NLP

Обработка естественного языка (или NLP — Natural Language Processing) — это раздел искусственного интеллекта, который объединяет лингвистику, компьютерные науки и машинное обучение. Его цель — научить компьютеры анализировать и имитировать человеческий язык.

NLP: применение и кейсы

Применение NLP охватывает разнообразные сценарии — от автоматизации рутинных процессов в бизнесе до улучшения пользовательского опыта в повседневной жизни.

Чат-боты и **голосовые ассистенты**. Популярный пример применения NLP, встречающийся каждый день. Виртуальные помощники способны отвечать на вопросы пользователей, анализировать контекст общения, запоминать предпочтения. Например, если вы попросите Siri включить музыку, она не только выполнит команду, но и предложит песни в стиле, который вам нравится, основываясь на предыдущих запросах.

Системы рекомендаций. Еще одна сфера, где NLP помогает сделать взаимодействие с пользователями максимально персонализированным. Маркетплейсы, стриминговые площадки и онлайн-кинотеатры — яркие примеры таких систем. Используя NLP, они анализируют отзывы, поисковые запросы, предпочтения и другую текстовую информацию, чтобы предложить более точные рекомендации. Благодаря таким алгоритмам мы получаем те самые товары, фильмы и песни, которые идеально подходят под наши интересы, даже если мы сами еще не осознаем, что именно нам нужно.

Анализ пользовательских отзывов. NLP помогает компаниям быстро и точно обрабатывать мнения клиентов. Система классифицирует отзывы по эмоциональной окраске (положительный, отрицательный, нейтральный), выделяют, на какие темы жалуются чаще всего, — и помогает улучшать продукт.

Саммаризация. Также NLP используется для упрощения большого объема текстовой информации. Например, на платформах с тысячами отзывов модели NLP могут автоматически извлечь самые важные аспекты. Вместо того чтобы читать сотни комментариев, пользователь сразу видит ключевые выводы: «Качественная сборка; быстрая доставка; дешевле, чем у других продавцов».

Поисковые системы. Алгоритмы NLP делают поиск информации более точным и быстрым. Они распознают не только слова, но и намерения человека, помогая находить нужную информацию, даже если запрос сформулирован неточно или не полностью.

Фильтрация контента. NLP используется для выявления и блокировки нежелательной информации. Алгоритмы распознают спам, оскорбления, фейковые новости и автоматически скрывают нежелательный контент.

Распознавание речи. NLP помогает превращать аудио в текст, чтобы с ним было проще работать. Например, так расшифровывают звонки в колл-центрах, интервью, голосовые сообщения.

Генерация естественного языка. Нейросети могут автоматически создавать осмысленные тексты, писать статьи, отчеты, описания товаров, отвечать на вопросы, вести диалог и даже писать программный код. Это экономит время и упрощает многие интеллектуальные задачи.

Машинный перевод. Автоматический перевод текстов с одного языка на другой помогает быстро обмениваться информацией и преодолевать языковые барьеры. Современные системы перевода используют NLP, чтобы учитывать контекст и получать более качественные и естественные переводы.

Как работает NLP?

Модели обработки естественного языка (NLP) основаны на двух компонентах: данных для обучения и алгоритмах, которые позволяют их анализировать и использовать.

Методы и алгоритмы NLP

Обработка естественного языка включает несколько подходов, которые помогают компьютерам понимать и обрабатывать человеческий язык. Эти подходы развивались постепенно — от простых правил до сложных нейросетевых алгоритмов.

Методы на основе правил (Rule-Based Methods)

Это самый ранний подход к NLP. Он работает по простому принципу: если в тексте есть определенные слова или конструкции, заранее прописанные в системе, она выдает нужный результат.

Например, чтобы найти в тексте имена, города или даты, система использует готовые списки слов или шаблоны. Если фраза совпадает с шаблоном — задача выполнена.

Плюсы	Минусы
Легко реализовать.	Правила нужно писать и обновлять вручную.
Результат понятный и предсказуемый.	Плохо справляется с нестандартными ситуациями.

Статистические методы (Statistical Methods)

Статистические методы используют вероятность появления слов или словосочетаний в тексте. Эти вероятности вычисляются на основе анализа больших массивов текстовых данных.

