Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.М07-мм

Анализирование традиционных методов и современных подходов к анализу языка

Карими Хурматулла

Отчёт по учебной практике
в форме «Теоретическое исследование»

Научный руководитель:
к. ф.-м. н.доцент Д. В. Луцив

Санкт-Петербург
2023

Оглавление

Введение

Существуют адекватные традиционные методы, которые приводят нас к созданию средства проверки грамматики. Эти методы использовались для различных частей NLP и, в частности, для систем проверки грамматики, например: Natural Language Toolkit (NLTK) - это платформа, используемая для создания программ на Python, которые работают с данными человеческого языка для использования в статистической обработке естественного языка (NLP).NLTK использует методы, основанные на правилах, этот метод хорош для базовой проверки грамматики, с другой стороны, NLTK обладает производительностью и масштабируемостью, которые довольно хороши для базовой проверки грамматики. Pymorphy2 - это морфологический анализатор, который анализирует русские тексты с помощью словаря opencorpora. Это один из наиболее распространенных алгоритмов искусственного интеллекта, который использовался при многоклассовой классификации, прогнозировании кликов. Pymorphy2 в первую очередь фокусируется на морфологическом анализе, который включает в себя классификацию форм слов и грамматических особенностей.Pymorphy2 не предоставляет комплексных решений для устранения неоднозначности в продвинутых системах проверки грамматики. LemmInflect - использует словарный подход для лемматизации английских слов и преобразования их в формы, заданные предоставляемым пользователем тегом Universal Dependencies или Penn Treebank.LemmInflect в первую очередь сосредоточен на формах словаря лемматизации, в то время как для создания средства проверки грамматики это всего лишь один аспект. LemmInlfect фокусируется только на английской лемматизации, в то время как он может быть полезен для английского языка, но не для других языков. Таким образом, все эти алгоритмы могут быть пригодны для построения базовых систем проверки грамматики, а для более продвинутой системы проверки грамматики эти традиционные методы и инструменты не будут адекватными и ответственными.Таким образом, для решения этой проблемы и преодоления ограничений традиционных мето-

дов необходимо проанализировать современные методы, которые могут обеспечить расширенные возможности и позволить разработать более совершенные системы проверки грамматики.

1. Цель работы

Для анализа современных методов и выбора подходящего метода для реализования были поставлены следующие задачи. следующие задачи:

1. проанализировать традиционные и современные методы.
2. Выбрать лучшую модель, которая может соответствовать вашим требованиям.
3. Реализовать выбранный алгоритм в систему.

2. Обзор

2.1. Обзор NLTK

Инструментарий естественного языка (NLTK) - это платформа, используемая для создания программ на Python, которые работают с данными человеческого языка для применения в статистической обработке естественного языка (NLP).Он содержит библиотеки обработки текста для токенизации, синтаксического анализа, классификации, выделения элементов, пометки и семантического обоснования. Он также включает графические демонстрации и примеры наборов данных, а также сопровождается кулинарной книгой и книгой, в которой объясняются принципы, лежащие в основе задач обработки языка, которые поддерживает NLTK [14].Итак, как мы знаем, NLTK слишком велик, чтобы его объяснять, но мы сосредоточимся на очень специфической области нашей работы: создании структур и внедрении различных анализаторов для наших структур.

2.1.1. Рекурсия в синтаксической структуре

Грамматика называется рекурсивной, если категория, встречающаяся в левой части произведения, также появляется в правой части произведения, как показано рис 1. Производственный Nom -> Adj Nom (где Nom - категория номиналов) включает прямую рекурсию по номинации категории, тогда как косвенная рекурсия по S возникает в результате комбинации двух производств, а именно S -> NP VP и VP -> V S. [7]

2.1.2. Синтаксический анализ рекурсивного спуска (метод сверху вниз)

Это один из самых простых методов синтаксического анализа, который мы можем использовать для наших грамматических структур. Этот метод делит цели на подцели, и каждая подцель подразделяется на другую подцель до тех пор, пока структура ваших предложений не будет завершена. [8]

