Российский университет транспорта РУТ (МИИТ)
Высшая инженерная школа (ВИШ)

# Анализ больших текстовых данных и текстовый поиск

## Лекция 1
## Основные этапы работы с текстом. Задача классификации.

**8 сентября 2025**

# Промежуточный контроль

1. Лабораторные работы на семинарах (всего две)
    a. Первая будет в эту пятницу (12 сентября) у ШАД-311, и в следующий понедельник (15 сентября) у ШЦТ-311
    b. Вторая - примерно через месяц после первой
    c. Лабораторные по системе зачёт/незачёт
    d. Если не успели доделать и сдать на семинаре, когда были выданы задания, можно будет сдать на следующем занятии
2. Контрольная работа
    a. Будет в формате письменной работы (тест и короткие ответы на вопросы)
    b. Вопросы будут по материалам лекций и теории из практических занятий
3. Для прохождения промежуточного контроля важно сдать обе лабораторные работы и написать контрольную не на “2”

# Вступление: Основные задачи обработки текстовых данных

- **Информационный/текстовый поиск** (**Information retrieval**) - задача поиска в большой коллекции документов тех, что удовлетворяют запросам пользователя (поисковые системы, базы данных)
- **Интеллектуальный анализ текстов** (**Text mining**) - задача выделения из неструктурированных текстовых данных полезной информации (классификация, кластеризация, аннотирование документов, поиск ключевых понятий, связей между ними)
- Извлечение информации и **построение графов знаний** (**Knowledge graphs**) - задача автоматического построения структурированных семантических сетей из неструктурированной информации, например, из коллекции текстов
- **Языковое моделирование** (**Language Modeling**) и **генерация текстов**

# Вступпление: Natural Language Processing

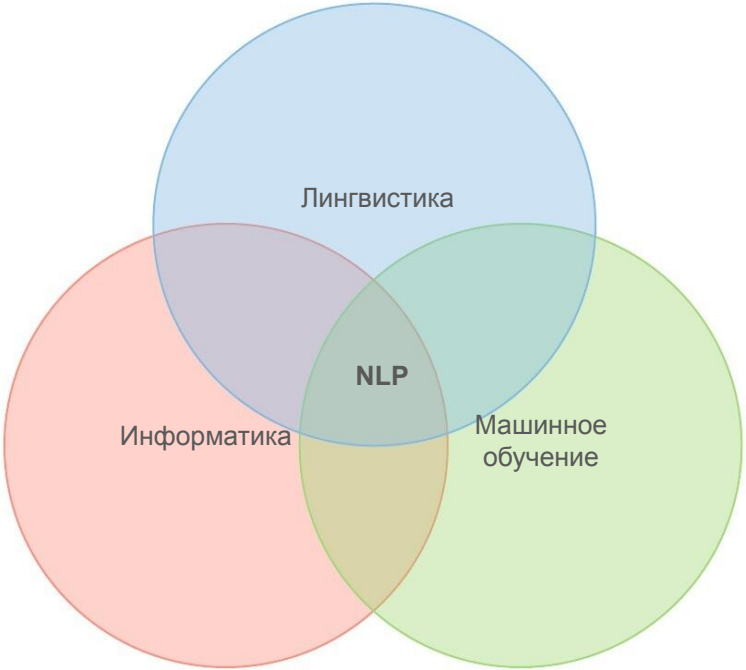
**NLP** — обработка естественного языка

NLP позволяет применять алгоритмы машинного обучения для текста и речи

**Цель**: научить компьютер понимать, интерпретировать и генерировать человеческий язык

Можем разделить на 2 глобальные задачи:

- Понимание естественного языка (NLU)
- Генерация естественного языка (NLG)



# Вступление: NLU и NLG

- Понимание естественного языка (Natural Language Understanding)

