Автор: Мельник Геннадий, ПИ21-2

Контакты:

vk: Bue Skim(@genudza),

tg: Gena aka Gиena(@ g1e1n1a1a1k1a1g1i1e1n1a1)

1.     NLP как одна из ведущих областей искусственного интеллекта.

2.     Естественный язык как объект автоматической обработки.

3.     Популярные задачи NLP и общие подходы к их решению.

4.     Предварительная обработка текста. Регулярные выражения.

5.     Стеммеры, лемматизаторы, морфологические анализаторы.

6.     N-граммы. Диструбитиваня гипотеза. Матрица совместной встречаемости.

7.     Применение языковых моделей: предсказание ввода, исправление ошибок правописания.

8.     Проблемы с языковыми моделями и их решения.

9.     Задача ранжирования текстов (Learning to Rank). Метрики близости текстов.

10. Принципы работы модели ELMo - Embeddings from Language Models

11. Принципы работы модели GPT - Generative Pre-Training

12. Большие языковые модели (LLM).

13. Перспективы больших язоковых моделей LLM

14. Классификация текстов: постановка задачи и методы.

15. Классические методы машинного обучения. Проблемы с классификацией текста.

16. Анализ тональности, извлечение аспектов

17. Меры оценки системы NLP.

18.  Рекуррентные нейронные сети в задачах NLP. Основные проблемы.

19. Принципы генерации текстов методами глубокого обучения. Методы жадного поиска и поиска по лучу.

20. Метод векторизации TF-IDF.

21. Технология attention. Принципы работы на примере машинного перевода.

22. Модель sec2sec

23.  Векторные модели GloVe и FastText. Принципы работы.

24. Python как язык программирования и инструмент для написания проектов NLP.

25. Векторная модель word2vec.

26. Принципы работы модели BERT - Bidirectional Encoder Representations from Transformers

27. Машинное обучение в НЛП.

Оглавление

[1.     NLP как одна из ведущих областей искусственного интеллекта. 3](#_Toc154884272)

[2. Естественный язык как объект автоматической обработки. 5](#_Toc154884273)

[3. Популярные задачи NLP и общие подходы к их решению. 6](#_Toc154884274)

[4.     Предварительная обработка текста. Регулярные выражения. 10](#_Toc154884275)

[5.     Стеммеры, лемматизаторы, морфологические анализаторы. 14](#_Toc154884276)

[6.     N-граммы. Диструбитиваня гипотеза. Матрица совместной встречаемости. 16](#_Toc154884277)

[7.     Применение языковых моделей: предсказание ввода, исправление ошибок правописания. 19](#_Toc154884278)

[8.     Проблемы с языковыми моделями и их решения. 20](#_Toc154884279)

[9.     Задача ранжирования текстов (Learning to Rank). Метрики близости текстов. 22](#_Toc154884280)

[10. Принципы работы модели ELMo - Embeddings from Language Models 23](#_Toc154884281)

[11. Принципы работы модели GPT - Generative Pre-Training 25](#_Toc154884282)

[12. Большие языковые модели (LLM). 29](#_Toc154884283)

[13. Перспективы больших языковых моделей LLM. 31](#_Toc154884284)

[14. Классификация текстов: постановка задачи и методы. 32](#_Toc154884285)

[15. Классические методы машинного обучения. Проблемы с классификацией текста. 36](#_Toc154884286)

[16. Анализ тональности, извлечение аспектов 42](#_Toc154884287)

# 1.     NLP как одна из ведущих областей искусственного интеллекта.

Natural Language Processing (NLP) - это область компьютерных наук, которая занимается разработкой методов для работы с естественным языком, который используется людьми для общения. Она объединяет в себе множество технологий, методов и алгоритмов, которые позволяют компьютерам понимать, анализировать и генерировать естественный язык.

Суть NLP заключается в том, чтобы научить компьютер понимать и обрабатывать естественный язык, который является достаточно сложным для машинного понимания. Ключевые задачи NLP включают в себя распознавание речи, классификацию текста, извлечение информации, анализ тональности, машинный перевод и генерацию текста.

Для решения этих задач NLP использует различные методы и алгоритмы, такие как статистический анализ, машинное обучение, нейронные сети и другие. Эти методы позволяют создавать приложения и системы, которые способны обрабатывать большие объемы естественного языка, что может быть полезно для многих областей, включая бизнес, медицину, образование, науку и другие.

В последние годы в NLP произошли значительные улучшения: модели стали лучше понимать человеческий язык и обрабатывать большие объемы текстовых данных. Это привело к улучшению языкового перевода, анализа настроений и систем ответов на вопросы.

Примеры отраслей, где используется nlp

* **Информационные технологии и компьютерные науки**

NLP широко используется в разработке программного обеспечения, создании чат-ботов, анализе данных и обработке текстов.

* **Банковское дело и финансы**

NLP применяется для анализа текстовых данных, таких как отчеты о финансовых результатах и новости, а также для разработки инструментов анализа рисков и управления портфелем.

* **Медицина и науки о здоровье**

NLP используется для анализа текстовых данных, таких как медицинские записи и научные статьи, а также для разработки инструментов диагностики и лечения различных заболеваний.

* **Маркетинг и реклама**

NLP применяется для анализа текстовых данных, таких как отзывы клиентов, комментарии в социальных сетях и обзоры продуктов, а также для создания персонализированных рекламных кампаний.

* **Образование**

NLP используется для создания инструментов автоматической оценки и анализа студенческих работ, а также для создания персонализированных образовательных материалов.

* **Право и юриспруденция**

NLP применяется для анализа юридических документов, таких как договоры и законы, а также для создания инструментов автоматического рассмотрения дел и принятия решений.

* **Государственное управление**

NLP используется для анализа текстовых данных, таких как государственные документы и новости, а также для создания инструментов автоматической обработки запросов от граждан.

# 2. Естественный язык как объект автоматической обработки.

автоматическая обработка текстов (Natural Language Processing) и компьютерная лингвистика (Computational Linguistics).

Естественный язык как объект автоматической обработки – это область ии, направленная на обработку и понимание человеческого языка компьютерами. Это включает в себя различные задачи, начиная от генерации текстов, анализа данных до синтеза речи и машинного перевода.

Одно из ключевых применений NLP — это создание чат-ботов, которые могут взаимодействовать с пользователем, отвечая на вопросы или генерируя тексты на различные темы. Примерами таких систем являются ChatGPT от OpenAI, GigaChat от Сбера и YandexGPT от Яндекса.

# 3. Популярные задачи NLP и общие подходы к их решению.

#перевод англоязычной статьи(Two minutes NLP — 33 important NLP tasks explained) через chatgpt

Классификация

Текстовая классификация: присвоение категории предложению или документу (например, фильтрация спама).

Анализ тональности: определение полярности текста.

Извлечение информации и ранжирование документов

Сходство предложений/документов: определение степени схожести двух текстов.

Ответы на вопросы: задача ответа на вопросы естественным языком.

Генерация текста из текста(text to text)

Машинный перевод: перевод с одного языка на другой.

Генерация текста: создание текста, неотличимого от написанного человеком.

Суммирование текста: создание укороченной версии нескольких документов, сохраняя большую часть их смысла.

Упрощение текста: упрощение текста для удобства чтения и понимания, сохраняя основные идеи и приблизительный смысл.

Лексическая нормализация: перевод/преобразование нестандартного текста в стандартный регистр.

Генерация парафраз: создание выходного предложения, сохраняющего смысл входного, но с вариациями в выборе слов и грамматике.

Базы знаний, сущности и отношения

Извлечение отношений: извлечение семантических отношений из текста. Извлеченные отношения обычно возникают между двумя или более сущностями и относятся к конкретным семантическим категориям (например, живет в, сестра и т.д.).

Прогнозирование отношений: определение названного отношения между двумя именованными семантическими сущностями.

Распознавание именованных сущностей: маркировка сущностей в тексте их соответствующим типом, обычно в нотации BIO.

Связывание сущностей: распознавание и разрешение неоднозначностей именованных сущностей с базой знаний (обычно Wikidata).

Темы и ключевые слова

Моделирование тем: определение абстрактных "тем", лежащих в основе коллекции документов.

Извлечение ключевых слов: определение наиболее релевантных терминов для описания предмета документа.

Чат-боты

Обнаружение намерений: захват семантики сообщений от пользователей и их классификация.

Заполнение слотов: извлечение значений определенных типов атрибутов (или слотов, таких как города или даты) для данной сущности из текстов.

Управление диалогами: управление состоянием и потоком бесед.

Текстовое рассуждение

Рассуждение на основе здравого смысла: использование "здравого смысла" или знаний о мире для выводов.

Логический анализ естественного языка: определение, верна ли "гипотеза" (следствие), ложна (противоречие) или неопределенна (нейтральна) при заданной "премиссе".

Обнаружение фейковых новостей и хейт спича

Обнаружение фейковых новостей: обнаружение и фильтрация текстов с ложной и вводящей в заблуждение информацией.

Обнаружение позиции: определение реакции индивида на утверждение основного актора. Это важная часть набора подходов к оценке фейковых новостей.

Обнаружение речи ненависти: определение, содержит ли текст речь ненависти.

Текст-в-данные и наоборот

Текст в речь: технология, которая озвучивает цифровой текст.

Речь в текст: преобразование речи в текст.

Текст в изображение: генерация фотореалистичных изображений, семантически соответствующих текстовым описаниям.

Данные в текст: создание текста из нелингвистического ввода, такого как базы данных записей, таблицы и базы знаний экспертных систем.

Предварительная обработка текста

Разрешение кореференции: группировка упоминаний в тексте, относящихся к одним и тем же реальным объектам мира.

Морфологическая разметка: разметка слова в тексте его частью речи. Часть речи - это категория слов с похожими грамматическими свойствами, такими как существительное, глагол, прилагательное, наречие, местоимение, предлог, союз и т.д.

Разрешение лексической многозначности: связывание слов в контексте с их наиболее подходящим значением в предопределенном словаре смыслов (обычно WordNet).

Исправление грамматических ошибок: исправление различных видов ошибок в тексте, таких как орфографические, пунктуационные, грамматические и ошибки в выборе слов.

Извлечение признаков: извлечение общих числовых признаков из текста, обычно вложений.

Конец

**Машинное обучение**

1. **Обучение с учителем**: Используется для классификации текстов, анализа тональности, распознавания именованных сущностей и других задач. Модели обучаются на размеченных данных, где каждому примеру соответствует метка.
2. **Обучение без учителя**: Применяется для моделирования тем и извлечения ключевых слов, где структура данных изучается без явных меток.
3. **Обучение с частичным привлечением учителя**: Используется в сценариях, где доступна лишь ограниченная размеченная выборка.

**Глубокое обучение**

1. **Нейронные сети**: Сюда входят свёрточные нейронные сети (CNN) для анализа текстов и рекуррентные нейронные сети (RNN), включая LSTM и GRU, для последовательностей.
2. **Трансформеры**: Мощная архитектура для многих задач, включая машинный перевод, генерацию текста, ответы на вопросы. Особенностью является механизм внимания.

**Лингвистические подходы**

1. **Морфологический анализ**: Используется для морфологической разметки, включая стемминг и лемматизацию.
2. **Синтаксический анализ**: Разбор структуры предложений для извлечения грамматических отношений, важен для кореференции и разрешения лексической многозначности.
3. **Семантический анализ**: Используется для понимания смысла слов в контексте, важен для рассуждения на основе здравого смысла и логического анализа.

**Специализированные инструменты и техники**

1. **Предварительно обученные модели языка**: Такие как BERT, GPT, Transformer-XL, которые обучаются на больших текстовых корпусах и затем могут быть дообучены для конкретных задач.
2. **Обработка специфических запросов**: В чат-ботах, например, используются методы обнаружения намерений и заполнения слотов.
3. **Комбинированные подходы**: Например, в задачах обнаружения фейковых новостей и речи ненависти часто используются сочетания различных методов классификации и верификации фактов.
4. **Кросс-модальные подходы**: В задачах типа "текст в изображение" и "текст в речь" используются сети, способные обрабатывать как текстовые, так и нелингвистические данные.

**Предварительная обработка данных**

1. **Чистка данных**: Удаление шума, нормализация текста.
2. **Извлечение признаков**: Преобразование текста в числовые векторы (вложения слов).
3. **Сегментация текста**: Разбиение на более мелкие единицы, такие как предложения и слова.

# 4.     Предварительная обработка текста. Регулярные выражения.

Предварительная обработка текста - одна из самых важных задач в обработке естественного языка (NLP). Например, вы можете удалить все знаки препинания из текстовых документов, прежде чем их можно будет использовать для классификации текста. Точно так же вы можете извлечь числа из текстовой строки. Написание ручных сценариев для таких задач предварительной обработки требует больших усилий и подвержено ошибкам. Принимая во внимание важность этих задач предварительной обработки, регулярные выражения (также известные как Regex) были разработаны на разных языках, чтобы упростить эти задачи предварительной обработки текста.

Регулярное выражение - это текстовая строка, описывающая шаблон поиска, который можно использовать для сопоставления или замены шаблонов внутри строки с минимальным объемом кода. В этом руководстве мы реализуем различные типы регулярных выражений на языке Python.

Для реализации регулярных выражений можно использовать пакет re Импортируйте re с помощью следующей команды:

import re

**Поиск шаблонов в строке**

Одна из наиболее распространенных задач НЛП - поиск, содержит ли строка определенный шаблон или нет. Например, вы можете захотеть выполнить операцию со строкой при условии, что строка содержит число.

Для поиска шаблона в строке findall функция match и findall пакета re

**Функция соответствия**

Инициализируйте переменный text текстовой строкой следующим образом:

text = "The film Titanic was released in 1998"

Напишем регулярное выражение, которое соответствует строке любой длины и любого символа:

result = re.match(r".\*", text)

Первый параметр match - это выражение регулярного выражения, которое вы хотите найти. Выражение регулярного выражения начинается с алфавита r за которым следует шаблон, который вы хотите найти. Шаблон следует заключать в одинарные или двойные кавычки, как и любую другую строку.

Вышеупомянутое выражение регулярного выражения будет соответствовать текстовой строке, поскольку мы пытаемся сопоставить строку любой длины и любого символа. Если совпадение найдено, match функция возвращает \_sre.SRE\_Match объект , как показано ниже:

type(result)

**Выход:**

\_sre.SRE\_Match

Теперь, чтобы найти совпадающую строку, вы можете использовать следующую команду:

result.group(0)

**Выход:**

'The film Titanic was released in 1998'

В случае, если функция match null объект.

Теперь предыдущее выражение регулярного выражения соответствует строке любой длины и любого символа. Он также будет соответствовать пустой строке нулевой длины. Чтобы проверить это, обновите значение текстовой переменной пустой строкой:

text = ""

Теперь, если вы снова выполните следующее выражение регулярного выражения, совпадение будет найдено:

result = re.match(r".\*", text)

Поскольку мы указали соответствие строки любой длины и любого символа, даже пустая строка будет сопоставлена.

Чтобы сопоставить строку длиной не менее 1, используется следующее выражение регулярного выражения:

result = re.match(r".+", text)

Здесь знак плюса указывает, что в строке должен быть хотя бы один символ.