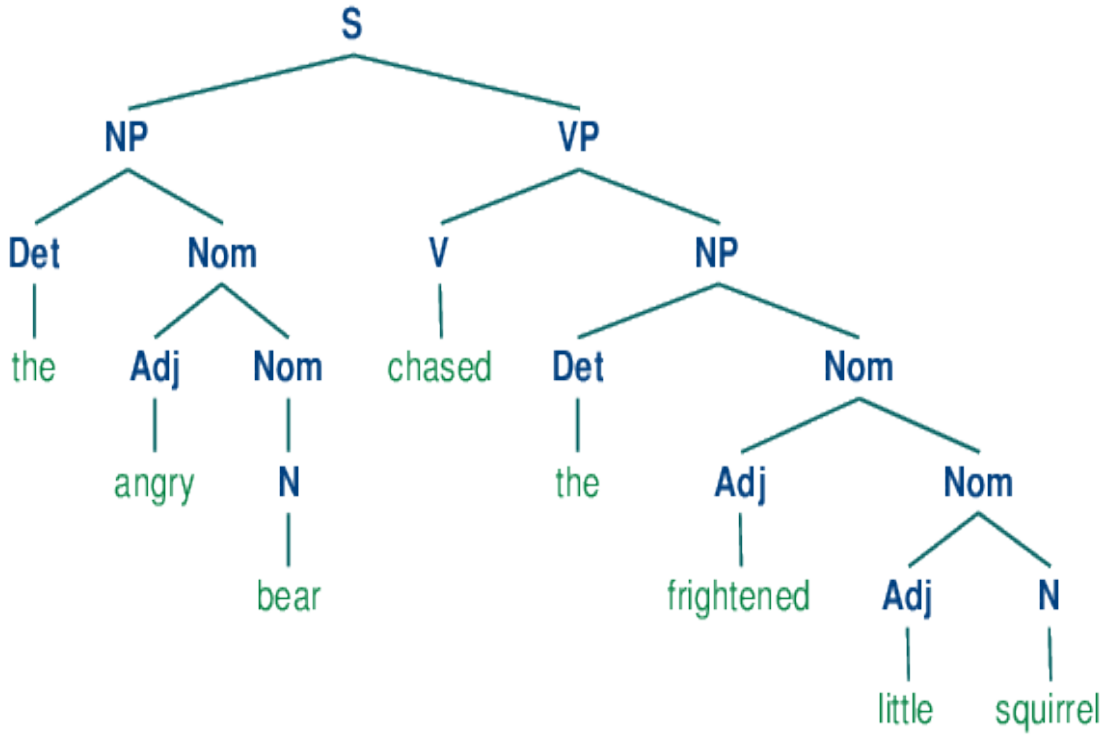


Рис. 1: Пример рекурсивности в синтаксической структуре)

2.2. PyMorphy2

Прежде чем мы начнем обсуждать, что такое pymorphy2, давайте обсудим морфологический анализ.Морфологический анализ - это анализ внутренних структур слов, поэтому, если мы увидим, что русский и украинский языки имеют богатую морфологию, поэтому, используя морфологический анализ, мы определим состояние наших слов, что это состояние, например, существительное или глагол [5]. Pymorphy2 это морфологический анализатор, который анализирует русские тексты с помощью словаря opencorpora. Алгоритм Pymorphy выполняет морфологическую обработку на основе грамматической характеристики типа (слова, лемматизация), но если слово не существует в словаре, поэтому предиктор в pymorphy2 объединит два алгоритма: 1 - по префиксу 2 - по окончанию слов. В то же время, когда мы анализируем слова, мы столкнемся с несколькими состояниями слов и оценками, затем, выбрав состояние слова с наивысшим баллом Мы можем исправить предложение. [10]. Морфологический анализатор pymorphy2 умеет:

1. приводить слово к нормальной форме (например, “люди -> чело-

век", или "гулял -> гулять").

2. ставить слово в нужную форму. Например, ставить слово во множественное число, менять падеж слова и т.д.