Примеры таких алгоритмов:

- **Модели n-грамм** — предсказывают, какое слово в предложении будет следующим, опираясь на предыдущие.
- **TF-IDF** — определяет, какие слова важны в тексте, сравнивая их частоту внутри документа и по всей коллекции текстов.

Плюсы	Минусы
Более гибкие по сравнению с методами на основе правил.	Не учитывают смысл и контекст.
Хорошо работают с большими объемами данных.	Чувствительны к качеству и однородности текста.

Машинное обучение (Machine Learning)

Машинное обучение позволяет системам NLP учиться автоматически — на основе примеров. Алгоритмы анализируют тексты с заранее известными метками и учатся находить закономерности.

Например, чтобы понять, позитивный отзыв или негативный, модель изучает много размеченных отзывов и замечает, какие слова чаще встречаются в положительных или отрицательных текстах. Потом она может анализировать новые отзывы — даже те, которые не видела раньше.

Часто используемые алгоритмы:

- Наивный байесовский классификатор.
- Метод опорных векторов (SVM).
- Деревья решений.

Плюсы	Минусы
Автоматизация процесса обучения.	Нужно заранее разметить много текстов.
Высокая точность при достаточном количестве обучающих данных.	Модели плохо понимают контекст без дополнительных доработок.

Глубокое обучение (Deep Learning)

Самый современный и мощный подход в NLP основан на нейросетях. Глубокое обучение позволяет компьютеру понимать не только отдельные слова, но и контекст,

смысловые связи и нюансы языка.

Наиболее популярные модели:

- Рекуррентные нейронные сети (RNN, LSTM).
- Transformer (BERT, GPT).

Плюсы	Минусы
Отлично работают с контекстом и смыслом.	Требуют больших вычислительных ресурсов.
Подходят для самых разных задач — от перевода до генерации текста.	Высокие затраты на обучение.

Роль данных в NLP

Сбор данных

Для создания NLP-модели необходимы большие массивы текстовых данных, которые могут быть собраны из различных источников:

1. **Внешние данные**. В эту категорию входит публично доступная информация — тексты из социальных сетей, новостных сайтов, книг, научных публикаций и других открытых ресурсов.

Особенности:

- Большой объем и разнообразие, позволяющие обучить модель работать с разными стилями и форматами.
- Много «шума», из-за чего данные требуют дополнительной очистки и проверки.

2. **Внутренние данные**. Включают в себя внутренние данные компании, например, базы знаний, клиентские запросы, отзывы, отчеты и другие документы. Эти данные позволяют разрабатывать узкоспециализированные NLP-системы.

Особенности:

- Лучше всего подходят для конкретных бизнес-задач. Например, для создания чат-бота, настроенного под нужды компании.
- Объем может быть ограниченным, а качество данных — неидеальным (ошибки, опечатки и т. д.).

3. **Сгенерированные данные.** Когда нужных текстовых данных недостаточно, их можно создать искусственно. Команды разработчиков могут самостоятельно писать и редактировать примеры текстов или диалогов, имитируя реальные сценарии, в том числе используя уже предобученные языковые модели.

- Неограниченный объем (при необходимости можно сгенерировать любое количество примеров).
- Позволяют имитировать редкие сценарии.
- Создание таких текстов требует экспертных знаний, а также тщательного контроля качества, чтобы данные были реалистичными.

Разметка данных для NLP

Разметка текста — это процесс добавления к текстовым данным специальной информации (меток). Именно разметка делает данные понятными для алгоритмов, позволяет им выявлять закономерности и правильно решать NLP-задачи.

Типы разметки текстов

Для каждой NLP-задачи нужен свой тип разметки.

Классификация текстов. Каждому тексту присваивается метка на основе его содержания. Это может быть тема («спорт», «экономика», «культура»), тип документа («новость», «отзыв», «инструкция») или любая другая категория.
Анализ тональности текста. Классификация текстов по эмоциональной окраске («отрицательный», «нейтральный», «положительный»).

Разметка данных для NLP

Разметка текста — это процесс добавления к текстовым данным специальной информации (меток). Именно разметка делает данные понятными для алгоритмов, позволяет им выявлять закономерности и правильно решать NLP-задачи.

Типы разметки текстов

Для каждой NLP-задачи нужен свой тип разметки.