**Поисковые алфавиты**

Функция match может использоваться для поиска любых букв алфавита в строке. Давайте инициализируем текстовую переменную следующим текстом:

text = "The film Titanic was released in 1998"

Теперь, чтобы найти все буквы алфавита, как в верхнем, так и в нижнем регистре, мы можем использовать следующее выражение регулярного выражения:

result = re.match(r"[a-zA-z]+", text)

В этом регулярном выражении указано, что текстовая строка соответствует любым алфавитам от маленького a до малого z или заглавной A до заглавной Z Знак плюс указывает, что в строке должен быть хотя бы один символ. Напечатаем совпадение, найденное по приведенному выше выражению:

print(result.group(0))

**Выход:**

The

В выходных данных вы можете видеть, что возвращается The Это связано с тем, что match возвращает только первое найденное совпадение. В регулярном выражении мы указали, что находят шаблоны с маленькими и прописными буквами от a до z . Первым найденным совпадением был The . После слова The идет пробел, который не рассматривается как буква алфавита, поэтому сопоставление остановлено, и выражение вернуло только The , которое является первым совпадением.

Однако с этим есть проблема. Если строка начинается с числа, а не с алфавита, match вернет null, даже если после числа есть алфавиты. Давайте посмотрим на это в действии:

text = "1998 was the year when the film titanic was released"

result = re.match(r"[a-zA-z]+", text)

type(result)

**Выход:**

NoneType

В приведенном выше скрипте мы обновили текстовую переменную, и теперь она начинается с цифры. Затем мы использовали match для поиска алфавитов в строке. Хотя текстовая строка содержит алфавиты, будет возвращено значение null, поскольку match только первому элементу в строке.

Чтобы решить эту проблему, мы можем использовать функцию search

# 5.     Стеммеры, лемматизаторы, морфологические анализаторы.

3. Лемматизация и стемминг текста

Обычно тексты содержат разные грамматические формы одного и того же слова, а также могут встречаться однокоренные слова. Лемматизация и стемминг преследуют цель привести все встречающиеся словоформы к одной, нормальной словарной форме.  
  
**Примеры:**  
  
Приведение разных словоформ к одной:

dog, dogs, dog’s, dogs’ => dog

То же самое, но уже применительно к целому предложению:

the boy’s dogs are different sizes => the boy dog be differ size

Лемматизация и стемминг – это частные случаи нормализации и они отличаются.  
  
Стемминг – это грубый эвристический процесс, который отрезает «лишнее» от корня слов, часто это приводит к потере словообразовательных суффиксов.  
  
Лемматизация – это более тонкий процесс, который использует словарь и морфологический анализ, чтобы в итоге привести слово к его канонической форме – лемме.  
  
Отличие в том, что стеммер (конкретная реализация алгоритма стемминга – прим.переводчика) действует без знания контекста и, соответственно, не понимает разницу между словами, которые имеют разный смысл в зависимости от части речи. Однако у стеммеров есть и свои преимущества: их проще внедрить и они работают быстрее. Плюс, более низкая «аккуратность» может не иметь значения в некоторых случаях.  
  
**Примеры:**

1. Слово good – это лемма для слова better. Стеммер не увидит эту связь, так как здесь нужно сверяться со словарем.
2. Слово play – это базовая форма слова playing. Тут справятся и стемминг, и лемматизация.
3. Слово meeting может быть как нормальной формой существительного, так и формой глагола to meet, в зависимости от контекста. В отличие от стемминга, лемматизация попробует выбрать правильную лемму, опираясь на контекст.

Теперь, когда мы знаем, в чем разница, давайте рассмотрим пример:

Вывод:

Stemmer: seen

Lemmatizer: see

Stemmer: drove

Lemmatizer: drive

морфологические анализаторы:

* **Pymorphy2**: Широко используемый анализатор для русского языка.
* **NLTK**: Библиотека для Python, включающая инструменты для морфологического анализа.
* **SpaCy**: Ещё одна библиотека для Python, предлагающая мощные инструменты для NLP, включая морфологический анализ.

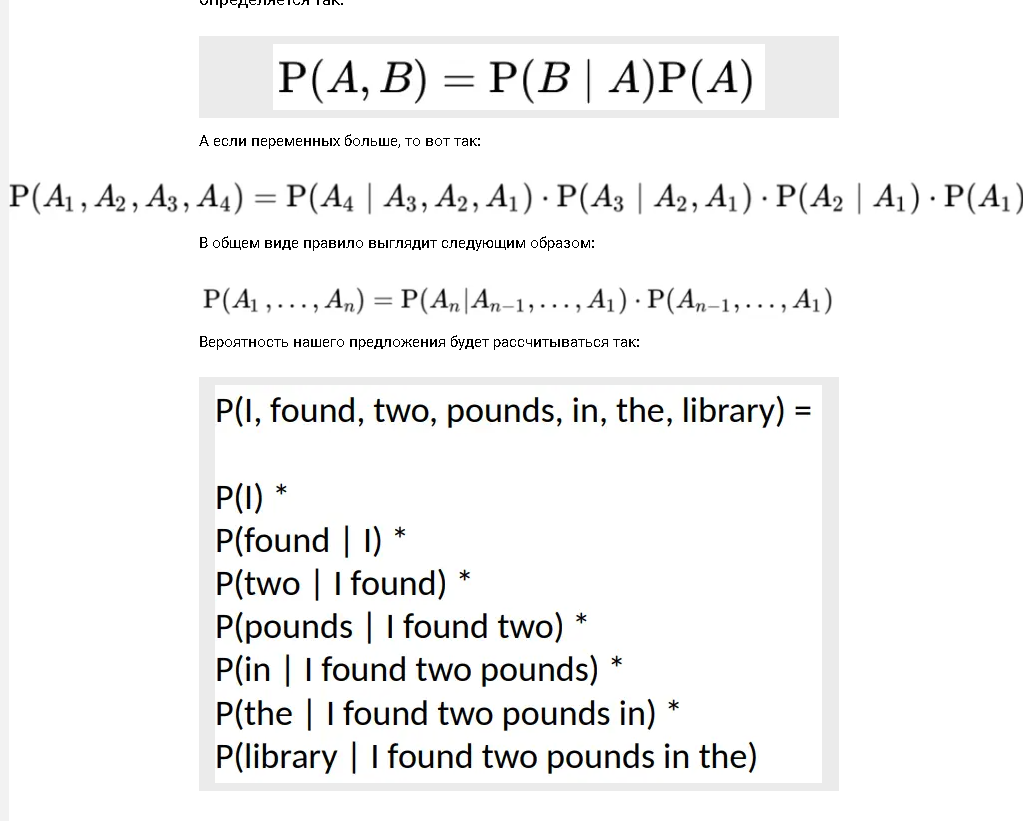
# 6.     N-граммы. Диструбитиваня гипотеза. Матрица совместной встречаемости.

N-граммы

N-граммы – это статистические модели, которые предсказывают следующее слово после N-1 слов на основе вероятности их сочетания. Например, сочетание I want to в английском языке имеет высокую вероятностью, а want I to – низкую. Говоря простым языком, N-грамма – это последовательность n слов. Например, биграммы – это последовательности из двух слов (I want, want to, to, go, go to, to the…), триграммы – последовательности из трех слов (I want to, want to go, to go to…) и так далее.

Такие распределения вероятностей имеют широкое применение в машинном переводе, автоматической проверке орфографии, распознавании речи и умном вводе. Например, при распознавании речи, по сравнению с фразой eyes awe of an, последовательность I saw a van будет иметь большую вероятность. Во всех этих случаях мы подсчитываем вероятность следующего слова или последовательности слов. Такие подсчеты называются языковыми моделями.

Как же рассчитать P(w)? Например, вероятность предложения P(I, found, two, pounds, in, the, library). Для этого нам понадобится цепное правило, которое определяется так:



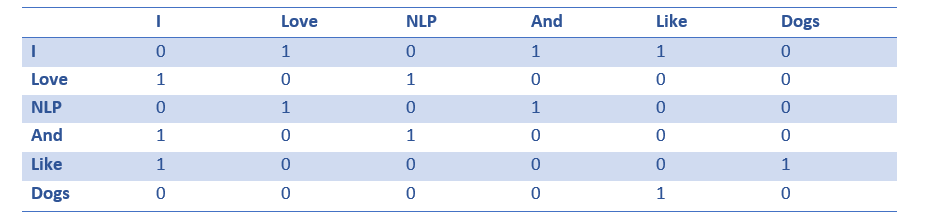
Дистрибутивная гипотеза

Дистрибутивная семантика основывается на дистрибутивной гипотезе: лингвистические единицы, встречающиеся в схожих контекстах, имеют близкие значения.

Психологические эксперименты подтвердили истинность данной гипотезы. Например, в одной из работ участников эксперимента просили высказать своё суждение о синонимичности предъявляемых им пар слов. Данные опроса затем сравнивали с контекстами, в которых встречались изучаемые слова. Эксперимент показал наличие положительной корреляции между семантической близостью слов и схожестью контекстов, в которых они встречаются.

Что-то про матрицу

Матрица совместной встречаемости содержит количество раз, которое каждое слово встретилось в корпусе (обучающем наборе) после каждого другого слова этого корпуса.



# 7.     Применение языковых моделей: предсказание ввода, исправление ошибок правописания.

1. **Предсказание ввода**: Языковые модели, такие как GPT (Generative Pre-trained Transformer), обучаются на большом объеме текстовых данных, что позволяет им эффективно предсказывать следующие слова или фразы в тексте. Это широко используется в системах автозаполнения, например, в почтовых клиентах, мессенджерах и текстовых редакторах. Модель анализирует контекст ввода пользователя и предлагает наиболее вероятное продолжение.
2. **Исправление ошибок правописания**: Языковые модели также могут быть обучены на распознавание и исправление орфографических и грамматических ошибок. Это достигается за счет анализа больших объемов текстов, где модели учатся распознавать стандартные языковые паттерны и выявлять отклонения от них. Такие системы исправления ошибок интегрированы в текстовые редакторы, электронную почту и другие сервисы, где важно поддерживать высокое качество письменного текста.

# 8.     Проблемы с языковыми моделями и их решения.

Проблемы:

* Фактоиды (галлюцинации) – Модели выдумывают несуществующие факты, дают неверные определения. Это связано с тем, что алгоритм вынужден дать хоть какой-нибудь ответ, даже если pretrained модель ответа не знает. Так генерируются более-менее “похожие на правду”, но искаженные по смыслу ответы.
* Имплементация мультимодальности. Современные ИИ-сервисы стремятся обеспечить мультимодальность – соединить несколько моделей разных типов в одном решении. Например, обрабатывать и выдавать не только текстовые данные, но и генерировать изображения, звук, обрабатывать программный код, табличные данные и проч. Интеграция различных кодеров/декодеров в один сервис – нетривиальная задача для разработчиков.
* Выравнивание и неинтерпретируемость. Под выравниванием в данном случае понимается способность модели верно интерпретировать промпт – выдавать то, что от нее хочет пользователь не только по форме, но и по содержанию.
* Трансформерные модели, зачастую хуже классических умеют решать некоторые задачи. Например, известно, что они практически не могут выполнять арифметические вычисления, лишь запомнить, что “дважды два будет четыре”.
* Проблема композициональности. Некоторые индуктивные зависимости между упоминаемыми сущностями, существующие в языке – очевидные человеку, трансформеру трудно вывести. Моменты генерализации, правил – вообще с трудом обучаемы посредством алгоритма Feed Forward, используемого в блоках Attentions.
* Проблема актуальности данных. Предобученная модель в любом случае обучена на корпусе текстов, опубликованных до известной даты. А новая информация поступает постоянно. Дообучение – дорогостоящий процесс; обновлять LLM-модель регулярно крайне трудоемко.
* Ограниченная длина контекста. Сам механизм MultiHead Attention имеет низкую рекурентность – “физические” ограничения по объему сохраняемого внутренними слоями окна данных. Невозможно учитывать контекст при обработке большого текста или длинной чат-сессии вопрос-ответ . Система начинает “забывать” контекст обработанный ранее, если общий объем входа достаточно велик.
* Отсутствие Тьюринг-полноты и ограниченная рекурентности. Отсутствие у предобученной модели возможности имитировать любой вычислительный процесс или алгоритм. Рекурентность – это свойство, которое позволяет сети использовать информацию, полученную на предыдущих шагах, для принятия решений на текущем шаге. Такая возможность, с известными ограничениями, имелась у предшественников Attentions – RNN типа LSTM или GRU, но в классических трансформерах с этим возникают трудности.
* Высокие требования к вычислительным ресурсам.
* Ограничения режима чата. Интеллектуальные задачи не всегда удобно решать из-за ограничений GUI конкретного чат-бота, скажем, интерфейса Telegram.
* Multi Embodiment – проблемы воплощения. Желание связать вход и выход модели с различными интеллектуальными агентами, возможность работать на разных платформах и в разных прикладных окружениях.
* Открытые вопросы интеграции с унаследованной ИТ-инфраструктурой.

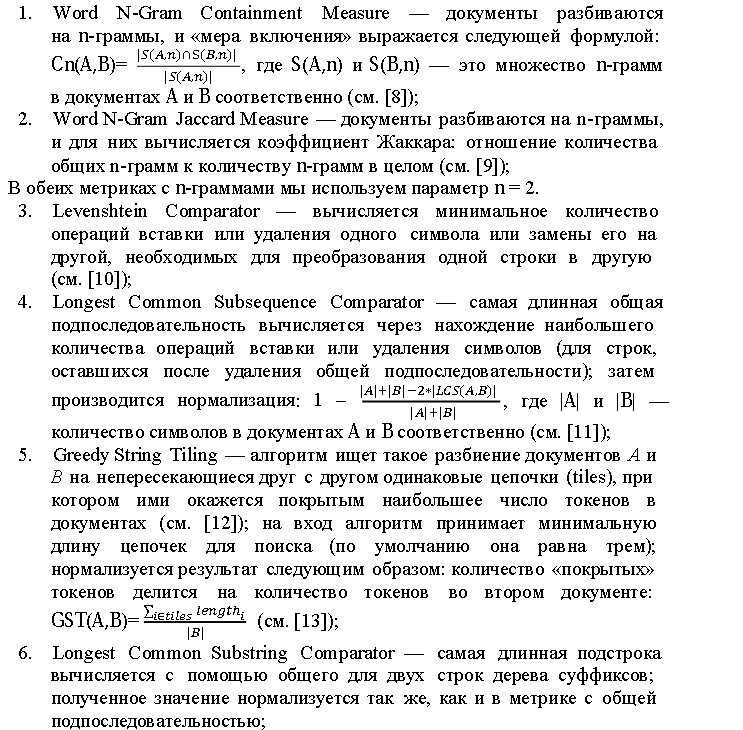
Решение проблем:

1. **Противодействие предвзятости и стереотипам**: Разработка методов фильтрации и коррекции данных обучения, внедрение механизмов критического отношения к собственному контенту.
2. **Борьба с неправильной информацией**: Включение механизмов обновления знаний модели, улучшение способов верификации фактов.
3. **Обеспечение конфиденциальности**: Разработка строгих протоколов обработки данных и анонимизации личной информации.
4. **Улучшение понимания контекста**: Улучшение алгоритмов понимания контекста, введение механизмов для запроса уточнений у пользователя.
5. **Обеспечение безопасности**: Внедрение систем мониторинга, ограничения на генерацию определенных видов контента.
6. **Развитие интерактивности**: Разработка методов интерактивного обучения и адаптации модели в процессе использования.
7. **Оптимизация масштабирования**: Оптимизация архитектуры моделей, поиск эффективных методов обучения.