3. возвращать грамматическую информацию о слове (число, род, падеж, часть речи и т.д.)[12]

2.3. Высокоэффективное повышение градиента Дерево решений (LightGBM)

Это один из наиболее распространенных алгоритмов искусственного интеллекта, который использовался в многоклассовой классификации, прогнозировании кликов. В настоящее время перед GBP стоит задача получения информации по каждой функции и их расчета, поскольку это отнимает много времени. Итак, для решения этой проблемы GBPT внедрила два метода: 1 - ГОСС (односторонняя выборка на основе градиента): Итак, для наших экземпляров данных нет предопределенных весов, и мы говорим, что разные экземпляры данных с разным градиентом играют разную роль в вычислениях, и в результате мы можем сказать, что больше информации может быть получено из экземпляров больших данных с большие градиенты. 2 - EFB (эксклюзивный пакет функций) - Обычно в реальных приложениях дополнительные функции создают эффективный подход без потерь. [3].

2.4. LemmInflect

LemmInflect использует словарный подход для лемматизации английских слов и преобразования их в формы, заданные предоставленным пользователем тегом Universal Dependencies или Penn Treebank. Библиотека работает со словами, не входящими в словарный запас (OOV), применяя методы нейронных сетей для классификации словоформ и выбора соответствующих правил морфинга [6].

2.5. NLTK4Russian

Как было написано, проект направлен на создание лингвистического комплекса для анализирования корпусов русскоязычных текстов, основанного на различных методах и алгоритмах в рамках наиболее популярных инструментов современной компьютерной лингвистики (NLTK, Pattern, GenSim и т.д.). Таким образом, в этом проекте они имеют работал над двумя основными задачами:

1. разработка морфологического анализатора для русского языка на основе NLTK и Pymorphy

2. Создание парсера для русского языка на основе категориальной грамматики и парсера, встроенного в nltk

В рамках проекта производится разработка синтаксического анализатора для русского языка. Модуль синтаксического анализа в NLTK позволяет исследователям самостоятельно создавать формальные грамматики различных типов для разных естественных языков и применять их в конкретных целях автоматической обработки текстов. [11].

2.6. Метод повторного использования документации семейств программных продуктов

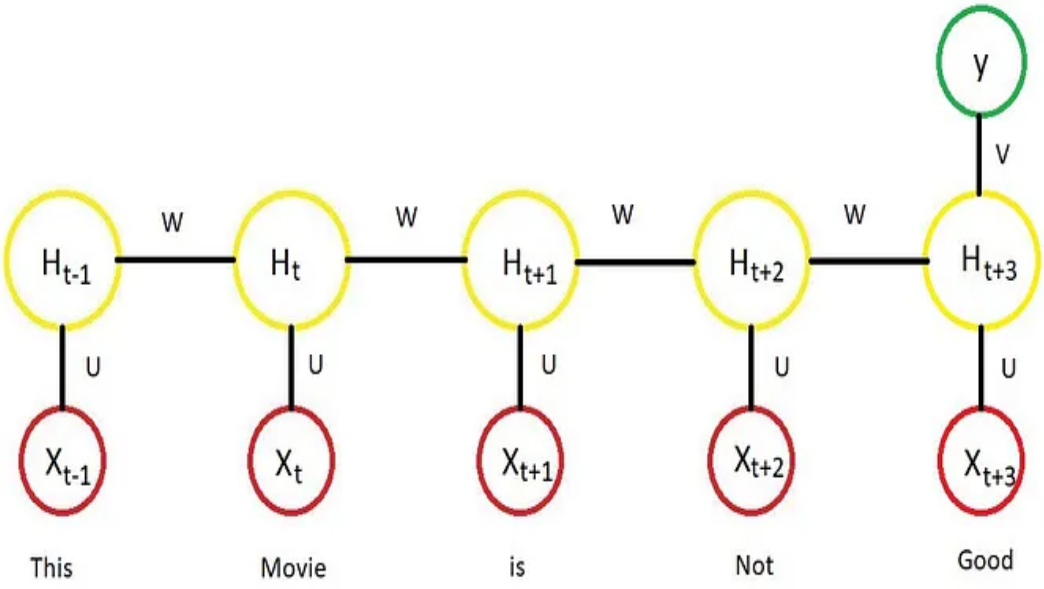
Существуют существующие технологии разработки многоразовой технической документации с акцентом на возможность повторного использования, как указано в диссертации Романовского Константина Юрьевича. Существует два типа повторного использования: случайное и запланированное.

Случайное повторное использование предполагает, что разработчики подключают существующие компоненты, когда и если представится такая возможность. Запланированное повторное использование предполагает систематическую разработку компонентов, подготовленных для повторного использования, и дальнейшее применение таких компонентов в процессе разработки.