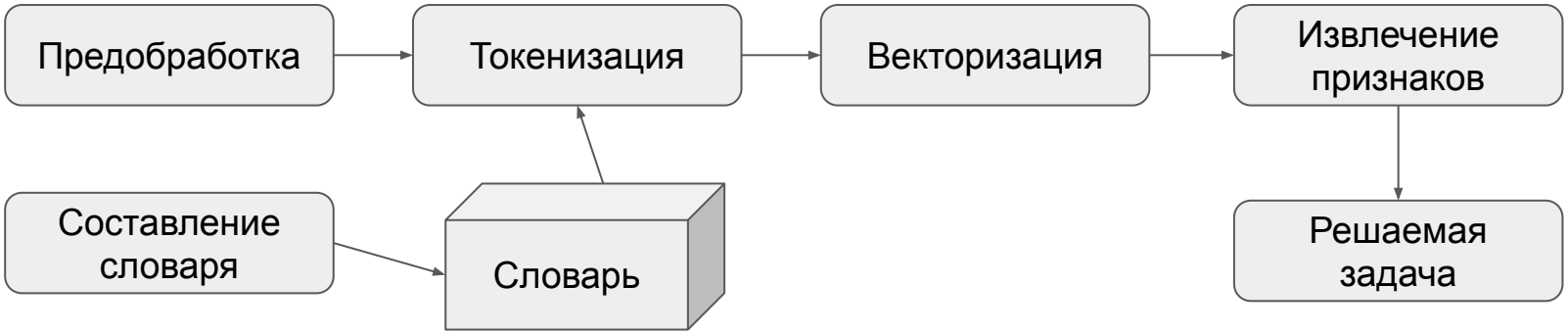
  Понимание машиной прочитанного на основании грамматики и контекста, определение предполагаемого значения предложения. Извлечение семантики из текста - значение слов в контексте. (Примеры: анализ тональности - настроений, текстовый поиск)

- Генерация естественного языка (Natural Language Generation)

  Генерация текста на основе заданного набора данных. (Пример: краткий пересказ текста, ChatGPT)

# Этапы работы с текстом

1. Предварительная обработка текста - удаление шума и нормализация
2. Токенизация - разбиение на части для последующего кодирования (можно считать частью предобработки)
3. Векторизация токенов
4. Извлечение признаков
5. Решаемая задача: анализ тональности, выделение тем, классификация…



# Кодирование символов в классическом понимании

- Символьная кодировка – это таблица соответствия, где каждому символу (букве, цифре, знаку) присваивается уникальный числовой код
- Вы вводите текст, компьютер преобразует эти символы в числовые коды для хранения и обработки.
- Ранее использовалась символьная кодировка ASCII. Сейчас стандартом является Юникод (Unicode), в частности UTF-8

| символ | 10-й код | 2-й код | символ | 10-й код | 2-й код | символ | 10-й код | 2-й код | символ | 10-й код | 2-й код |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 32 | 00100000 | 8 | 56 | 00111000 | P | 80 | 01010000 | h | 104 | 01101000 |
| ! | 33 | 00100001 | 9 | 57 | 00111001 | Q | 81 | 01010001 | i | 105 | 01101001 |
| " | 34 | 00100010 | : | 58 | 00111010 | R | 82 | 01010010 | j | 106 | 01101010 |
| # | 35 | 00100011 | ; | 59 | 00111011 | S | 83 | 01010011 | k | 107 | 01101011 |
| $ | 36 | 00100100 | < | 60 | 00111100 | T | 84 | 01010100 | l | 108 | 01101100 |
| % | 37 | 00100101 | = | 61 | 00111101 | U | 85 | 01010101 | m | 109 | 01101101 |
| & | 38 | 00100110 | > | 62 | 00111110 | V | 86 | 01010110 | n | 110 | 01101110 |
| ' | 39 | 00100111 | ? | 63 | 00111111 | W | 87 | 01010111 | o | 111 | 01101111 |
| ( | 40 | 00101000 | @ | 64 | 01000000 | X | 88 | 01011000 | p | 112 | 01110000 |
| ) | 41 | 00101001 | A | 65 | 01000001 | Y | 89 | 01011001 | q | 113 | 01110001 |
| * | 42 | 00101010 | B | 66 | 01000010 | Z | 90 | 01011010 | r | 114 | 01110010 |
| + | 43 | 00101011 | C | 67 | 01000011 | [ | 91 | 01011011 | s | 115 | 01110011 |
| , | 44 | 00101100 | D | 68 | 01000100 | \ | 92 | 01011100 | t | 116 | 01110100 |
| - | 45 | 00101101 | E | 69 | 01000101 | ] | 93 | 01011101 | u | 117 | 01110101 |
| . | 46 | 00101110 | F | 70 | 01000110 | ^ | 94 | 01011110 | v | 118 | 01110110 |
| / | 47 | 00101111 | G | 71 | 01000111 | _ | 95 | 01011111 | w | 119 | 01110111 |
| 0 | 48 | 00110000 | H | 72 | 01001000 | ` | 96 | 01100000 | x | 120 | 01111000 |
| 1 | 49 | 00110001 | I | 73 | 01001001 | a | 97 | 01100001 | y | 121 | 01111001 |
| 2 | 50 | 00110010 | J | 74 | 01001010 | b | 98 | 01100010 | z | 122 | 01111010 |
| 3 | 51 | 00110011 | K | 75 | 01001011 | c | 99 | 01100011 | { | 123 | 01111011 |
| 4 | 52 | 00110100 | L | 76 | 01001100 | d | 100 | 01100100 | \| | 124 | 01111100 |
| 5 | 53 | 00110101 | M | 77 | 01001101 | e | 101 | 01100101 | } | 125 | 01111101 |
| 6 | 54 | 00110110 | N | 78 | 01001110 | f | 102 | 01100110 | ~ | 126 | 01111110 |
| 7 | 55 | 00110111 | O | 79 | 01001111 | g | 103 | 01100111 | □ | 127 | 01111111 |