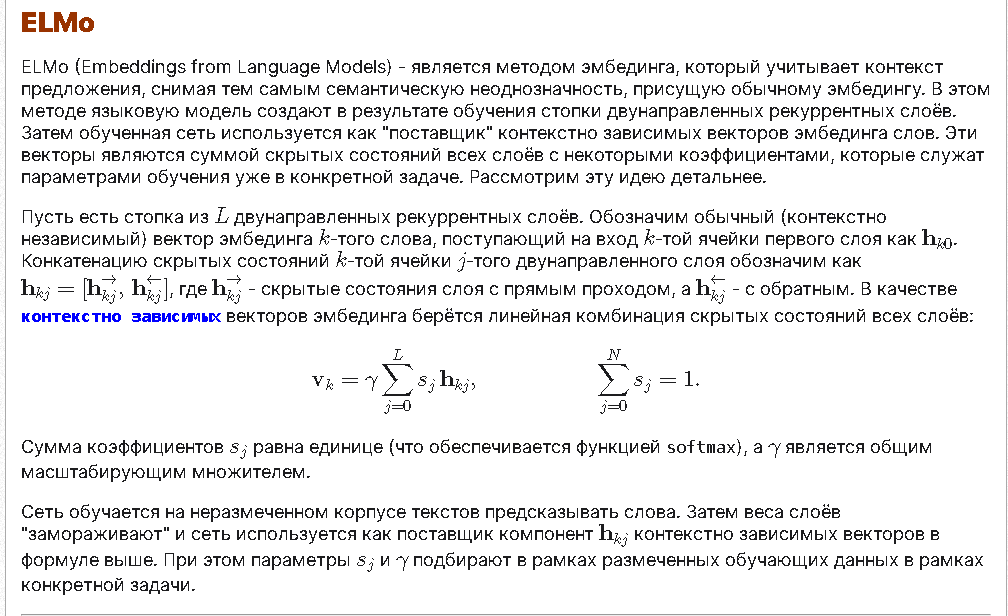
# 9.     Задача ранжирования текстов (Learning to Rank). Метрики близости текстов.

Ранжирование (англ. learning to rank) — это класс задач машинного обучения с учителем, заключающихся в автоматическом подборе ранжирующей модели по обучающей выборке, состоящей из множества списков и заданных частичных порядков на элементах внутри каждого списка. Частичный порядок обычно задаётся путём указания оценки для каждого элемента (например, «релевантен» или «не релевантен»). Цель ранжирующей модели — наилучшим образом приблизить и обобщить способ ранжирования в обучающей выборке на новые данные.

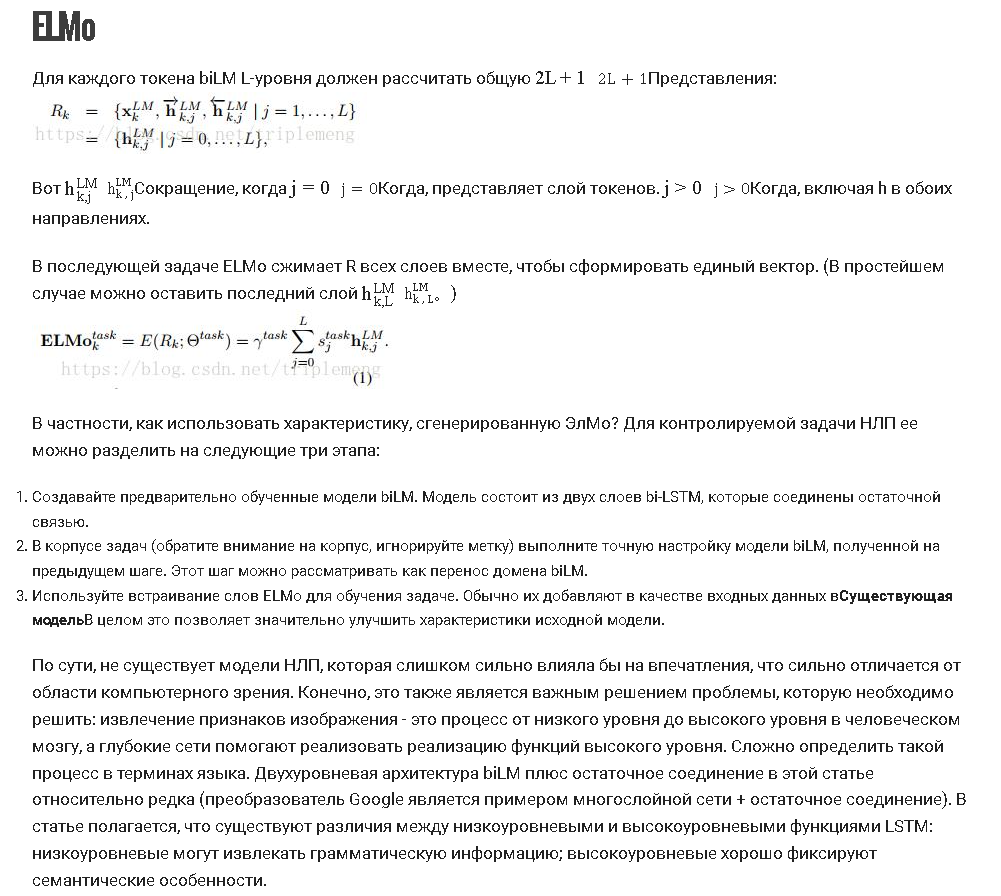
Метрики семантичесской близости



# 10. Принципы работы модели ELMo - Embeddings from Language Models



Альтернатива



Вариант от chatgpt

Модель ELMo (Embeddings from Language Models) является одним из ключевых прорывов в области обработки естественного языка (NLP). Вот основные принципы ее работы:

1. **Двунаправленная архитектура LSTM**: ELMo использует двунаправленную архитектуру LSTM (Long Short-Term Memory), что позволяет модели обрабатывать текст как в прямом, так и в обратном направлениях. Это означает, что для каждого слова модель учитывает контекст как до, так и после этого слова.
2. **Контекстно-зависимые векторные представления**: В отличие от более ранних подходов, таких как word2vec или GloVe, которые генерируют одинаковое представление для каждого слова независимо от контекста, ELMo создает векторы, которые динамически меняются в зависимости от контекста слова в предложении. Это означает, что одно и то же слово может иметь разные представления в зависимости от его использования.
3. **Глубокое обучение**: ELMo обучается на большом корпусе текста для генерации этих векторных представлений. После обучения ELMo может быть использована для извлечения признаков из текстовых данных, которые затем можно использовать в различных задачах NLP.
4. **Использование в задачах NLP**: После обучения ELMo может быть интегрирована в различные NLP-модели, такие как модели для распознавания именованных сущностей, анализа тональности или классификации текста. ELMo обычно используется для улучшения качества этих моделей, добавляя к ним более богатое и глубокое понимание языка.
5. **Преимущества ELMo**: Ключевым преимуществом ELMo является ее способность лучше понимать смысловые и синтаксические нюансы языка благодаря учету контекста. Это позволяет достигать более высокой точности в различных задачах обработки естественного языка.

ELMo стала важным шагом на пути к более продвинутым языковым моделям, таким как BERT и GPT, которые развили и расширили идеи, заложенные в ELMo.

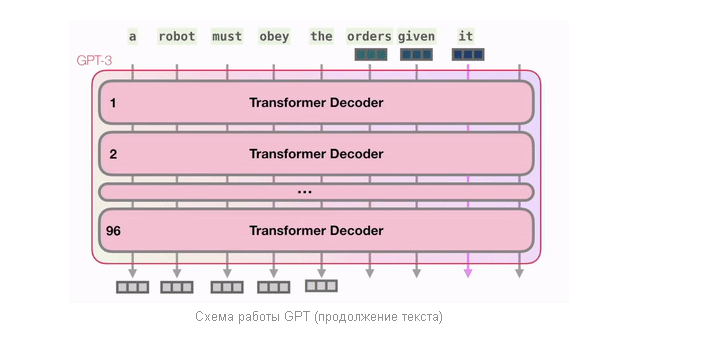
# 11. Принципы работы модели GPT - Generative Pre-Training

GPT для чайников: от токенизации до файнтюнинга(тут я использовал куски из этой статьи)

Архитектура GPT

Если коротко, то при генерации продолжения текста с помощью GPT происходит следующее:

1. Входной текст токенизируется в последовательность чисел (токенов).
2. Список токенов проходит через Embedding Layer (линейный слой) и превращается в список эмбеддингов (очень похоже на word2vec).
3. К каждому эмбеддингу прибавляется positional embedding, о котором я расскажу в сл. секции.
4. Далее список эмбеддингов начинает своё путешествие через несколько одинаковых блоков (Transformer Decoder Block), о которых я тоже расскажу потом.
5. После того как список эмбеддингов пройдёт через последний блок, эмбеддинг, соответствующий последнему токену матрично умножается на всё тот же входной, но уже транспонированный Embedding Layer и после применения SoftMax получается распределение вероятностей следующего токена.
6. Из этого распределения выбираем следующий токен (например с помощью функции argmax).
7. Добавляем этот токен к входному тексту и повторяем шаги 1-6.

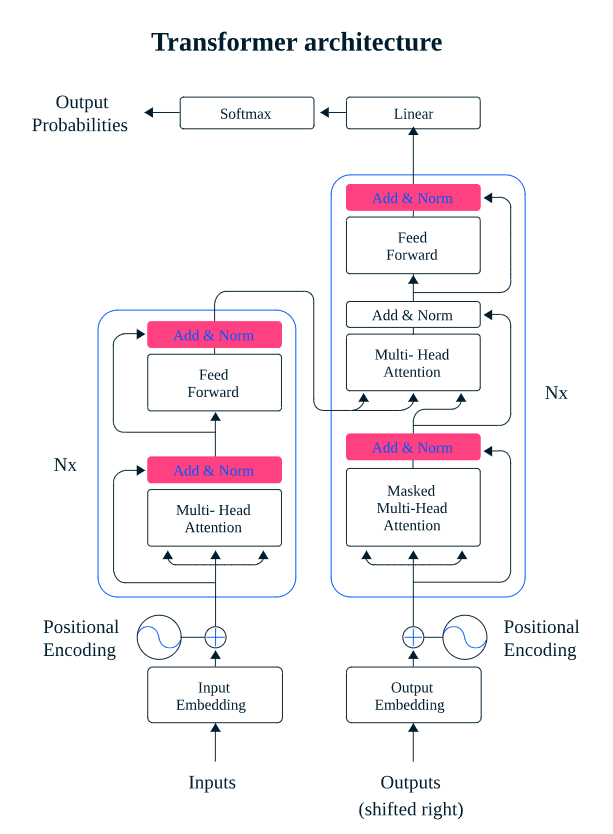


Альтернатива

**Как построить модель GPT? (neuroni.co)**

**Рабочий механизм моделей GPT**

GPT — это языковая модель ИИ, основанная на архитектуре преобразователя, которая является предварительно обученной, генеративной, неконтролируемой и способной хорошо работать в многозадачных условиях с нулевым/одним/несколько выстрелов. Он предсказывает следующий токен (экземпляр последовательности символов) из последовательности токенов для задач NLP, на которых не обучался. Увидев всего несколько примеров, он может достичь желаемых результатов в определенных тестах, включая машинный перевод, вопросы и ответы и задачи закрытия. Модели GPT рассчитывают вероятность появления слова в тексте при условии, что оно появляется в другом тексте, в основном на основе условной вероятности. Например, в предложении «Маргарет организует гаражную распродажу… Возможно, мы могли бы купить этот старый…» слово «стул» более подходит, чем слово «слон». Также, Модели-трансформеры используют несколько единиц, называемых блоками внимания, которые узнают, на каких частях текстовой последовательности нужно сосредоточиться. У одного трансформера может быть несколько блоков внимания, каждый из которых изучает разные аспекты языка.



Архитектура преобразователя имеет два основных сегмента: кодировщик, который в основном работает с входной последовательностью, и декодер, который работает с целевой последовательностью во время обучения и предсказывает следующий элемент. Например, преобразователь может взять последовательность английских слов и предсказать французское слово в правильном переводе, пока он не будет завершен.

Кодер определяет, какие части ввода следует выделить. Например, кодировщик может прочитать такое предложение, как «Быстрая коричневая лиса прыгнула». Затем он вычисляет матрицу встраивания (встраивание в NLP позволяет словам с похожим значением иметь одинаковое представление) и преобразует ее в серию векторов внимания. Итак, что такое вектор внимания? Вы можете рассматривать вектор внимания в модели-трансформере как специальный калькулятор, который помогает модели понять, какие части той или иной информации наиболее важны для принятия решения. Предположим, вам задали несколько вопросов на экзамене, на которые вы должны ответить, используя разную информацию. Вектор внимания помогает выбрать наиболее важную информацию для ответа на каждый вопрос. Точно так же работает и в случае с моделью-трансформером.

Блок внимания с несколькими головками первоначально создает эти векторы внимания. Затем они нормализуются и передаются в полносвязный слой. Перед передачей в декодер снова выполняется нормализация. Во время обучения кодер работает непосредственно с целевой выходной последовательностью. Предположим, что целевой вывод — это французский перевод английского предложения «Быстрая коричневая лисица прыгнула». Декодер вычисляет отдельные векторы встраивания для каждого французского слова предложения. Дополнительно применяется позиционный энкодер в виде функций синуса и косинуса. Кроме того, используется замаскированное внимание, что означает, что используется первое слово французского предложения, тогда как все остальные слова маскируются. Это позволяет преобразователю научиться предсказывать следующие французские слова.

Наряду с этим модели GPT используют некоторое сжатие данных, потребляя миллионы и миллионы образцов текстов для преобразования слов в векторы, которые представляют собой не что иное, как числовые представления. Затем языковая модель распаковывает сжатый текст в удобные для человека предложения. Точность модели повышается за счет сжатия и распаковки текста. Это также позволяет вычислить условную вероятность каждого слова. Модели GPT могут хорошо работать в настройках «несколько снимков» и реагировать на текстовые образцы, которые вы видели раньше. Им требуется всего несколько примеров, чтобы дать соответствующие ответы, потому что они были обучены на многих образцах текста.

Кроме того, у моделей GPT есть много возможностей, например, создание образцов синтетического текста беспрецедентного качества. Если вы заполните модель входными данными, она сгенерирует длинное продолжение. Модели GPT превосходят другие языковые модели, обученные на таких доменах, как Википедия, новости и книги, без использования данных для обучения, специфичных для предметной области. GPT изучает языковые задачи, такие как понимание прочитанного, обобщение и ответы на вопросы, только из текста, без данных для обучения для конкретных задач. Оценки этих задач («оценка» относится к числовому значению, которое модель присваивает для представления правдоподобия или вероятности данного выхода или результата) не самые лучшие, но они предлагают неконтролируемые методы с достаточным количеством данных и вычислений, которые могут принести пользу задачам.

# 12. Большие языковые модели (LLM).

Большие языковые модели (LLM) – это очень большие модели глубокого обучения, которые предварительно обучены на огромных объемах данных. Лежащий в основе трансформер – это набор нейронных сетей, каждая из которых состоит из кодера и декодера с возможностью самонаблюдения. Кодер и декодер извлекают значения из последовательности текста и понимают отношения между имеющимися в ней словами и фразами.