Классификация текстов. Каждому тексту присваивается метка на основе его содержания. Это может быть тема («спорт», «экономика», «культура»), тип документа («новость», «отзыв», «инструкция») или любая другая категория.
Анализ тональности текста. Классификация текстов по эмоциональной окраске («отрицательный», «нейтральный», «положительный»).
Распознавание именованных сущностей (Named Entity Recognition). Разметка отдельных слов или фраз в тексте, относящихся к конкретным категориям: имена людей, названия компаний, даты, денежные единицы и т. д.
Side-by-side. Сравнение двух вариантов текста и выбор более точного или качественного. Такая разметка часто применяется для оценки работы генеративных моделей.

Сравнение может выполняться по таким параметрам, как:

- Соответствие запросу

- Фактическая точность.
- Грамматика и стилистика.
- Ясность и логичность.
- Отсутствие токсичности и предвзятости.

Оценка саджестов (подсказок). Отдельный подвид разметки, когда разметчик оценивает релевантность и качество предложенных системой подсказок. Это могут быть варианты автодополнения фраз в поисковых системах или чат-ботах.

Распознавание намерений (Intent Recognition). Определение цели или смысла сообщения и отнесение его к конкретному намерению. Например, текст «включи музыку» относится к намерению пользователя «запрос на воспроизведение». Такая разметка необходима для чат-ботов и голосовых ассистентов.

Предобработка текста

Перед подачей данных в NLP-модель текст обычно проходит предварительную обработку. Одни из самых распространенных методов на этом этапе — очистка, токенизация, удаление стоп-слов, а также лемматизация и стемминг.

Очистка текста — предполагает удаление или исправление всего, что мешает точному анализу текста. Сюда могут входить:

- Удаление ненужных и дублирующихся записей.
- Удаление нерелевантной информации из текста, например ссылок, HTML-тегов.
- Удаление лишних символов (знаков препинания, дублирующихся пробелов).
- Приведение всех букв к верхнему или нижнему регистру.

Кроме того, нередко добавляется этап нормализации и приведения данных к единому стандарту: например, унификация написания дат, единиц измерения или сокращений. Все это позволяет избавиться от «шума» и повысить качество последующей обработки текста.

Токенизация — разделение текста на более мелкие единицы (токены). Это могут быть отдельные слова, фразы или предложения. Практически все методы NLP требуют обязательной токенизации текста.

Стоп-слова — это распространенные слова, не влияющие на смысл текста (предлоги, союзы, частицы). Их удаление помогает алгоритмам лучше концентрироваться на важных словах и признаках, улучшая точность анализа текста.

Удаление стоп-слов полезно для статистических методов и классических моделей машинного обучения, таких как SVM, логистическая регрессия, TF-IDF. В глубоких моделях стоп-слова чаще всего оставляют, так как их удаление может привести к потере смысла сказанного.

Стемминг (stemming) — способ упрощения слов и получения их основы, при котором отбрасываются окончания и суффиксы.

Стемминг позволяет уменьшить количество уникальных слов и ускорить анализ текста. Главный недостаток метода — полученные основы не всегда являются осмысленными словами и могут не совпадать с реальным корнем.

Лемматизация (lemmatization) — более интеллектуальный метод, который приводит слова к начальной словарной форме (лемме), учитывая их грамматический контекст и значение в предложении. Лемматизация применяется в случаях, когда важна точность смысла и правильная интерпретация текста.

Когда использовать лемматизацию и стемминг?

Необходимость в этих методах зависит от того, какие методы NLP вы планируете использовать:

- **Для методов на основе правил** стемминг и лемматизация бывают очень полезны, особенно если вы составляете словари или правила обработки. Например, если вы хотите создать список ключевых слов, приведение слов к единой форме поможет уменьшить объем данных и избежать повторов.
- **Для статистических методов и классического машинного обучения** лемматизация и стемминг также обычно необходимы, поскольку они помогают значительно сократить размерность текстовых данных. Это улучшает точность модели и ускоряет ее работу.
- **Для глубокого обучения** лемматизация и стемминг, как правило, не используются. Современные нейросети самостоятельно улавливают нюансы языка, поэтому упрощение текста до базовой формы может привести к потере важных смысловых оттенков.

Векторизация текста: что это и для чего используется?