**Когда работает**: анализ аббревиатур, сокращений

Пример:

CAT

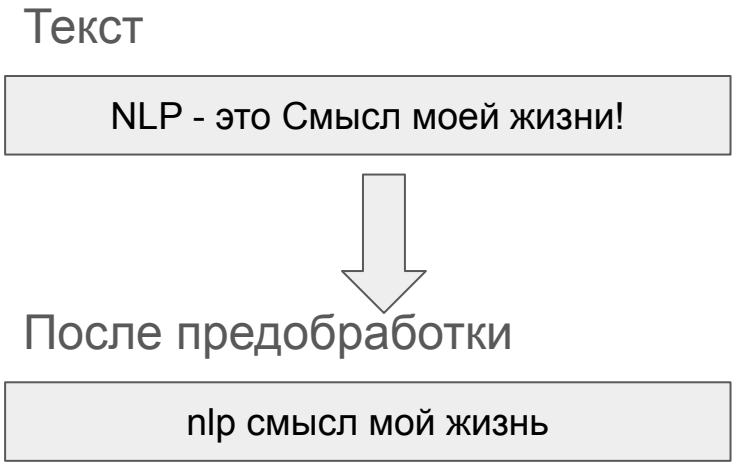
C – 01000011
A – 01000001
T – 01010100

Проблема расхода памяти

# Предварительная обработка текстовых данных (Data preprocessing)

Включает в себя:

- Определение языка текста
- Приведение к нижнему регистру
- Удаление нерелевантных символов
- Удаление стоп-слов
- Приведение к нормальной форме - лемматизация/стемминг
- *Токенизация (рассмотрим отдельно)



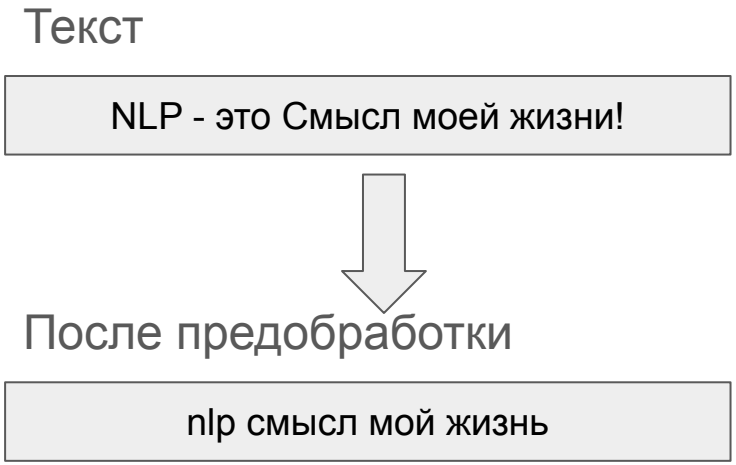
# Предварительная обработка текстовых данных (Data preprocessing)