Трансформеры LLM способны обучаться без наблюдения, хотя точнее будет сказать, что трансформеры осуществляют самообучение. Именно благодаря этому процессу трансформеры учатся понимать базовую грамматику и языки, а также усваивать знания.

В отличие от предыдущих рекуррентных нейронных сетей (RNN), которые последовательно обрабатывают входные данные, трансформеры обрабатывают целые последовательности параллельно. Это позволяет специалистам по обработке данных использовать графические процессоры для обучения LLM на основе трансформеров, что значительно сокращает время обучения.

Архитектура нейронной сети трансформера позволяет использовать очень большие модели, часто с сотнями миллиардов параметров. Такие сверхбольшие модели могут получать огромные объемы данных, часто из Интернета, а также из таких источников, как индекс Common Crawl, насчитывающий более 50 миллиардов веб-страниц, и Википедия, насчитывающая около 57 миллионов страниц.

Альтернатива:

LLM используют комплексный подход, включающий множество компонентов.

На базовом уровне LLM необходимо обучать на большом объеме (иногда называемом корпусом) данных, размер которых обычно составляет петабайты. Обучение может состоять из нескольких этапов, обычно начиная с подхода обучения без присмотра. При таком подходе модель обучается на неструктурированных и немаркированных данных. Преимущество обучения на неразмеченных данных заключается в том, что зачастую доступно гораздо больше данных. На этом этапе модель начинает выявлять связи между различными словами и понятиями.

Следующим шагом для некоторых инструментов LLM является обучение и доработка в форме самостоятельного обучения. Здесь происходит некоторая маркировка данных, помогающая модели более точно идентифицировать различные концепции.

Затем LLM проводит глубокое обучение, проходя процесс преобразования нейронной сети. Архитектура модели преобразователя позволяет LLM понимать и распознавать отношения и связи между словами и понятиями, используя механизм самообслуживания. Этот механизм способен присвоить оценку, обычно называемую весом, данному элементу (называемому токеном) , чтобы определить взаимосвязь.

После обучения LLM создается база, на которой AI можно использовать в практических целях. Запрашивая LLM с помощью подсказки, вывод модели AI может генерировать ответ, который может быть ответом на вопрос, вновь созданным текстом, обобщенным текстом или отчетом анализа настроений.

# 13. Перспективы больших языковых моделей LLM.

1. **Автоматизация и улучшение обслуживания клиентов**: Языковые модели могут использоваться для создания более эффективных и интуитивно понятных чат-ботов, способных обрабатывать запросы клиентов и предоставлять полезную информацию.
2. **Образование и обучение**: Языковые модели могут быть интегрированы в образовательные платформы для предоставления персонализированных рекомендаций по обучению, создания интерактивных обучающих материалов и оказания помощи в изучении языков.
3. **Перевод и локализация**: Способность языковых моделей к обработке и переводу текста на разные языки может значительно ускорить процесс локализации контента для международных рынков.
4. **Создание контента**: От написания статей до генерации маркетинговых материалов, языковые модели могут помочь в создании разнообразного текстового контента, сэкономив время и ресурсы.
5. **Помощь в исследованиях и анализе данных**: Языковые модели могут обрабатывать большие объемы текстовой информации, помогая исследователям и аналитикам получать ценные выводы и упрощая процесс анализа данных.
6. **Интеграция с другими технологиями**: Языковые модели могут работать совместно с другими технологиями, такими как искусственный интеллект и машинное обучение, для создания более комплексных и эффективных систем.
7. **Поддержка творчества**: Языковые модели могут использоваться для генерации креативных текстов, стихов, сценариев и даже для помощи в написании музыки.
8. **Повышение доступности информации**: Для людей с ограниченными возможностями, языковые модели могут облегчить доступ к информации, преобразуя текст в речь или обеспечивая интерактивную поддержку.
9. **Улучшение анализа сентиментов и социальных медиа**: Языковые модели могут анализировать большие объемы данных из социальных сетей для понимания общественных настроений и тенденций.
10. **Этические и социальные вопросы**: Большие языковые модели также ставят вопросы этики и безопасности использования ИИ, требуя разработки стандартов и норм для регулирования их применения.

# 14. Классификация текстов: постановка задачи и методы.

**Цели классификации текстов**

Классификация текстов – это процесс разделения текстов на группы или категории в соответствии с определенными критериями. Цель классификации текстов заключается в том, чтобы упорядочить и организовать большие объемы текстовой информации, чтобы облегчить их анализ и поиск.

**Организация информации**

Классификация текстов позволяет организовать большие объемы информации, разделяя тексты на группы с общими характеристиками. Это помогает упорядочить информацию и сделать ее более доступной для анализа и поиска.

**Поиск и извлечение информации**

Классификация текстов облегчает поиск и извлечение информации из больших объемов текстов. Путем классификации текстов по определенным категориям можно быстро найти нужную информацию и избежать необходимости просматривать все тексты в поисках нужных данных.

**Анализ и исследование текстов**

Классификация текстов позволяет проводить анализ и исследование текстов по определенным критериям. Например, можно классифицировать тексты по жанру, стилю, теме или автору, чтобы изучить различия и сходства между ними и выявить закономерности.

**Автоматическая обработка текстов**

Классификация текстов является важным инструментом в области компьютерной лингвистики и обработки естественного языка. Она позволяет разрабатывать алгоритмы и модели, которые автоматически классифицируют тексты по определенным категориям. Это может быть полезно, например, для автоматической фильтрации спама, определения тональности текстов или категоризации новостей.

В целом, классификация текстов помогает организовать, анализировать и извлекать информацию из текстовой информации, делая ее более доступной и полезной для различных целей.

**Методы классификации текстов**

**Метод наивного Байеса**

Метод наивного Байеса основан на теореме Байеса и предполагает, что все признаки (слова) в тексте независимы друг от друга. Он использует статистические методы для определения вероятности принадлежности текста к определенной категории. Метод наивного Байеса хорошо работает с большими объемами данных и может быть эффективно применен для классификации текстов.

**Метод опорных векторов (SVM)**

Метод опорных векторов (SVM) строит гиперплоскость в многомерном пространстве, которая разделяет тексты разных категорий. Он основан на поиске оптимальной разделяющей гиперплоскости, которая максимизирует расстояние между классами. SVM может быть эффективным методом классификации текстов, особенно когда данные нелинейно разделимы.

**Метод k-ближайших соседей (k-NN)**

Метод k-ближайших соседей (k-NN) классифицирует тексты на основе их близости к другим текстам в обучающем наборе данных. Он определяет класс текста, основываясь на классах его ближайших соседей. Метод k-NN прост в реализации и может быть эффективным для классификации текстов, особенно когда данные имеют простую структуру.

**Методы машинного обучения**

Методы машинного обучения, такие как решающие деревья, случайные леса и градиентный бустинг, также могут быть использованы для классификации текстов. Они основаны на обучении модели на обучающем наборе данных и использовании этой модели для классификации новых текстов. Методы машинного обучения могут быть эффективными для классификации текстов, особенно когда данные имеют сложную структуру и большой объем.

**Глубокое обучение**

Глубокое обучение – это метод машинного обучения, который использует нейронные сети с большим количеством слоев для классификации текстов. Глубокое обучение может автоматически извлекать признаки из текстов и строить сложные модели для классификации. Оно может быть эффективным для классификации текстов, особенно когда данные имеют сложную структуру и большой объем.

Это лишь некоторые из методов классификации текстов, которые могут быть использованы в области компьютерной лингвистики и обработки естественного языка. Каждый метод имеет свои преимущества и ограничения, и выбор метода зависит от конкретной задачи и доступных данных.

**Примеры классификации текстов**

**Классификация текстов по жанру**

Один из наиболее распространенных способов классификации текстов – это их разделение по жанру. Жанр текста определяется его структурой, стилем и тематикой. Некоторые примеры жанров текстов включают новости, научные статьи, романы, поэзию, рекламу и т. д. Классификация текстов по жанру может быть полезна для организации и поиска текстов в больших коллекциях, а также для анализа и понимания различных стилей и тематик текстов.

**Классификация текстов по тональности**

Другой важный способ классификации текстов – это их разделение по тональности или эмоциональной окраске. Тональность текста может быть положительной, отрицательной или нейтральной. Классификация текстов по тональности может быть полезна для анализа общественного мнения, отзывов пользователей, анализа социальных медиа и многих других приложений. Например, классификация текстов по тональности может помочь компаниям определить общественное мнение о своих продуктах или услугах.

**Классификация текстов по теме**

Тексты также могут быть классифицированы по теме или содержанию. Например, новостные статьи могут быть классифицированы по различным темам, таким как политика, спорт, наука и т. д. Классификация текстов по теме может быть полезна для организации и поиска информации в больших коллекциях текстов, а также для анализа и понимания различных тематик и трендов.

**Классификация текстов по языку**

Еще один пример классификации текстов – это их разделение по языку. Тексты могут быть написаны на разных языках, и классификация текстов по языку может быть полезна для автоматического определения языка текста. Это может быть полезно для многих приложений, таких как машинный перевод, определение языка ввода пользователя и другие.

Это лишь некоторые примеры классификации текстов. В зависимости от конкретной задачи и доступных данных, можно использовать различные методы и признаки для классификации текстов.

**Преимущества классификации текстов**

Классификация текстов имеет ряд преимуществ, которые делают ее полезной и востребованной в различных областях:

* **Автоматизация процесса:** Классификация текстов позволяет автоматизировать процесс анализа больших объемов текстовой информации. Это особенно полезно в случаях, когда ручной анализ занимает слишком много времени и ресурсов.
* **Быстрота и эффективность:** Классификация текстов позволяет быстро и эффективно обрабатывать большие объемы текстовой информации. Это позволяет сэкономить время и ресурсы, которые могут быть использованы для других задач.
* **Объективность:** Классификация текстов основана на алгоритмах и моделях, что делает ее более объективной и независимой от субъективных мнений и предпочтений.
* **Масштабируемость:** Классификация текстов может быть применена к различным типам текстов и в различных областях. Она может быть масштабирована для обработки больших объемов текстовой информации.

**Ограничения классификации текстов**

Классификация текстов также имеет некоторые ограничения, которые следует учитывать при ее применении:

* **Неоднозначность:** Некоторые тексты могут быть неоднозначными и иметь несколько возможных классификаций. Это может создавать сложности при точной классификации.
* **Неполнота данных:** Классификация текстов требует наличия достаточного количества данных для обучения модели. Если данных недостаточно или они не представляют всю разнообразность текстов, то точность классификации может быть низкой.
* **Зависимость от качества данных:** Качество классификации текстов зависит от качества данных, на которых модель обучается. Если данные содержат ошибки или неточности, то это может отразиться на точности классификации.
* **Сложность языка:** Некоторые языки могут быть сложными для классификации из-за своей грамматической структуры, словарного запаса или других особенностей. Это может затруднить точную классификацию текстов на таких языках.

В целом, классификация текстов является мощным инструментом для обработки и анализа текстовой информации, но требует внимательного подхода и учета ограничений для достижения высокой точности и надежности результатов.

# 15. Классические методы машинного обучения. Проблемы с классификацией текста.

Классические алгоритмы машинного обучения

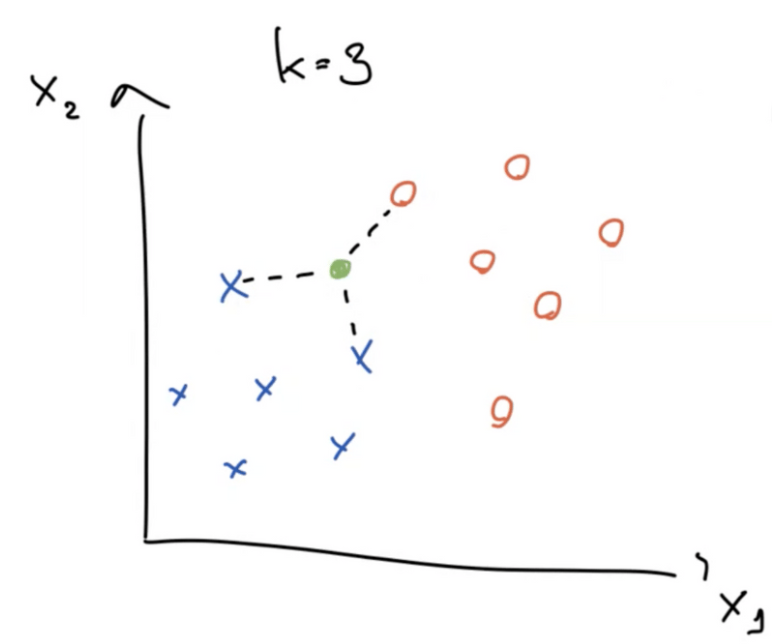
K-ближайших соседей

Метод K-ближайших соседей – простой и эффективный алгоритм, его можно описать известной поговоркой: “Скажи мне, кто твой друг, и я скажу, кто ты”.

***Принцип работы***

Пусть имеется набор данных с заданными классами. Мы можем определить класс неизвестного объекта, если рассмотрим определенное количество ближайших объектов (k) и присвоим тот класс, который имеет большинство “соседей”. Посмотрим на рисунок ниже.

Есть набор точек с двумя классами: синие крестики и красные кружки. Мы хотим определить, к какому классу относится неизвестная зеленая точка. Для этого мы берем k ближайших соседей, в данном случае 3, и смотрим, к каким классам они относятся. Из трех ближайших соседей больше оказалось синих крестиков, соответственно, мы можем предположить, что зеленая точка также, скорее всего, относится к этому классу.



***Как настраивать***

Необходимо подобрать параметр k (количество ближайших соседей) и метрику для измерения расстояний между объектами.

***Задачи:***классификация, также может применяться и для задач регрессии.

Линейная регрессия

Линейная регрессия – простая и эффективная модель машинного обучения, способная решать задачи быстро и недорого.

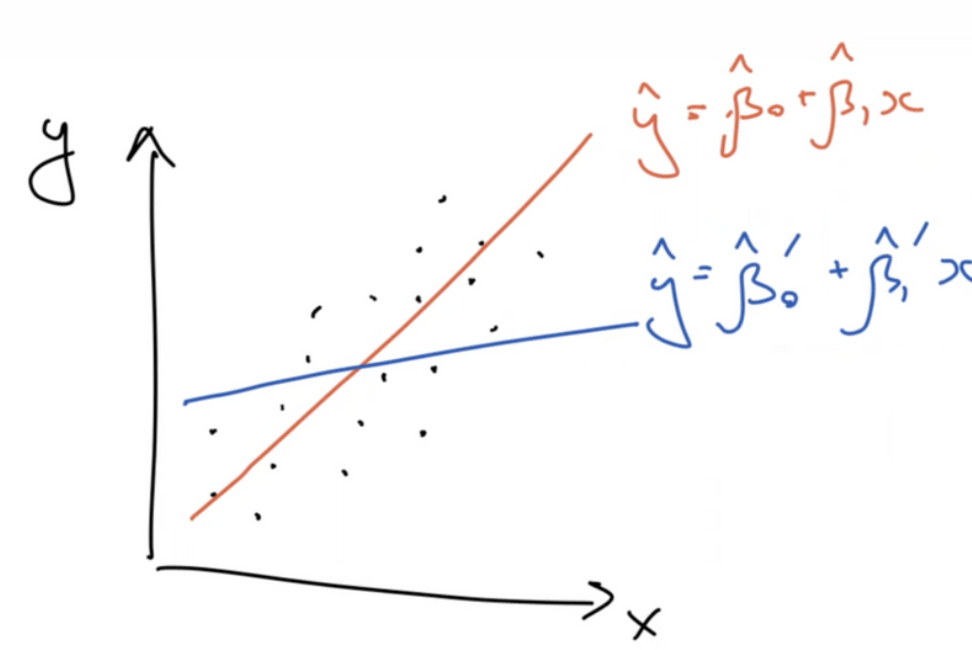
***Принцип работы***

Модель линейной регрессии можно описать уравнением

Здесь *x* – это значения признаков, *y* – целевая переменная, *a* – весовые коэффициенты признаков. При обучении модели весовые коэффициенты подбираются таким образом, чтобы как можно лучше описывалась линейная зависимость признаков от целевой переменной.