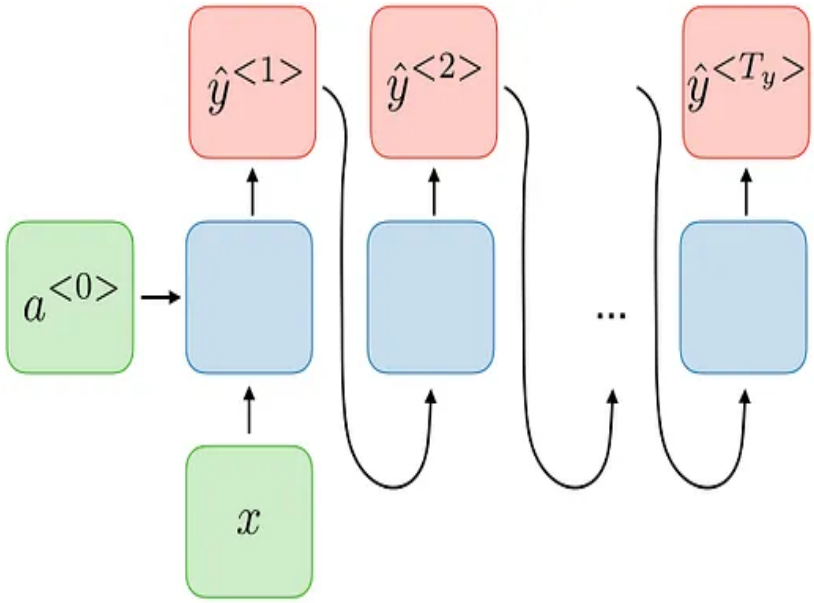
В методе SPP есть два прогрессивных метода: упреждающий и гибкий. Мы сможем начать с нуля в proactive, разработав семейство и добавив в него компоненты. Однако в flexible это похоже на то, как мы начнем использовать продукт, а затем добавим повторно используемые компоненты, но выпуск программного обеспечения отнимает много времени [13].

2.7. Рекуррентная нейронная сеть (RNN)

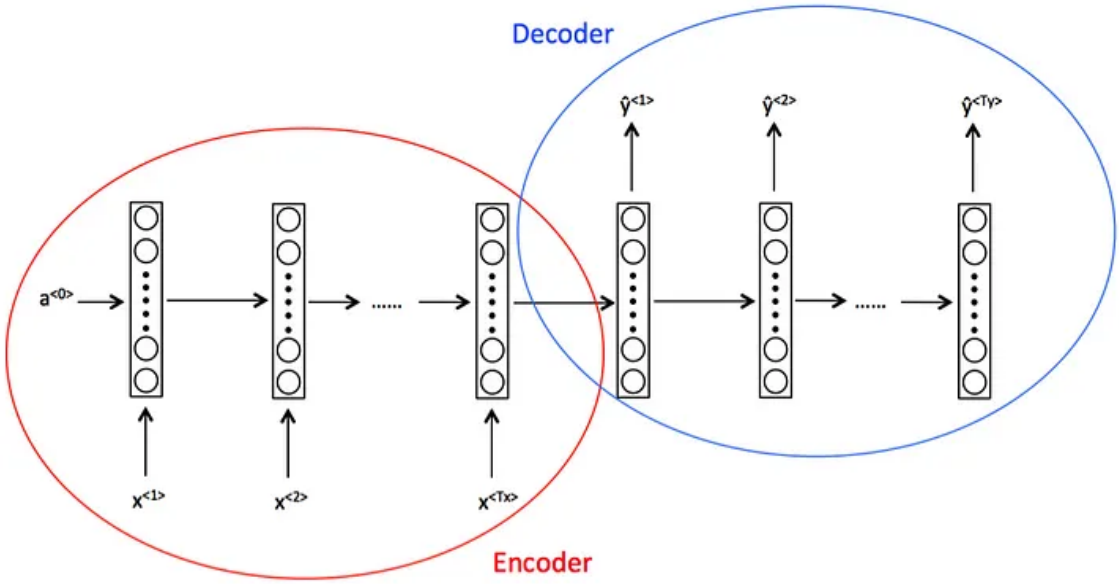
это тип искусственной нейронной сети, которая используется для обработки последовательных данных.Эти алгоритмы глубокого обучения обычно используются, например, для языкового перевода, NLP и т.д.Давайте возьмем выражение, например “плохое самочувствие”, которое означает, что кто-то болен. На этом примере мы объясним работу RNN.Итак, для более глубокого понимания этой идиомы и придания ей смысла, нам нужно выразить идиому в определенном порядке. Используя рекуррентную нейронную сеть, нам нужно обозначить слова, чтобы предсказать следующее слово в последовательности. [4]. Ниже мы поговорим о типе рекуррентной нейронной сети (RNN): 1 - Многие К Одному RNN: это означает, что у нас есть много входов под именем (Тх), но(И) один выход у нас будет под именем (Ту). [9]