Векторизация текста — это важный этап подготовки данных, при котором слова, предложения или целые тексты преобразуются в числовые представления (векторы). Простыми словами: векторизация позволяет компьютеру «понять», о чем текст, представив его в виде набора чисел.

Как выполняется векторизация?

Существуют разные подходы к переводу текста в числовой формат:

Bag-of-Words (Мешок слов)

Bag-of-Words — это простой подход, при котором текст представляется как набор слов и их частот. Порядок слов при этом не учитывается.

Сложность этой модели в том, как определить словарь и как подсчитать вхождение слов.

Когда размер словаря увеличивается, вектор документа тоже растет. В примере выше, длина вектора равна количеству известных слов.

В некоторых случаях, у нас может быть неимоверно большой объем данных и тогда вектор может состоять из тысяч или миллионов элементов. Более того, каждый документ может содержать лишь малую часть слов из словаря.

Как следствие, в векторном представлении будет много нулей. Векторы с большим количеством нулей называются разреженным векторами (sparse vectors), они требуют больше памяти и вычислительных ресурсов.

Однако мы можем уменьшить количество известных слов, когда используем эту модель, чтобы снизить требования к вычислительным ресурсам. Для этого можно использовать те же техники, что мы уже рассматривали до создания мешка слов:

- игнорирование регистра слов;
- игнорирование пунктуации;
- выкидывание стоп-слов;
- приведение слов к их базовым формам (лемматизация и стемминг);
- исправление неправильно написанных слов.

Другой, более сложный способ создания словаря – использовать сгруппированные слова. Это изменит размер словаря и даст мешку слов больше деталей о документе. Такой подход называется «N-грамма».

N-грамма это последовательность каких-либо сущностей (слов, букв, чисел, цифр и т.д.). В контексте языковых корпусов, под N-граммой обычно понимают последовательность слов. Юниграмма это одно слово, биграмма это последовательность двух слов, триграмма – три слова и так далее. Цифра N обозначает, сколько сгруппированных слов входит в N-грамму. В модель попадают не все возможные N-граммы, а только те, что фигурируют в корпусе.

TF-IDF

У частотного скоринга есть проблема: слова с наибольшей частотностью имеют, соответственно, наибольшую оценку. В этих словах может быть не так много информационного выигрыша для модели, как в менее частых словах. Один из способов исправить ситуацию – понижать оценку слова, которое часто встречается **во всех схожих документах**. Это называется TF-IDF.

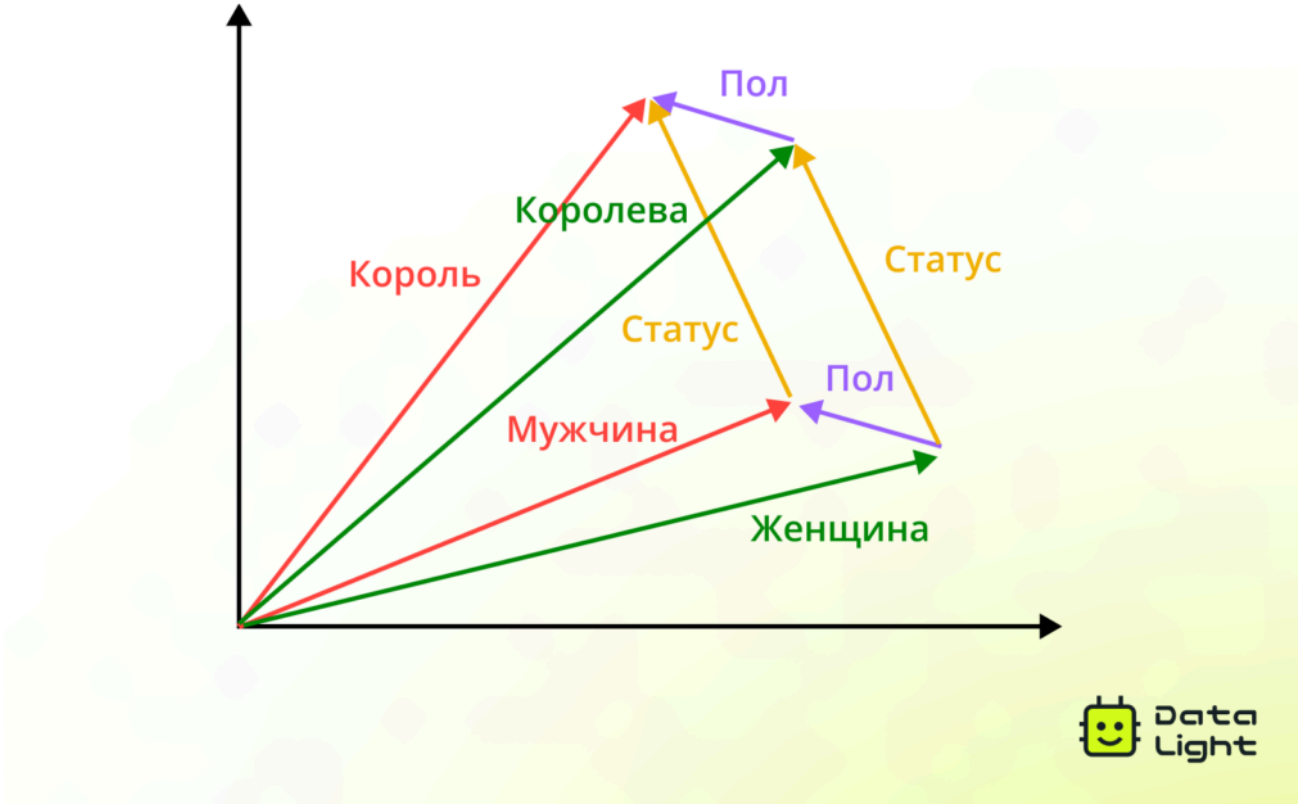
TF-IDF (сокращение от term frequency — inverse document frequency) – это статистическая мера для оценки важности слова в документе, который является частью коллекции или корпуса.