Пример: задача предсказания стоимости квартиры в зависимости от площади и удаленности от метро в минутах. Целевой переменной (*y*) будет являться стоимость, а признаками (*x*) – площадь и удаленность.

На рисунке ниже также представлен пример построения линейной регрессии. Красная прямая более точно описывает линейную зависимость *x* от *y*.



***Как настраивать***

Для многих моделей Machine Learning, в частности и для линейной регрессии, можно улучшить итоговое качество с помощью регуляризации.

**Регуляризация** в статистике, машинном обучении, теории обратных задач — метод добавления некоторых дополнительных ограничений к условию с целью решить некорректно поставленную задачу или предотвратить переобучение, то есть ситуацию, когда модель хорошо показывает себя на тренировочный данных, но перестаёт работать на новых.

Распространенные методы регуляризации для повышения качества модели линейной регрессии:

* Ridge — один из методов понижения размерности. Применяется для борьбы с переизбыточностью данных, когда независимые переменные коррелируют друг с другом (мультиколлинеарность), вследствие чего проявляется неустойчивость оценок коэффициентов линейной регрессии.
* LASSO — также как и Ridge, применяется для борьбы с переизбыточностью данных.
* Elastic-Net — модель регрессии с двумя регуляризаторами L1, L2. Частными случаями являются модели LASSO L1 = 0 и Ridge регрессии L2 = 0.

***Задачи:***регрессия.

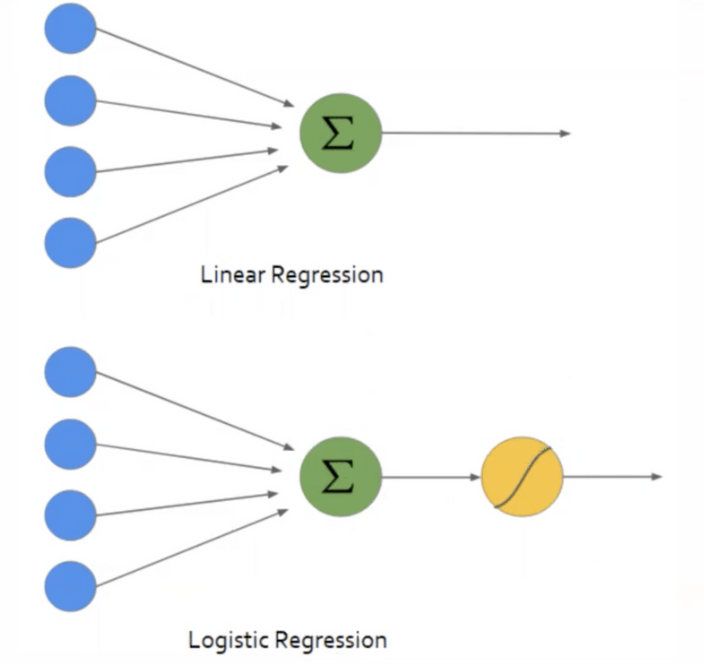
Логистическая регрессия

Логистическая регрессия – также простая и эффективная модель машинного обучения, способная решать задачи быстро и недорого.

***Принцип работы***

Алгоритм логистической регрессии очень похож на алгоритм линейной регрессии. Несмотря на свое название, решает задачу бинарной классификации (классы: 1 и -1).

Указанная выше сумма проходит через функцию сигмоиды, которая возвращает число от 0 до 1, характеризующее вероятность отнесения объекта к классу 1. Пример: логистическую регрессию часто применяют в задачах кредитного скоринга, когда по определенным данным о клиенте нужно определить, стоит ли выдавать ему кредит.



*Иллюстрация алгоритмов линейной и логистической регрессии (*[*источник*](https://www.machinelearningmastery.ru/building-a-logistic-regression-in-python-301d27367c24/)*)*

***Как настраивать***

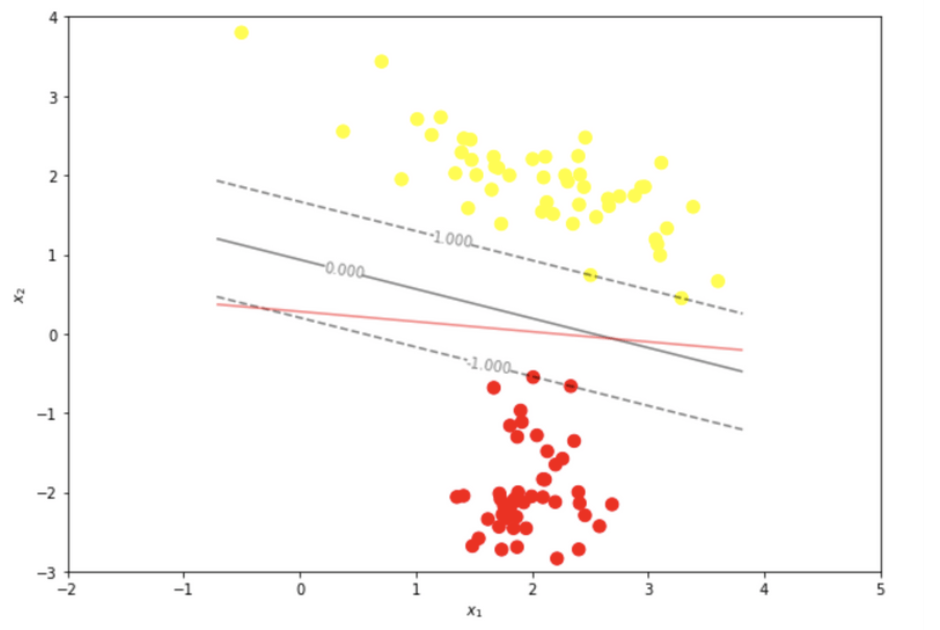
Как и в случае с линейной регрессией, существуют реализации с коэффициентом регуляризации можно выбрать один из методов регуляризации: Ridge, LASSO, Elastic-Net .

***Задачи:***классификация.

Метод опорных векторов (SVM)

***Принцип работы***

Чтобы лучше всего понять алгоритм метода опорных векторов, рассмотрим рисунок. На рисунке приведен пример двух линейно разделимых классов в двумерном пространстве. Идея алгоритма заключается в нахождении оптимальной разделяющей прямой (или гиперплоскости для более высоких пространств) для отделения объектов одного класса от другого. Пунктирные линии выделяют разделяющую полосу и проводятся через объекты, которые называют опорными. Чем шире разделяющая полоса, тем качественнее модель SVM. Чтобы определить класс объекта, достаточно определить, с какой стороны гиперплоскости он находится.



***Как настраивать***

Необходимо подобрать оптимальное ядро (функцию переводящую признаковое пространство в более высокую размерность), если линейная зависимость слабо выражена.

***Задачи:***классификация и регрессия.

Проблемы с классификацией текста методами классического обучения

Классические методы машинного обучения, такие как наивный байесовский классификатор, методы опорных векторов (SVM), решающие деревья и случайные леса, широко используются для классификации текстов. Однако при их применении могут возникать различные проблемы:

1. **Высокая размерность данных**: Текстовые данные часто преобразуются в вектора высокой размерности с использованием методов, таких как "мешок слов" или TF-IDF. Это может привести к проклятию размерности, когда многочисленные признаки затрудняют обучение и интерпретацию модели.
2. **Зависимость от предварительной обработки**: Качество классификации сильно зависит от предварительной обработки текста, включая лемматизацию, стемминг, удаление стоп-слов и выбор признаков. Неправильная предобработка может существенно снизить эффективность классификации.
3. **Не учитываются контекст и порядок слов**: Классические методы часто игнорируют порядок слов и контекст, что ограничивает их способность понимать нюансы языка.
4. **Обработка неструктурированных данных**: Текстовые данные часто являются неструктурированными и могут содержать иронию, жаргон или ошибки, что затрудняет их анализ классическими методами.
5. **Необходимость большого объема ручной разметки**: Для обучения классических моделей часто требуется большое количество ручно размеченных данных, что может быть ресурсоемким.
6. **Обобщение и переобучение**: Модели могут страдать от переобучения на специфических примерах обучающего набора и плохо обобщать на новые данные.
7. **Чувствительность к дисбалансу классов**: Если в обучающем наборе данных одни классы представлены значительно больше, чем другие, это может привести к предвзятости модели.

# 16. Анализ тональности, извлечение аспектов

Анализ тональности – это процесс анализа цифрового текста для определения того, является ли эмоциональный тон сообщения положительным, отрицательным или нейтральным. Сегодня у компаний есть большие объемы текстовых данных, таких как электронные письма, стенограммы чатов службы поддержки клиентов, комментарии в социальных сетях и отзывы. Инструменты анализа тональности могут сканировать этот текст, чтобы автоматически определить отношение автора к теме. Компании используют результаты анализа тональности для улучшения обслуживания клиентов и повышения репутации бренда.

Извлечение аспектов

На основе аспектов

Анализ на основе аспектов фокусируется на определенных аспектах продукта или сервиса. Например, производители ноутбуков опрашивают клиентов об их опыте работы со звуком, графикой, клавиатурой и сенсорной панелью. Они используют инструменты анализа тональности, чтобы связать намерения клиентов с ключевыми словами, связанными с оборудованием.

Например, в отзыве о ресторане аспектами могут быть "обслуживание", "качество пищи", "атмосфера" и так далее.

# 17. Меры оценки системы NLP.

1. Точность

Точность — это простая метрика, которая измеряет долю прогнозов, которые соответствуют действительности. В НЛП точность часто используется в качестве оценочной метрики для задач классификации, таких как анализ настроений или распознавание именованных объектов. Это простая метрика, которая дает общую оценку производительности модели, но она может вводить в заблуждение, если классы несбалансированы, поскольку модель может иметь высокую точность, даже если она делает прогнозы только для большинства классов.

1. Точность, отзыв и F1-Score

Точность, отзыв и оценка F1 — это показатели, которые измеряют способность модели правильно идентифицировать положительные экземпляры. Точность измеряет долю положительных прогнозов, которые на самом деле являются положительными, отзыв измеряет долю положительных случаев, которые были правильно предсказаны как положительные, а показатель F1 является гармоническим средним значением точности и отзыва.

Эти метрики полезны при наличии несбалансированных классов, поскольку они обеспечивают более детальное представление о производительности модели. Например, высокий показатель полноты указывает на то, что модель хорошо находит все положительные экземпляры, а высокий показатель точности указывает на то, что модель способна избегать ложноположительных прогнозов.

1. Матрица путаницы

Матрица путаницы — это таблица, которая обобщает производительность модели путем сравнения истинных меток с предсказанными метками. Он обеспечивает более подробное представление о производительности модели, включая количество истинно положительных, истинно отрицательных, ложноположительных и ложноотрицательных прогнозов. Матрица путаницы особенно полезна для несбалансированных классов, поскольку позволяет нам увидеть производительность модели для каждого класса.

1. ROC-кривая и AUC

Кривая рабочей характеристики приемника (ROC) представляет собой график отношения истинно положительных показателей к уровню ложноположительных результатов, а площадь под кривой (AUC) представляет собой площадь под кривой ROC. Эти метрики используются для оценки производительности моделей бинарной классификации, таких как анализ настроений. Высокий показатель AUC указывает на то, что модель хорошо различает положительные и отрицательные экземпляры, в то время как низкий показатель AUC указывает на то, что модель не работает должным образом.

1. СИНИЙ и РУЖ.

BLEU (дублер двуязычной оценки) и ROUGE (дублер, ориентированный на припоминание для оценки Gisting) — это метрики, используемые для оценки производительности моделей машинного перевода. BLEU измеряет точность предсказанного перевода по отношению к основной истине, а ROUGE измеряет полноту предсказанного перевода по отношению к основной истине. Эти показатели позволяют измерить качество сгенерированного перевода и сравнить его с переводами, созданными человеком.

# 18.  Рекуррентные нейронные сети в задачах NLP. Основные проблемы.

Рекуррентная нейронная сеть (Recurrent Neural Network, RNN) — популярный вид нейронных сетей, используемых в обработке естественного языка (NLP). Рекуррентная нейросеть оценивает произвольные предложения на основе того, насколько часто они встречались в текстах. Это дает меру грамматической и семантической корректности, что позволяет использовать такие модели для перевода текстов. Кроме того, такие модели генерируют новый текст. Обучение модели на поэмах Шекспира позволит генерировать новый текст, похожий на Шекспира.

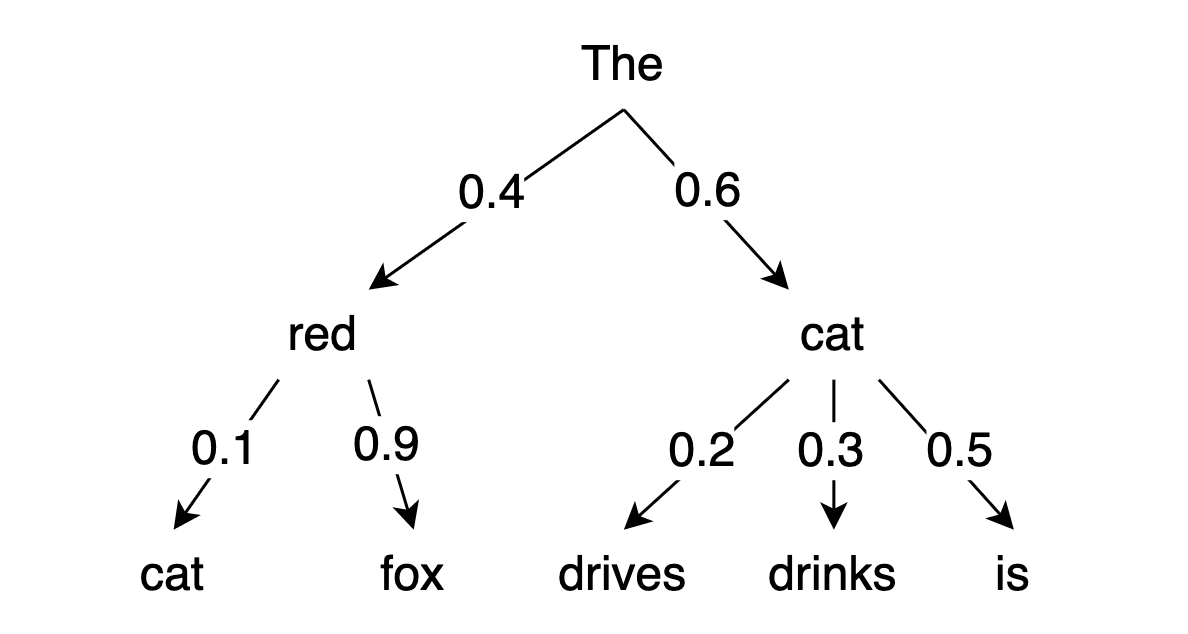
Проблемы

1. **Проблема затухающего и взрывающегося градиента**: В РНС информация передается через длительные последовательности, что может привести к затуханию или, наоборот, чрезмерному увеличению градиентов. Это затрудняет обучение сети, особенно когда речь идет о длинных последовательностях.
2. **Ограниченная память**: Традиционные РНС имеют ограничения по количеству информации, которую они могут запомнить. Хотя варианты как LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Units) справляются с этой проблемой лучше, они все равно страдают от ограничений в памяти при работе с очень длинными последовательностями.
3. **Последовательная обработка**: РНС обрабатывают данные последовательно, что затрудняет параллельную обработку и увеличивает время обучения и инференции, особенно для больших объемов данных.
4. **Трудности с долгосрочными зависимостями**: Несмотря на наличие специализированных архитектур, таких как LSTM и GRU, РНС все еще испытывают трудности с изучением долгосрочных зависимостей в тексте.
5. **Проблемы с контекстуализацией**: Традиционные РНС не всегда эффективно учитывают контекст всего предложения или абзаца, что может привести к недостаточному пониманию нюансов языка.