2 - Один Ко Многим: Эта архитектура означает, что RNN, которую мы обучили ранее, будет генерировать множество выходных данных на основе одного входного сигнала. [9]



3 - Многие Ко Многим: Эта архитектура означает, что у нас есть много входных данных, и на основе наших входных данных у нас есть много выходные данные. [9]



2.7.1. Преимущество RNN

3 основных преимущества RNN, которые делают RUN лучше других, заключаются в следующем:

1. Последовательная обработка: это означает, что она(РНН) будет обрабатывать данные последовательно.

2. Гибкость: RNN могут обрабатывать входы и выходы переменной длины, что делает их более гибкими по сравнению с другими моделями.

3. Интерпретируемое: Скрытое состояние RNN может быть интерпретировано как краткое изложение входной последовательности, что облегчает понимание того, как модель делает свои прогнозы.

2.7.2. Недостатки RNN

1. Трудно обрабатывать большую последовательность текста: это означает, что когда у нас будет больше контекстов, промежутков между каждым словом будет больше, и обрабатывать их будет сложно. Например, когда у нас есть предложение “Я вырос во Франции.... Я свободно говорю по-французски”, так что здесь мы, основываясь на прогнозе, будем думать, что следующим словом, вероятно, будет название языка. В случае, если сузить контекст, то пробелы уже становятся настолько большими, что невозможно научиться связывать информацию.

2. Забывание того, что произошло в начале: RNN, конечно, предназначен для сохранения информации с предыдущего шага и прогнозирования на текущем. И если мы знаем, что RNN хранит и распространяет информацию во времени через скрытое состояние, то оно будет обновляться на каждом временном шаге. Таким образом, в скрытом состоянии информация будет передаваться с прошлых шагов на текущие, и когда промежутки становятся

больше, этот процесс усложняется, что мы также можем назвать проблемой исчезающего градиента.

3. Трудно поддается обучению: Когда мы говорим о том, что RNN трудно поддаются обучению, это означает, что они не являются нейронными сетями прямой связи. Нейронная сеть с прямой связью подаст сигнал о перемещении только в одном направлении, и этот сигнал будет перемещаться от входных слоев к различным скрытым слоям, а затем перенаправляться на выход системы.

4. Трудно распараллелить

2.8. Трансформаторы (Transformers)

Transformers - это архитектура нейронной сети, которая отслеживает взаимосвязь последовательных данных и преобразует одну последовательность в другую с помощью кодера и декодера. Инновации, лежащие в основе трансформаторов, сводятся к трем основным концепциям:

1. Позиционное кодирование (Positional Encoding)

2. Внимание (Attention)

3. Внутреннее Внимание (Self Attention)

2.8.1. Позиционное кодирование

При позиционном кодировании мы сосредоточимся на двух основных моментах: 1 – кодировщик, 2- декодер.

1. Кодировщик (Encoder):Кодер состоит из стопки из N = 6 идентичных слоев. Каждый слой состоит из двух подслоев. Первый - это механизм саморегулирования с несколькими головками, а второй - простая, ориентированная на местоположение, полностью подключенная сеть прямой связи. Мы используем остаточное соединение вокруг каждого из двух подслоев с последующей нормализацией слоя. То есть выход каждого подслоя - это норма уровня (x

+ Подслой(x)), где подслой(x) - это функция, реализуемая самим подслоем. Чтобы облегчить эти остаточные соединения, все подслои в модели, а также слои внедрения выдают выходные данные размерности $d_{model} = 512$