TF-IDF — улучшенный подход к векторизации текста. Он учитывает не только частоту слова в конкретном документе, но и то, насколько это слово уникально и информативно в масштабах всего набора. Частые слова получают низкий вес, а уникальные слова, которые редко встречаются и являются важными для данного документа, получают высокий вес.

Эти способы векторизации особенно эффективны для статистических методов и классического машинного обучения. Они не требуют большого количества вычислительных ресурсов и хорошо подходят для простых задач классификации или анализа тональности текста.

Embeddings (Эмбеддинги)

Эмбеддинги — современные подходы к векторизации, в которых слова, предложения или даже целые тексты представлены в виде плотных, многомерных векторов. Эти векторы отражают смысловые связи и контекст использования слов.



Популярные типы эмбеддингов:

- **Word2Vec, GloVe, FastText** — не учитывают контекст и дают одно векторное представление для слова во всех ситуациях.
- **Контекстуальные эмбеддинги (BERT, GPT, ELMo)** — учитывают контекст использования слова. Одно и то же слово в разных предложениях будет представлено разными векторами.

Эмбеддинги особенно полезны и активно используются в **глубоком обучении**, где очень важен контекст.

Инструменты и библиотеки для NLP

Использование специализированных библиотек значительно облегчает и ускоряет обработку естественного языка.

NLTK (Natural Language Toolkit)

NLTK — одна из наиболее известных библиотек для обработки естественного языка на Python. Она включает инструменты для токенизации, удаления стоп-слов, лемматизации, стемминга, анализа частей речи и тональности.

Особенности:

- Поддержка большого количества языков, в том числе русского.
- Простота интеграции с другими библиотеками и инструментами.
- Медленная скорость работы на больших массивах данных.

spaCy

spaCy — быстрая и удобная библиотека для работы с текстами на Python. Она хорошо подходит для задач NLP, потому что использует Cython (расширение для ускорения Python-кода). Включает токенизацию, распознавание именованных сущностей (NER), разметку частей речи, синтаксический анализ, а также встроенные эмбеддинги слов и возможность обучения собственных моделей.

Особенности:

- Высокая скорость и производительность обработки текстов.
- Меньшее количество поддерживаемых языков по сравнению с NLTK.

scikit-learn

scikit-learn — еще одна популярная библиотека на Python. С ее помощью можно преобразовать текст в числовой вид, обучить модель классификации и оценить ее качество.

Особенности:

- Ограниченный набор инструментов для предварительной обработки текста.
- Используется в комбинации с другими библиотеками.