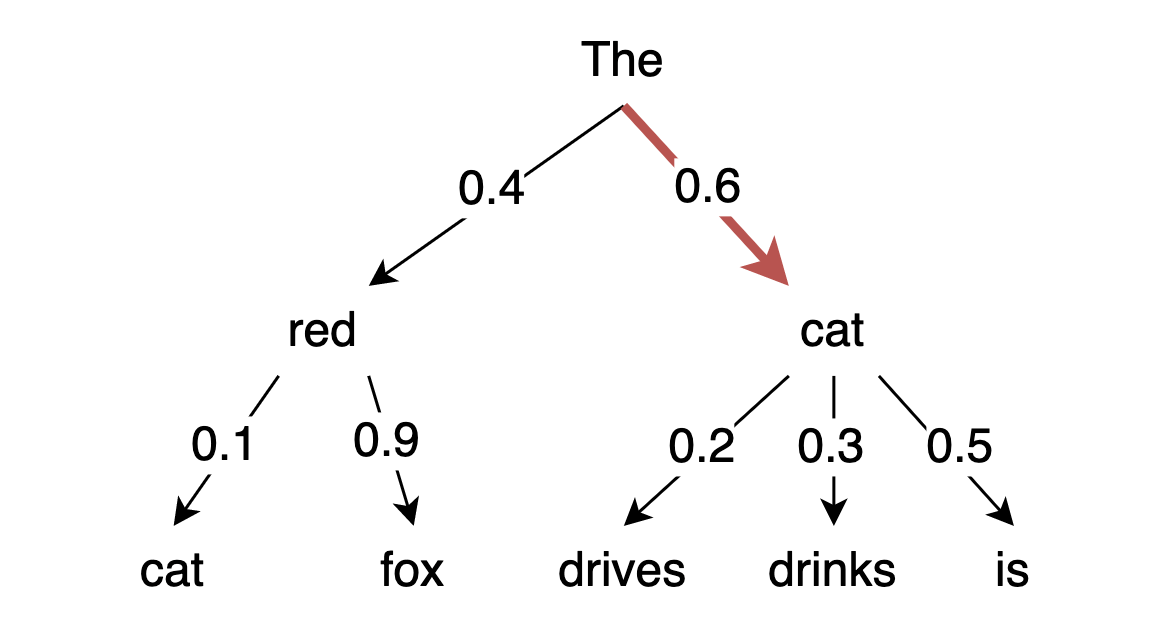
# 19. Принципы генерации текстов методами глубокого обучения. Методы жадного поиска и поиска по лучу.

Жадный поиск

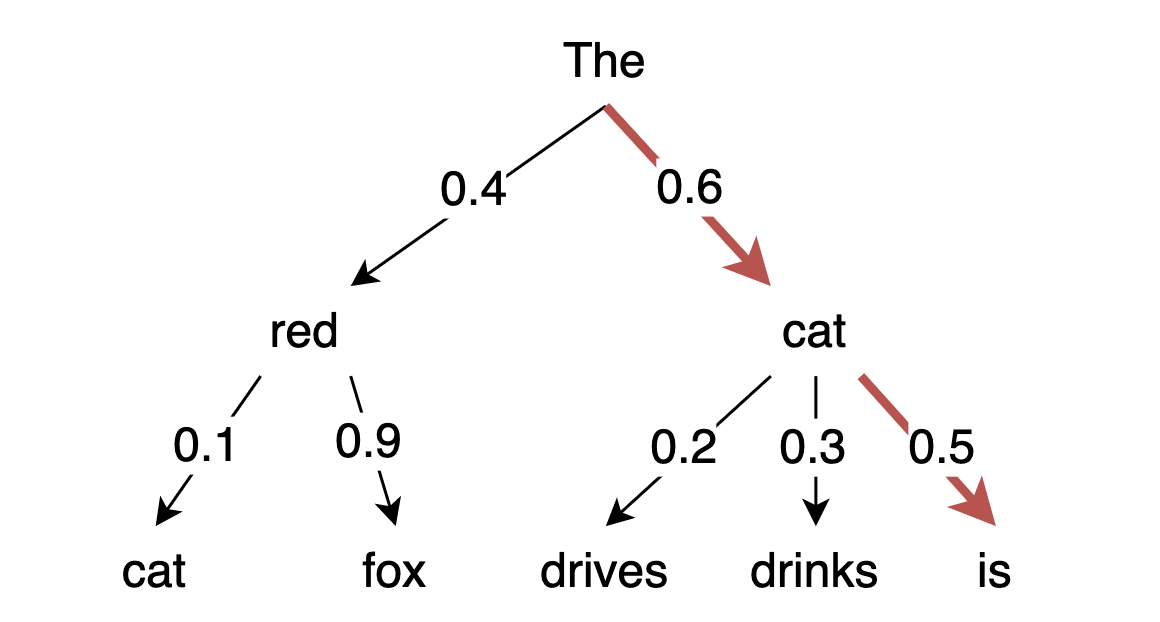
*Жадный поиск* выбирает из языковой модели слово с наибольшей вероятностью в качестве следующего слова. Предположим, что языковая модель предсказывает возможное продолжение предложения, начинающегося с «*The*», выбирая из слов «*красный*» и «*кот*». со связанными с ними вероятностями.



Жадный поиск всегда выбирает слово с наибольшей вероятностью, то есть «*кот*».



Впоследствии жадный поиск выбирает следующее слово с наибольшей вероятностью, то есть «*is*».

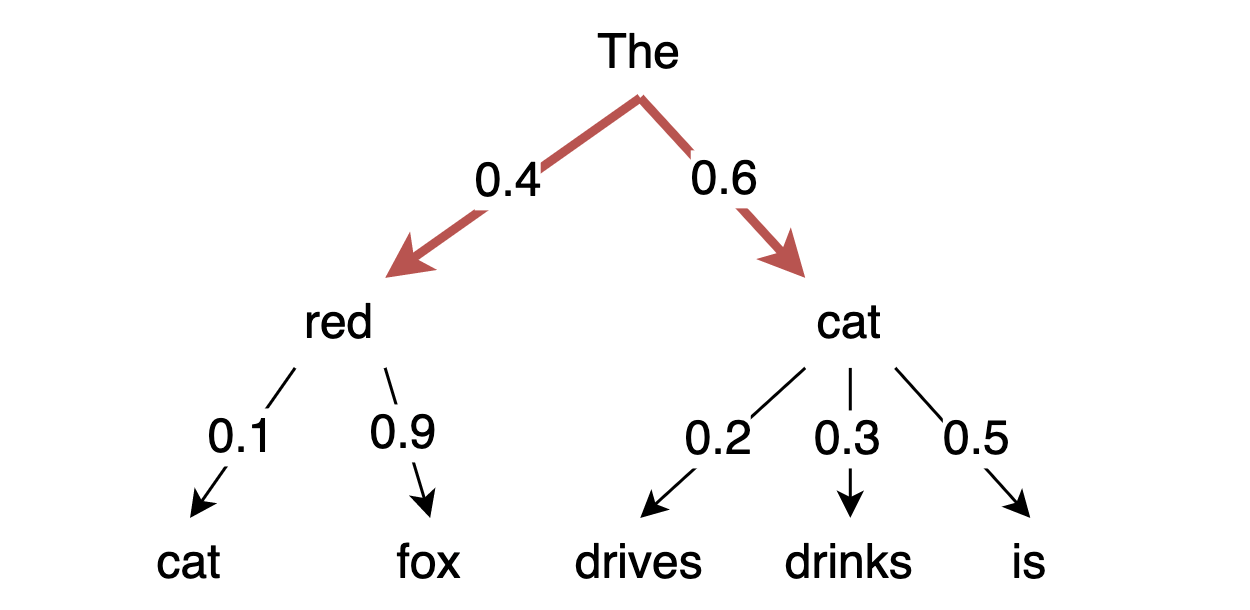


Главный недостаток жадного поиска заключается в том, что он не оптимален для создания предложений с высокой вероятностью, поскольку пропускает слова с высокой вероятностью, скрытые за словами с низкой вероятностью. Действительно, в нашем примере предложение с наибольшей вероятностью — «*Рыжая лиса*» (0,4 \* 0,9 = 0,36), а не «*Кошка*» (0,6). \* 0,5 = 0,30).

Поиск луча

*Поиск луча* решает эту проблему, сохраняя наиболее вероятные гипотезы (также известные как *лучи*) на каждом временном шаге и в конечном итоге выбирая гипотезу с наибольшей общей вероятностью.

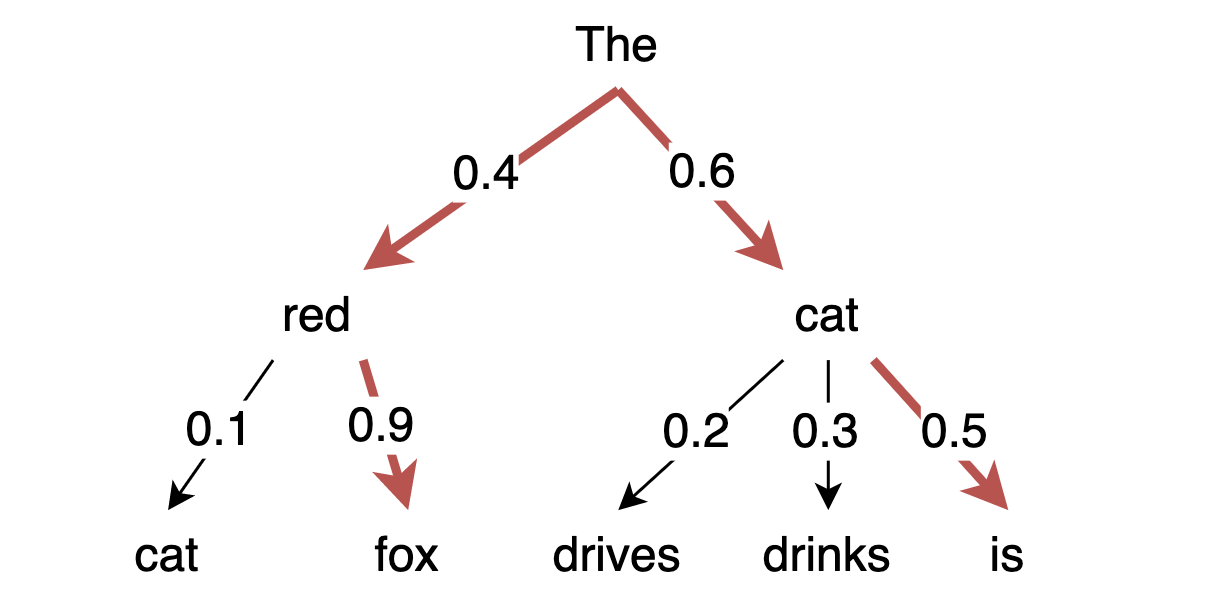
Предположим, мы выполняем поиск луча с двумя лучами. На первом временном шаге лучевой поиск будет учитывать слова «*red*» и «*cat*» вместе с их вероятностями.



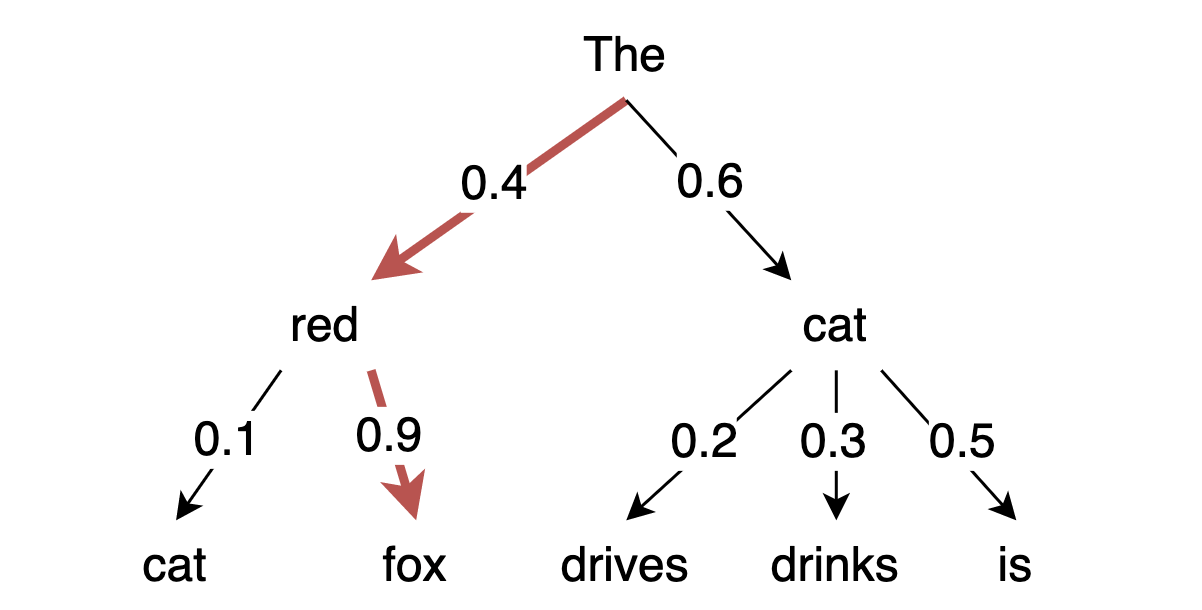
На втором временном шаге поиск луча будет рассматривать все возможные продолжения двух лучей и оставлять только два из них с наивысшей вероятностью. Все возможные продолжения с их вероятностями:

* *Рыжий кот*: 0,4 \* 0,1 = 0,01.
* *Рыжая лиса*: 0,4 \* 0,9 = 0,36 (самый высокий)
* *Кошка водит*: 0,6 \* 0,2 = 0,12.
* *Кошка пьет*: 0,6 \* 0,3 = 0,18.
* *Кошка*: 0,6 \* 0,5 = 0,30 (второе место)

Таким образом, двумя лучами на следующем временном шаге будут «*Рыжая лиса*» и «*Кот*».