2. Декодер (Decoder): Декодер также состоит из стека из $N = 6$ идентичных слоев. В дополнение к двум подуровням в каждом слое кодера декодер вставляет третий подуровень, который выполняет многоголовочную обработку выходных данных стека кодера. Аналогично кодировщику, мы используем остаточные соединения вокруг каждого из подуровней с последующей нормализацией уровня. Мы также модифицируем подуровень self-attention в стеке декодера, чтобы предотвратить переключение позиций на последующие. Эта маскировка в сочетании с тем фактом, что вложения выходных данных смещены на одну позицию, гарантирует, что предсказания для позиции i могут зависеть только от известных выходных данных в позициях, меньших, чем i.

2.8.2. Внимание (Attention)

Раньше в RNN мы помещали всю информацию в одно скрытое состояние, и это была своего рода seq2seq. Но, во внимание (Attention), не вся информация будет сохранена в одном скрытом состоянии, но каждое слово, основанное на соответствующем, будет иметь свое собственное скрытое состояние, и это поможет декодеру просто расшифровать его. Итак, идея, лежащая в основе этой теории, заключается в том, что для того, чтобы найти соответствующую информацию в наших предложениях, мы должны использовать эту теорию для того, что называется вниманием. [2]

2.8.3. Внутреннее Внимание

Модуль Внутреннего внимания принимает n входных данных и возвращает n выходных данных. Что происходит в этом модуле? С точки зрения непрофессионала, механизм внутреннего внимания позволяет

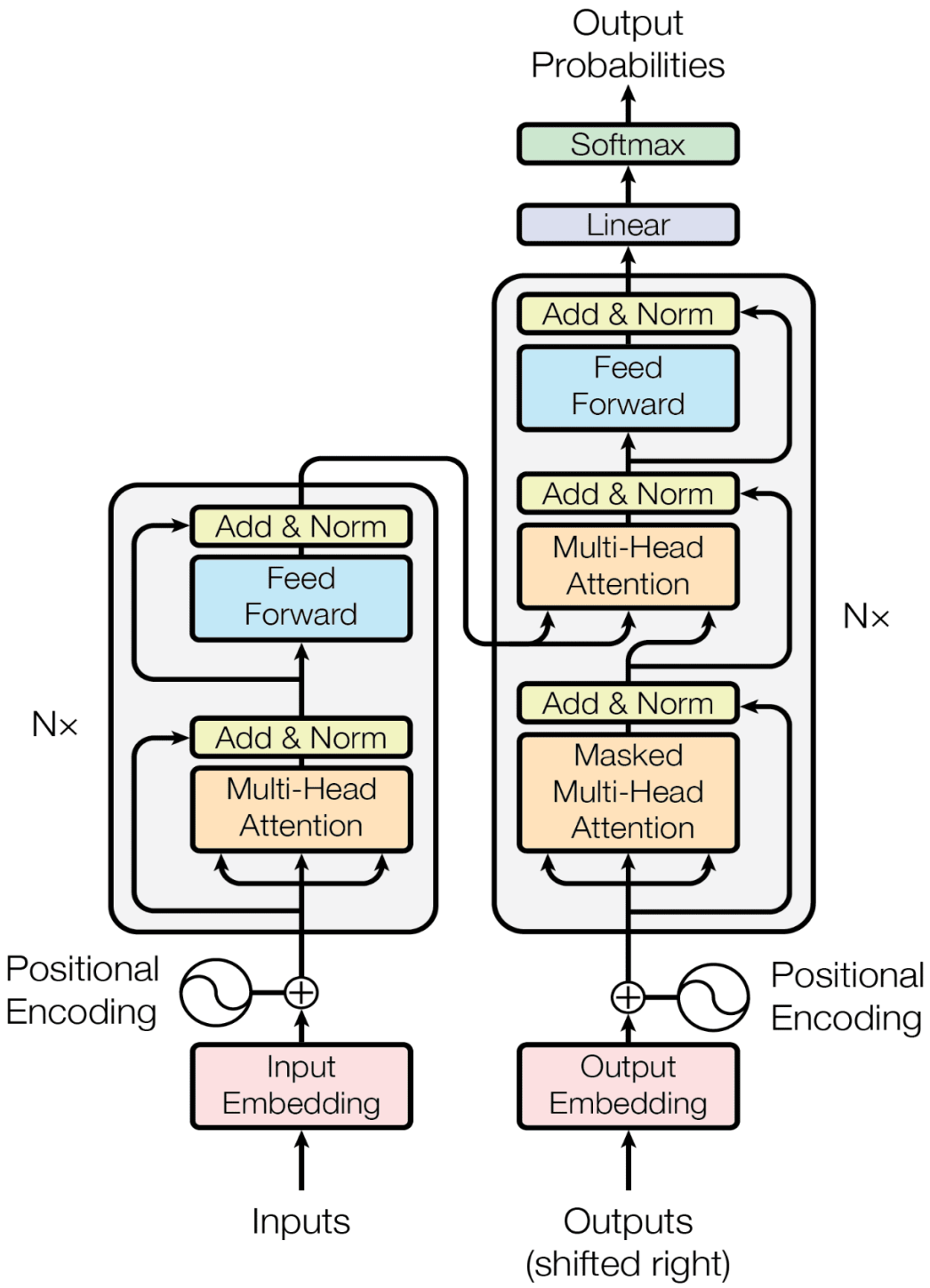
входным данным взаимодействовать друг с другом (“я”) и выяснять, на кого им следует обратить больше внимания (“attention”). Выходные данные представляют собой совокупность этих взаимодействий и показателей внимания. [2] Во внутреннем внимании у нас есть три основных момента:

