAD — russian reading comprehension dataset: Description and analysis,» в International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2020, с. 3—15.

A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman и K. Suleman, «NewsQA: A machine comprehension dataset,» arXiv preprint arXiv:1611.09830, 2016.

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder и L. Deng, «MS MARCO: A human-generated machine reading comprehension dataset,» 2016.

M. Joshi, E. Choi, D. S. Weld и L. Zettlemoyer, «TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension,» arXiv preprint arXiv:1705.03551, 2017.

D. Khashabi, T. Khot, A. Sabharwal, P. Clark, O. Etzioni и D. Roth, «Question answering via integer programming over semi-structured knowledge,» arXiv preprint arXiv:1604.06076, 2016.

H. Shibuki, K. Sakamoto, Y. Kano, T. Mitamura, M. Ishioroshi, K. Y. Itakura, D. Wang, T. Mori и N. Kando, «Overview of the NTCIR-11 QA-Lab Task.,» в Ntcir, 2014.

A. Penas, C. Unger и A.-C. N. Ngomo, «Overview of CLEF question answering track 2014,» в International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2014, с. 300—306.

G. Lai, Q. Xie, H. Liu, Y. Yang и E. Hovy, «RACE: Large-scale reading comprehension dataset from examinations,» arXiv preprint arXiv:1704.04683, 2017.

M. Richardson, C. J. Burges и E. Renshaw, «MCTest: A challenge dataset for the open-domain machine comprehension of text,» в Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, с. 193—203.

K. Papineni, S. Roukos, T. Ward и W.-j. Zhu, «BLEU: a Method for Automatic Evaluation of Machine Translation,» 2002, с. 311—318.

C.-Y. Lin, «ROUGE: A package for automatic evaluation of summaries,» в Text summarization branches out, 2004, с. 74—81.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest и A. M. Rush, HuggingFace's Transformers: State-of-the-art Natural Language Processing, 2019. arXiv: 1910.03771 [cs.CL].

1. https://universaldependencies.org/u/pos/ [↑](#footnote-ref-1)
2. Термины ‘морфологический‘ и ‘морфологический‘ относятся к одному и тому же понятию. Для простоты и последовательности термин ‘морфологический‘ используется везде [↑](#footnote-ref-2)