Если мы хотим закончить поиск луча здесь, нашим прогнозируемым результатом будет луч с наибольшей вероятностью, то есть «*Рыжая лиса*».



Было замечено, что результаты более беглы по сравнению с жадным поиском, но вывод часто включает повторения одних и тех же последовательностей слов. Простое решение состоит в том, чтобы наказывать результаты всякий раз, когда они содержат n-граммы, присутствующие в уже предсказанном тексте. Тем не менее, штрафы за n-граммы следует использовать с осторожностью. Например, в созданной статье о городе *Нью-Йорк* не следует использовать штраф в 2 грамма, иначе название города появится во всем тексте только один раз.

В общем, трудно настроить штрафы за n-граммы, и получается, что качественный человеческий язык не следует распределению следующих слов с высокой вероятностью.

Как следствие, для решения этих проблем были разработаны другие методы декодирования.

# 20. Метод векторизации TF-IDF.

Что такое TF-IDF в машинном обучении?

Введение в векторизацию TF-IDF в машинном обучении и его реализация с использованием Python.

Одним из важнейших способов изменения размера данных в процессе машинного обучения является использование термина "частота документа с инвертированной частотой", также известная как метод TF-IDF. В этой статье я расскажу, что такое метод TF-IDF в машинном обучении и как его реализовать с помощью языка программирования Python.

Что такое TF-IDF?

Идея метода TF-IDF заключается в том, чтобы придавать большое значение любому термину, который часто встречается в конкретном документе, но не во многих документах в корпусе. Если слово часто встречается в конкретном документе, но не во многих документах, оно, вероятно, удачно описывает содержание этого документа.

Scikit-Learn реализует метод TF-IDF в двух классах: TfidfTransformer, который принимает выходные данные разреженной матрицы, созданные CountVectorizer, и преобразует их, и TfidfVectorizer, который принимает текстовые данные и выполняет как извлечение функций из пакета слов, так и преобразование TF-IDF.

Зачем нужна векторизация TF-IDF?

Предположим, у поисковой системы есть база данных с тысячами описаний кошек, и пользователь хочет найти пушистых кошек, а затем вводит запрос «The furry cat». Поисковой системе необходимо решить, какой результат должен быть возвращен из базы данных.

Если в поисковой системе есть документы, которые соответствуют точному запросу, не будет никаких сомнений, но что, если ей нужно выбрать между частичными совпадениями? Чтобы упростить, скажем, он должен выбрать между этими двумя описаниями:

* «The pretty cat»
* «A furry kitten»

Первое описание содержит 2 из 3 слов запроса, а второе соответствует только 1 из 3, тогда поисковая система выберет первое описание. Как TF-IDF может помочь ему выбрать второе описание вместо первого?

TF одинаков для каждого слова, здесь никакой разницы. Однако можно было бы ожидать, что термины «cat» и «kitten» будут представлены во многих документах (высокая частота документов означает низкий IDF), а вот термин  «furry» будет встречаться в меньшем количестве документов (IDF выше). Таким образом, TF-IDF для «cat» и «kitten» имеет низкое значение, а TF-IDF будет больше для «hairy», то есть в нашей базе данных слово «hairy» имеет большую силу. Различение как «cat» и «kitten».

Если мы используем TF-IDF для взвешивания различных слов, соответствующих запросу, слово «furry» будет более релевантным, чем слово «cat», и поэтому мы могли бы выбрать «A furry kitten» как лучшее соответствие.

# 21. Технология attention. Принципы работы на примере машинного перевода.

Механизм внимания (англ. attention mechanism, attention model) — техника используемая в рекуррентных нейронных сетях (сокр. RNN) и сверточных нейронных сетях (сокр. CNN) для поиска взаимосвязей между различными частями входных и выходных данных.

Изначально механизм внимания был представлен в контексте рекуррентных Seq2seq сетей для "обращения внимания" блоков декодеров на скрытые состояния RNN для любой итерации энкодера, а не только последней.

После успеха этой методики в машинном переводе последовали ее внедрения в других задачах обработки естественного языка и применения к CNN для генерации описания изображения и порождающих состязательных сетях (сокр. GAN).

1. **Основная идея**: Механизм внимания помогает моделям сосредоточиться на наиболее важных частях входных данных при выполнении задачи. Например, при переводе предложения с одного языка на другой модель может сосредоточиться на конкретных словах в исходном предложении, которые наиболее важны для перевода в данный момент.
2. **Процесс работы**:
   * **Кодирование**: Исходное предложение обрабатывается кодирующей частью модели, которая генерирует представление каждого слова в контексте всего предложения.
   * **Внимание**: На этапе внимания модель оценивает, насколько важно каждое слово исходного предложения для каждого слова в целевом предложении. Это делается с помощью весов внимания, которые показывают степень важности каждого слова исходного текста.
   * **Декодирование**: На основе весов внимания и информации, полученной от кодирующей части, декодирующая часть модели генерирует переведённый текст.
3. **Преимущества**:
   * **Улучшенное понимание контекста**: Механизм внимания позволяет модели лучше понимать контекст и устанавливать связи между словами в исходном и целевом предложениях.
   * **Гибкость**: Модель может адаптироваться к различным длинам предложений и структурам, что важно в машинном переводе.
4. **Примеры использования**: Модели, такие как Transformer, широко используют механизм внимания для повышения точности и эффективности перевода.

# 22. Модель sec2sec

Я думаю, что тут речь про seq2seq

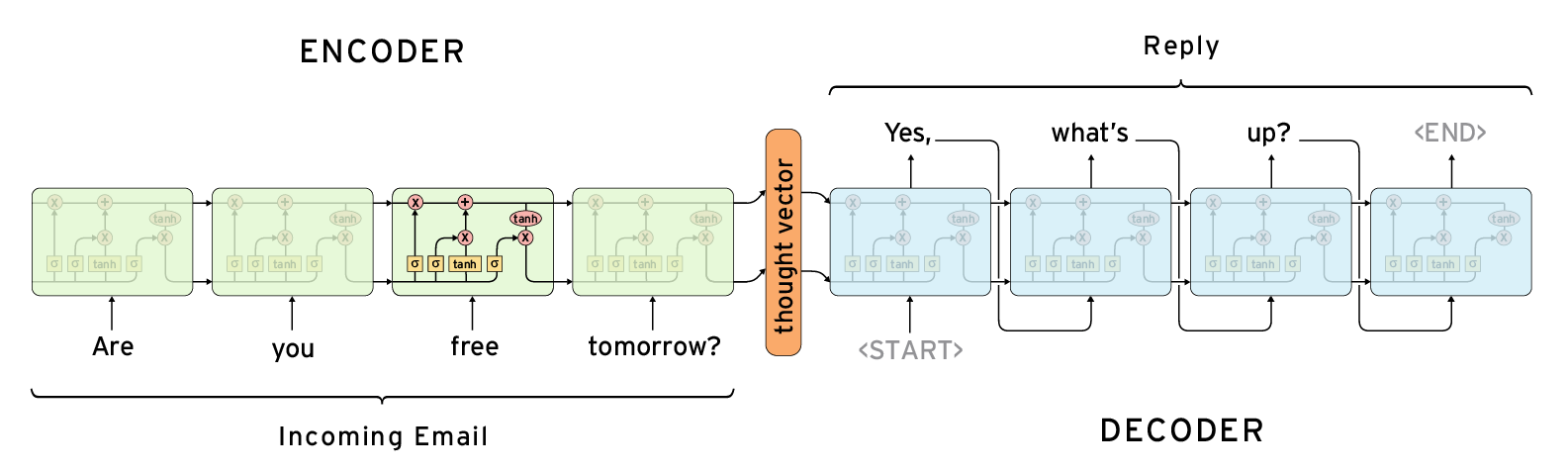
Определение модели «последовательность к последовательности»

Впервые представленная Google в 2014 году модель последовательность в последовательность направлена ​​на сопоставление ввода фиксированной длины с выходом фиксированной длины, где длина ввода и вывода может различаться.

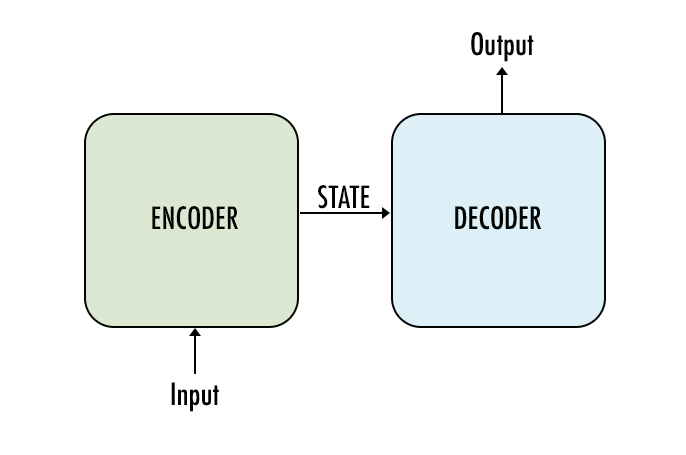
Например, перевод «Что ты делаешь сегодня?» с английского на китайский имеет ввод 5 слов и вывод 7 символов (今天你在做什麼？). Понятно, что мы не можем использовать обычную сеть LSTM для сопоставления каждого слова из английского предложения с китайским предложением.

Вот почему модель последовательности для последовательности используется для решения подобных проблем.

Архитектура Seq2Seq



Модель состоит из 3-х частей: энкодера, промежуточного (энкодера) вектора и декодера.



Общая архитектура

**КОДЕР**

* Стек из нескольких повторяющихся блоков (ячеек LSTM или GRU для повышения производительности), каждый из которых принимает один элемент входной последовательности, собирает информацию для этого элемента и распространяет ее дальше.
* В вопросно-ответной задаче входная последовательность представляет собой набор всех слов из вопроса. Каждое слово представлено как *x\_i*, где *i* — это порядок этого слова.

Кодировщик вектор

* Это окончательное скрытое состояние, полученное из кодирующей части модели. Он рассчитывается по приведенной выше формуле.
* Этот вектор предназначен для инкапсуляции информации для всех входных элементов, чтобы помочь декодеру делать точные прогнозы.
* Он действует как начальное скрытое состояние декодерной части модели.

Декодер

* Стек из нескольких повторяющихся блоков, каждый из которых предсказывает результат *y\_t* на временном шаге *t*.
* Каждая повторяющаяся единица принимает скрытое состояние от предыдущей единицы и создает и выводит, а также свое собственное скрытое состояние.
* В задаче «вопрос-ответ» выходная последовательность представляет собой набор всех слов из ответа. Каждое слово представлено как *y\_i*, где *i* — это порядок этого слова.

# 23.  Векторные модели GloVe и FastText. Принципы работы.

Глобальные векторы (GloVe)

Пеннингтон и др. утверждают, что подход к онлайн-сканированию, используемый word2vec, неоптимален, поскольку он не полностью использует глобальную статистическую информацию о совпадениях слов.

В модели, которую они называют Global Vectors (GloVe), они говорят: *«Модель создает векторное пространство со значимой подструктурой, о чем свидетельствует ее производительность 75% в недавней задаче аналогии слов. Он также превосходит связанные модели в задачах подобия и распознавании именованных объектов».*

Чтобы понять, как работает GloVe, нам нужно понять два основных метода, на которых GloVe был построен — глобальная матричная факторизация и локальное контекстное окно.

В НЛП глобальная матричная факторизация — это процесс использования методов матричной факторизации из линейной алгебры для уменьшения многочленных частотных матриц. Эти матрицы обычно представляют появление или отсутствие слов в документе. Глобальные матричные факторизации, применяемые к матрицам частот терминов, называются скрытым семантическим анализом (LSA).

Методами локального контекстного окна являются CBOW и Skip-Gram. Skip-gram хорошо работает с небольшими объемами обучающих данных и представляет даже слова, которые считаются редкими, тогда как CBOW обучается в несколько раз быстрее и имеет чуть лучшую точность для частых слов.

Авторы статьи отмечают, что вместо того, чтобы изучать необработанные вероятности совпадения, было полезнее узнать отношения этих вероятностей совпадения. Это помогает лучше различать тонкости релевантности терминов и повышает производительность в задачах аналогии слов.

Вот как это работает: вместо извлечения вложений из нейронной сети, которая предназначена для выполнения другой задачи, такой как предсказание соседних слов (CBOW) или предсказание фокусного слова (Skip-Gram), вложения оптимизируются напрямую, так что скалярное произведение двух векторов слов равно логарифму количества раз, когда два слова встречаются рядом друг с другом.

Например, если два слова «кошка» и «собака» встречаются в контексте друг друга, произнесите 20 раз в окне из 10 слов в корпусе документа, то:

**Вектор(кот) . Вектор(собака) = log(10)**

Это заставляет модель кодировать частотное распределение слов, которые встречаются рядом с ними, в более глобальном контексте.

быстрый текст

fastText — это еще один метод встраивания слов, являющийся расширением модели word2vec. Вместо того, чтобы изучать векторы для слов напрямую, fastText представляет каждое слово как n-грамму символов. Так, например, возьмем слово «*искусственный*» с n=3, представлением этого слова в fastText будет ‹*ar, art, rti, tif, ifi, fic, ici, ial. , al*›, где угловые скобки обозначают начало и конец слова.

Это помогает понять значение более коротких слов и позволяет встраиваниям понимать суффиксы и префиксы. Как только слово было представлено с помощью символьных n-грамм, модель пропуска грамм обучается для изучения вложений. Эта модель считается моделью мешка слов со скользящим окном над словом, поскольку не учитывается внутренняя структура слова. Пока символы находятся в пределах этого окна, порядок n-грамм не имеет значения.

fastText хорошо работает с редкими словами. Таким образом, даже если слово не было замечено во время обучения, его можно разбить на n-граммы, чтобы получить его вложения.

И Word2vec, и GloVe не могут обеспечить никакого векторного представления слов, которых нет в модельном словаре. Это огромное преимущество данного метода.

# 24. Python как язык программирования и инструмент для написания проектов NLP.

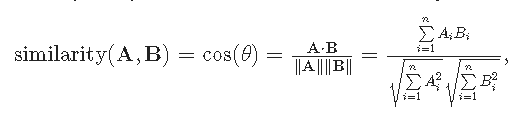
#gpt

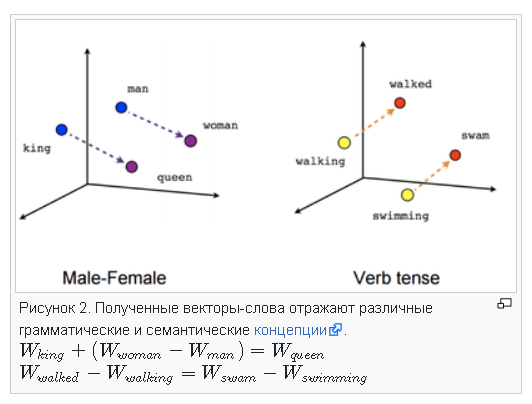
Python является одним из самых популярных языков программирования, особенно в области обработки естественного языка (NLP). Его популярность обусловлена несколькими ключевыми факторами:

1. **Простота и Читаемость**: Python легко читать и писать, благодаря чему разработчики могут быстро создавать и тестировать сложные алгоритмы NLP.
2. **Богатая Экосистема**: Python обладает мощной экосистемой библиотек и фреймворков для NLP, таких как NLTK, SpaCy, Gensim, и Transformers от Hugging Face. Эти инструменты облегчают выполнение широкого спектра задач NLP, включая токенизацию, частеречную разметку, извлечение сущностей, классификацию текста и генерацию языка.
3. **Сообщество и Поддержка**: У Python большое и активное сообщество разработчиков, что обеспечивает хорошую поддержку, множество обучающих материалов и регулярные обновления библиотек.
4. **Интеграция с Другими Технологиями**: Python легко интегрируется с другими языками и технологиями, что делает его удобным для создания сложных систем, включая веб-сервисы и приложения на основе NLP.
5. **Машинное Обучение и Искусственный Интеллект**: Благодаря библиотекам, таким как TensorFlow, PyTorch и Keras, Python является ключевым языком для разработки и реализации моделей машинного обучения, используемых в NLP.
6. **Гибкость и Масштабируемость**: Python поддерживает как быстрое прототипирование, так и масштабирование до крупных систем NLP.
7. **Возможности для Исследований и Разработки**: Python широко используется в академических кругах и индустрии для исследований и разработки в области NLP, что способствует постоянному обновлению и улучшению технологий.