1. Ключ
2. Запрос
3. Значение

Таким образом, вес для ключа, запроса и значения должен быть умножен. Поэтому мы должны делать это для каждого входного сигнала, который у нас есть. После того, как мы проделаем эту операцию, с помощью softmax мы узнаем эти максимальные значения входных данных.

2.8.4. Преимущества трансформаторов

1. Параллельная обработка: Как мы уже говорили о параллельной обработке, RNN не может распараллеливать процессы, но трансформаторы могут легко это делать.
2. Долгосрочные зависимости: Долгосрочная зависимость, как мы обсуждали для RNN, заключается в том, что когда расстояние становится больше, RNN не может запомнить предыдущую информацию на текущем шаге. Но, к счастью, эта проблема решается трансформерами.
3. Механизм внимания (Attention Mechanism): эта функция является одним из главных преимуществ Transformers, поскольку вся информация будет сохранена в одном скрытом состоянии, и каждое слово будет иметь свое собственное скрытое состояние, основанное на их соответствующем. [1]


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
N×
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
N×
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

2.9. Представления двунаправленного кодера от трансформаторов (BERT)

BERT, что расшифровывается как двунаправленные представления кодера от Transformers. BERT предназначен для предварительной обработки глубоких двунаправленных представлений из немаркированного текста путем совместного определения как левого, так и правого контекста на всех слоях.Существуют две существующие стратегии применения предтренингового представительства в Bert:

1. Предварительная тренировка (Feature-Based)
2. Тонкая настройка (Fine-Tuned)

2.9.1. Предварительная тренировка Берта

мы не используем традиционные языковые модели слева направо или справа налево для предварительного обучения Bert. Вместо этого мы предварительно обучаем Bert, используя две неконтролируемые задачи, описанные в этом разделе.

1. Замаскированный LM (Masked LM): Чтобы обучить глубокое двунаправленное представление, мы просто маскируем некоторый процент входных токенов случайным образом, а затем прогнозируем эти замаскированные токены. Мы называем эту процедуру "замаскированным фильмом" (MLM), хотя в литературе ее часто называют задачей Клоуза. В этом случае конечные скрытые векторы, соответствующие маркерам маски, передаются в выходной softmax поверх словаря, как в стандартном LM. Во всех наших экспериментах мы случайным образом маскируем 15 процентов всех лексем-словосочетаний в каждой последовательности.'[1]
2. Предсказание следующего предложения (NSP) (Next Sentence Prediction (NSP)): Многие важные последующие задачи, такие как ответы на вопросы (QA) и вывод на естественном языке (NLI), основаны

на понимании взаимосвязи между двумя предложениями, которая напрямую не фиксируется языковым моделированием. Чтобы обучить модель, которая понимает взаимосвязи предложений, мы предварительно готовимся к бинаризованной задаче предсказания следующего предложения, которая может быть тривиально сгенерирована из любого одноязычного корпуса. В частности, при выборе предложений A и B для каждого примера предварительного обучения в 50 процентов случаев B является фактическим следующим предложением, которое следует за A (помечено как IsNext), и в 50 процентов случаев это случайное предложение из корпуса (помечено как NotNext). [1]

2.9.2. Точная настройка Bert(Fine-Tuned (Bert))

Тонкая настройка проста (Fine-Tuned),поскольку механизм внутреннего внимания в Transformer позволяет Bert моделировать множество последующих задач - независимо от того, связаны ли они с отдельным текстом или текстовыми парами - путем замены соответствующих входов и выходов. Для приложений, использующих текстовые пары, общей схемой является независимое кодирование текстовых пар перед применением двунаправленного перекрестного внимания (Attention). Вместо этого Bert использует механизм внутреннего внимания, чтобы объединить эти два этапа, поскольку кодирование объединенной текстовой пары с помощью внутреннего внимания (Self-Attention) эффективно включает двунаправленное перекрестное внимание (Attention) между двумя предложениями. [1]

Заключение

В настоящий момент проведен анализ современных методов с выбором подходящего метода для проверки грамматики, а также выполнена часть реализации системы. Дальнейший план процесса реализования этой системы будет заключаться в доработке других частей системы и сравнении современных методов на практике, чтобы также определить их эффективность.

Список литературы

[1] Aidan N, Ashish, Gomez, Illia, Jakob, Jones, Kaiser, Llion, Lukasz, Niki, Noam, Parmar, Polosukhin, 'Shazeer', Uszkoreit, Vaswani. Attention is all you need. — 2017. — Vol. 30.

[2] Giacaglia Giuliano. How Transformers Work. — Access url - https://towardsdatascience.com/transformers-141e32e69591, Online accessed - 2019.

[3] Guolin Ke, Qi Meng, Qiwei Ye, Taifeng Wang, Thomas Finley, Wei Chen, Weidong Ma. LightGBM: A Highly Efficient Gradient Boosting Decision Tree." Advances in Neural Information Processing Systems 30 (NIPS 2017) - 2015, pp - 3149-3157.

[4] IBM. Что такое рекуррентные нейронные сети? — Access url - https://www.ibm.com/topics/recurrent-neural-networks, Online accessed - 2022.

[5] Korobov Mikhail. Morphological analyzer and generator for Russian and Ukrainian languages. — 2015. — P. 320–332.

[6] LemmInflect. LemmInflect. — https://lemminflect.readthedocs.io/en/latest//, Online accessed - 2022.

[7] NLTK. Рекурсия в синтаксической структуре. — Access url - https://www.nltk.org/book/ch08.html, Online accessed - 2022.

[8] NLTK. Синтаксический анализ рекурсивного спуска. — Access url - https://www.nltk.org/book/ch08.html, Online accessed - 2022.

[9] Nigam Vibhor. From Basics to using RNN and LSTM. — Access url - https://medium.com/analytics-vidhya/natural-language-processing-from-basics-to-using-rnn-and-lstm-ef6779e4ae66, Online accessed - 2022.

[10] P. V. Panicheva, A. P. Mirzagitova, E. B. Protopopova, O. A. Mitrofanova. Разработка лингвистического комплекса для морфологического анализа русскоязычных корпусов текстов на основе Pymorphy и NLTK, Труды международной конференции “Корпусная лингвистика”, СПб - 2015, С - 361-373. — 2015.

[11] P. V. Panicheva, E. B. Enikeeva Protopopova, A. P. Mirzagitova, A. D. Moskvina, O. A. Mitrofanova. Проект NLTK4Russian. — Access url - http://mathling.phil.spbu.ru/node/160, Online accessed - 2022.

[12] Pymorphy2. Морфологический анализатор pymorphy2. — Access url - https://pymorphy2.readthedocs.io/en/stable/, Online accessed - 2022.

[13] К.Ю.Романовский. Метод повторного использования документации семейств программных продуктов / К.Ю.Романовский ; СПбГУ. — 2010.

[14] Технопедия. Что означает Natural Language Toolkit (NLTK)? — Access url - https://www.techopedia.com/definition/30343/natural-language-toolkit-nltk, Online accessed - 2022.