# 25. Векторная модель word2vec.

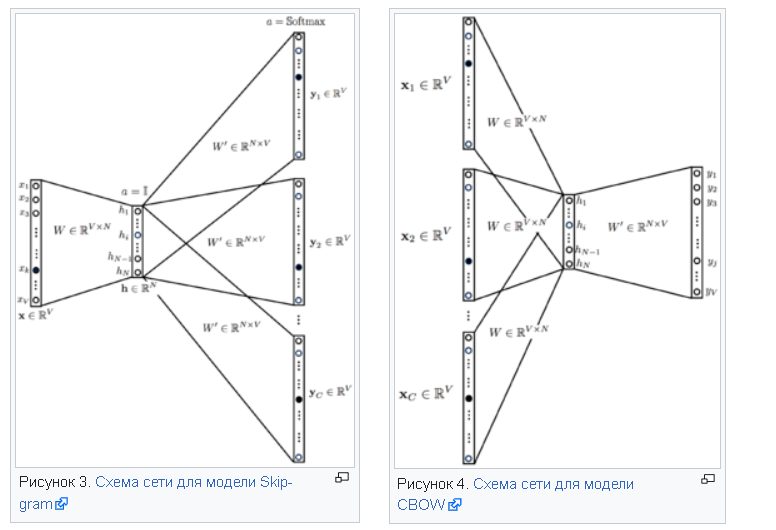
word2vec — способ построения сжатого пространства векторов слов, использующий нейронные сети. Принимает на вход большой текстовый корпус и сопоставляет каждому слову вектор. Сначала он создает словарь, а затем вычисляет векторное представление слов. Векторное представление основывается на контекстной близости: слова, встречающиеся в тексте рядом с одинаковыми словами (а следовательно, имеющие схожий смысл) (рис. 2), в векторном представлении имеют высокое *косинусное сходство* (англ. [cosine similarity](https://en.wikipedia.org/wiki/Cosine_similarity)):





В word2vec существуют две основных модели обучения: *Skip-gram* (рис. 3) и *CBOW* (англ. *Continuous Bag of Words*) (рис. 4). В модели *Skip-gram* по слову предсказываются слова из его контекста, а в модели *CBOW* по контексту подбирается наиболее вероятное слово. На выходном слое используется функция softmax или его вариация, чтобы получить на выходе распределение вероятности каждого слова. В обеих моделях входные и выходные слова подаются в one-hot encoding, благодаря чему при умножении на матрицу W, соединяющую входной и скрытый слои, происходит выбор одной строки W. Размерность N является гиперпараметром алгоритма, а обученная матрица W— выходом, так как ее строки содержат векторные представления слов.

Для ускорения обучения моделей *Skip-gram* и CBOW используются модификации softmax, такие как иерархический softmax и *negative sampling*, позволяющие вычислять распределение вероятностей быстрее, чем за линейное время от размера словаря.



# 26. Принципы работы модели BERT - Bidirectional Encoder Representations from Transformers

Модель BERT (Bidirectional Encoder Representations from Transformers — “двунаправленные презентации кодировщика для трансформеров”) была представлена миру в статье, опубликованной исследователями из Google AI Language. Она вызвала нешуточный ажиотаж в сообществе машинного обучения, представив самые передовые на сегодняшний день результаты для целого ряда разных NLP (Natural Language Processing — “обработка естественного языка”) задач, включая формирование ответов на вопросы (SQuAD v1.1), формирование рассуждений на естественном языке (MNLI) и множество других категорий.

Ключевым техническим нововведением BERT является применение двунаправленного обучения трансформеров (популярной нынче модели с механизмом “внимания”) к языковому моделированию. Этот подход идет в разрез с предыдущими работами, которые рассматривали текстовую последовательность либо только слева направо, либо сочетали обучение слева направо и справа налево. Результаты этой работы показывают, что языковая модель с двунаправленным обучением способна достичь более глубокого понимания языкового контекста и потока, чем однонаправленные языковые модели. В статье исследователи подробно описывают новую технику под названием MLM (Masked Language Model — “маскированное языковое моделирование”), которая позволяет проводить двунаправленное обучение в моделях, для которых ранее это было невозможно.

Как работает BERT

BERT использует трансформер — механизм “внимания”, который изучает контекстуальные отношения между словами (или подсловами) в тексте. В своей оригинальной форме трансформер включает в себя два отдельных механизма — кодировщик, который считывает введенный текст, и декодер, который выдает прогноз для задачи. Поскольку целью BERT является создание языковой модели, то ей необходим только кодировщик. Подробный разбор работы трансформера приведен в [статье](https://arxiv.org/pdf/1706.03762.pdf) Google.

В отличие от направленных моделей, которые считывают вводимый текст последовательно (слева направо или справа налево), кодировщик трансформера считывает сразу всю последовательность слов. Поэтому он считается двунаправленным, хотя правильнее было бы сказать, что он не имеет конкретного направления. Эта особенность позволяет модели изучать контекст слова на основе всего его окружения (слева и справа от слова).

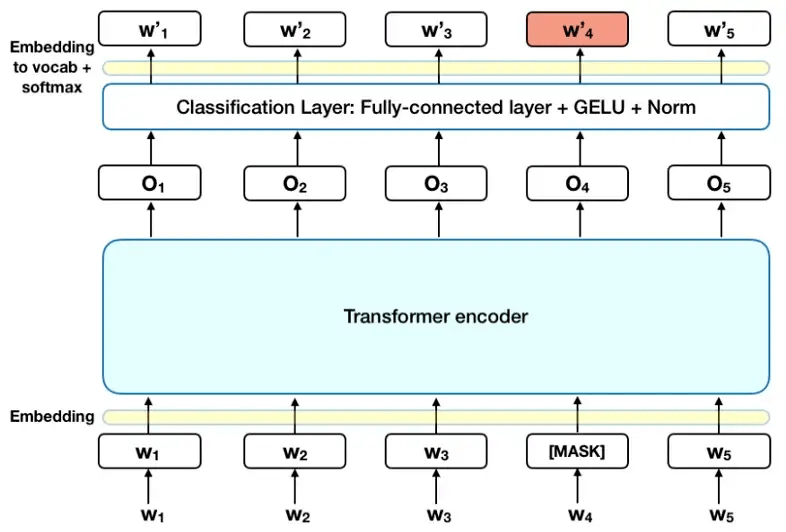
На приведенном ниже графике представлена общая структура кодировщика трансформера. Вход представляет собой последовательность токенов, которые сначала встраиваются в векторы, а затем обрабатываются в нейронной сети. Выход представляет собой последовательность векторов размера H, в которой каждый вектор соответствует входному токену с тем же индексом.

При обучении языковых моделей возникает проблема определения цели прогнозирования. Многие модели предсказывают следующее слово в последовательности (например, “Ребенок пришел домой из \_\_\_”) — это направленный подход, который по своей сути ограничивает контекстное обучение. Чтобы преодолеть эту проблему, BERT использует две стратегии обучения:

Маскированное языковое моделирование (MLM)

Перед вводом последовательности слов в BERT 15% слов в каждой последовательности заменяется токеном [MASK]. Затем модель пытается предсказать исходное значение замаскированных слов на основе контекста, предоставляемого другими, не замаскированными словами в последовательности. С технической точки зрения, предсказание выходных слов требует:

1. Добавления слоя классификации поверх выходных данных кодировщика.
2. Умножение выходных векторов на матрицу векторных представлений словаря (embedding matrix), преобразующую их к размерности словаря.
3. Расчет вероятности каждого слова в словаре с помощью softmax.



Функция потерь BERT учитывает только прогнозы замаскированных значений и игнорирует прогнозы не замаскированных слов. Как следствие, модель сходится медленнее, чем направленные модели, что компенсируется большей осведомленностью о контексте (смотрите Выводы #3).

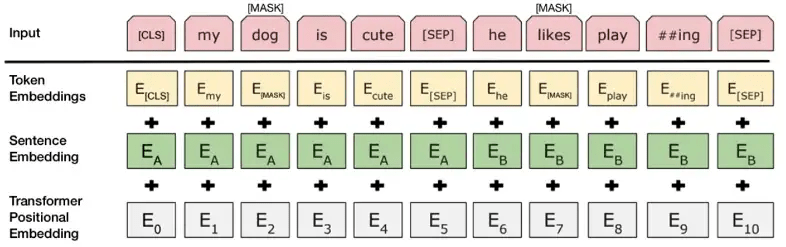
*Примечание: На практике реализация BERT немного сложнее и не заменяет все 15% замаскированных слов.*

Прогнозирование следующего предложения (NSP)

В рамках процесса обучения BERT модель в качестве входных данных получает пары фраз, на которых она учится предсказывать, является ли вторая фраза в паре следующей после первой в исходном тексте. Во время обучения 50% входных данных представляют собой пары, в которых вторая фраза действительно является следующей фразой в исходном тексте, а в остальных 50% в качестве второй фразы выбирается случайная фраза из того же текста. Предполагается, что случайная фраза будет не связана по смыслу с первой фразой.

Чтобы помочь модели различить две фразы в процессе обучения, перед входом в модель входные данные обрабатываются следующим образом:

1. В начало первой фразы вставляется токен [CLS]. В конец каждой из фраз вставляется токен [SEP].
2. К каждому токену добавляется эмбеддинг (векторное представление) фразы, обозначающий Фразу A или Фразу B. Эмбеддинги фраз по своей концепции аналогичны эмбеддингам токенов со словарем из двух элементов.
3. К каждому токену добавляется позиционный эмбеддинг, чтобы указать его положение в последовательности. Концепция и реализация позиционного эмбеддинга хорошо раскрыты в статье, посвященной трансформеру.



*Источник: BERT [Devlin et al., 2018], с некоторыми изменениями*

Чтобы предсказать, действительно ли вторая фраза связана с первой, выполняются следующие шаги:

1. Вся входная последовательность проходит через модель-трансформер.
2. Выход токена [CLS] преобразуется в вектор размерности 2×1 с помощью простого слоя классификации (обученные матрицы весов и смещений).
3. Вычисление вероятности IsNextSequence с помощью softmax.

В процессе обучения BERT-модели, MLM и NSP обучаются вместе с целью минимизировать комбинированную функцию потерь двух стратегий.

# 27. Машинное обучение в НЛП.

Перевод статьи "Использование машинного обучения в обработке естественного языка":

Перед тем как глубоко погрузиться в применение машинного обучения и ИИ для обработки естественного языка (NLP) и текстового анализа, давайте проясним некоторые основные идеи.

Важно понимать, что "машинное обучение" на самом деле означает "обучение машины". Мы знаем, чему машина должна научиться, поэтому наша задача - создать учебную программу и предоставить машине правильно оформленные, релевантные, чистые данные для обучения.

[Изображение: учитель направляет класс студентов-машин]

Когда мы говорим о "модели", мы имеем в виду математическое представление. Ввод данных ключевой. Модель машинного обучения - это сумма знаний, полученных из обучающих данных. Модель изменяется по мере приобретения новых знаний.

В отличие от алгоритмического программирования, модель машинного обучения способна обобщать и справляться с новыми случаями. Если случай похож на то, что модель видела ранее, она может использовать этот предыдущий "опыт" для оценки ситуации. Цель - создать систему, где модель постоянно улучшает свои навыки в задаче, которую вы ей поставили.

Машинное обучение для NLP и текстового анализа включает набор статистических методов для идентификации частей речи, сущностей, настроений и других аспектов текста. Эти методы могут быть выражены в виде модели, которая затем применяется к другим текстам, что известно как обучение с учителем. Это также может быть набор алгоритмов, работающих с большими массивами данных для извлечения смысла, что известно как обучение без учителя. Важно понимать разницу между обучением с учителем и без учителя, и как можно получить лучшее из обоих в одной системе.

[Изображение: горничная заправляет кровать, рядом с ней облако слов с оценками клиентов о состоянии номера, такими как отрицательные "обслуживание в номере" и положительные "вежливость"]

Машинное обучение для NLP помогает аналитикам данных превратить неструктурированный текст в полезные данные и инсайты. Текстовые данные требуют особого подхода к машинному обучению. Это связано с тем, что текстовые данные могут иметь сотни тысяч измерений (слов и фраз), но при этом они очень редки. Например, в английском языке около 100 000 слов, которые часто используются. Но в любом конкретном твите содержится только несколько десятков из них. Это отличается от, например, видеоконтента, где у вас очень высокая размерность, но у вас есть множество данных, с которыми можно работать, поэтому они не такие редкие. Обучение с учителем для обработки естественного языка и текстового анализа В обучении с учителем пакет текстовых документов маркируется или аннотируется примерами того, что машина должна искать и как она должна интерпретировать этот аспект. Эти документы используются для "обучения" статистической модели, которая затем анализирует не помеченный текст.

[Изображение: компьютер в облике человека сидит за столом, усердно работая над учебником]

Позже вы можете использовать более крупные или лучшие наборы данных для повторного обучения модели, поскольку она узнает больше о документах, которые анализирует. Например, вы можете использовать обучение с учителем, чтобы обучить модель анализировать обзоры фильмов, а затем позже обучить ее учитывать рейтинг рецензента.

Наиболее популярные алгоритмы машинного обучения для NLP:

* Метод опорных векторов
* Байесовские сети
* Максимальная энтропия
* Условное случайное поле
* Нейронные сети/Глубокое обучение

Все, что вам действительно нужно знать, если вы столкнетесь с этими терминами, это то, что они представляют собой набор алгоритмов машинного обучения, направляемых данными ученых.

Lexalytics использует обучение с учителем для создания и улучшения наших основных функций текстового анализа и NLP.

Токенизация Токенизация включает разбиение текстового документа на части, которые машина может понять, например, на слова. Вероятно, вы довольно хорошо умеете различать, что является словом, а что бессмыслицей. Английский язык особенно прост. Видите все это белое пространство между буквами и абзацами? Это делает токенизацию действительно легкой. Таким образом, правила NLP достаточны для токенизации на английском языке.

Но как научить алгоритм машинного обучения распознавать слова? И что, если вы работаете не с документами на английском языке? Иероглифические языки, такие как китайский, не имеют пробелов.

Вот где мы используем машинное обучение для токенизации. Китайский язык следует правилам и шаблонам, как и английский, и мы можем обучить модель машинного обучения идентифицировать и понимать их.