**Цель**: избавиться от шума в текстовых данных

Основная идея заключается в уменьшении размеров формируемого словаря для повышения точности дальнейшего решения задачи

В случае токенизации не на уровне слов методы будут отличаться

Сейчас для каждой задачи существуют свои подходы



# Регулярные выражения

- С помощью регулярных выражений можно удалять конкретные символы из строки
- Основные выражения:

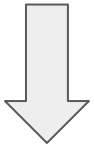
. – любой символ, кроме перевода строки;
\w – один символ;
\d – одна цифра;
\s – один пробел;
\W – один НЕсимвол;
\D – одна НЕцифра;
\S – один НЕпробел;
[abc] – находит любой из указанных символов match any of a, b, or c;
[^abc] – находит любой символ, кроме указанных;
[a-g] – находит символ в промежутке от a до g.

# Удаление нерелевантных символов

- Удаление различных знаков препинания и специальных символов
- В зависимости от задачи необходимость в удалении символов может отсутствовать
- Например, в задачах генерации текста знаки препинания играют важную роль в донесении смысла и интонации предложения

Привели к нижнему регистру

nlp - это смысл моей жизни!



После удаления символов

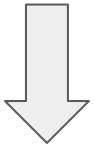
nlp это смысл моей жизни

# Удаление стоп-слов

- Стоп-слова — это часто встречающиеся служебные слова (например: и, в, на, это, что, как), которые обычно не несут значимой смысловой нагрузки для анализа текста
- Включают в себя предлоги, союзы, местоимения, междометия и тд.
- Точный перечень зависит от конкретной решаемой задачи

После удаления символов

nlp **это** смысл **моей** жизни



После удаления стоп-слов

nlp смысл **моей** жизни

Когда нам лучше удалить слово “моей”, а когда оставить?

# Приведение слова к нормальной форме

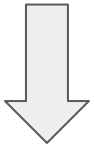
- Входит в морфологический анализ текста
- Обычно тексты содержат разные грамматические формы одного и того же слова
- Также могут встречаться однокоренные слова и близкие по значению
- Нормализацию используют для приведения всех встречающиеся форм слова к одной, нормальной форме
- Цель: значительное уменьшение размеров словаря с целью повышения точности классификации
- Способы нормализации слов
  - Лемматизация
  - Стемминг

# Лемматизация - приведение к нормальной форме

- Приведение к словарной начальной форме (единственное число, именительный падеж, для глаголов - неопределённая форма)
- Инструменты: pymorphy3, spaCy, Stanza
- Примеры
  - Лесной, лесного, лесному → лесной
  - леса → лес
  - Танцующая → танцевать
  - детей → ребенок
  - делаешь → делать

После удаления стоп-слов

nlp смысл моей жизни



После лемматизации

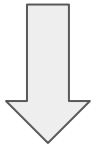
nlp смысл мой жизнь

# Стемминг - выделение  псевдоосновы

- Стемминг отрезает окончания, иногда искажая слова
- Применяется в поисковых системах для расширения поискового запроса пользователя
- Инструменты: nltk.stem.SnowballStemmer (есть русский)
- Примеры
  - Лесной, лесного, лес, лесистый → лес
  - Система, системный, систематизировать → систем
  - Машины, машиной → машин

После удаления стоп-слов

nlp смысл моей жизни



После стемминга

nlp смысл мо жизн

# Омонимия - одна из основных проблем обработки естественного языка

*Пример:*
*На завод привезли **стекло**.*
*Масло **стекло** на пол.*

*Нес медведь, шагая к **рынку**,*
*На продажу меду **крынку**.*
*Вдруг на мишку - вот **напасть**!*
*Осы вздумали **напасть**.*
*Мишка с армией **осиной***
*Дрался вырванной **осиной**.*
*Мог ли в ярость он не **впасть**,*
*Если осы лезли в **пасть**,*
*Жалили куда **попало**,*
*Им за это и **попало**.*

# Морфологическая неоднозначность - омонимия

- Неоднозначность по леммам: косой, стали

  Самая важная для информационного поиска

- Неоднозначность по частям речи: стали
- Неоднозначность по грамматическим характеристикам (падеж, число и др.)

  Например, очень часто неоднозначность именительный vs. винительный падеж

# Представление текста в виде числовых признаков

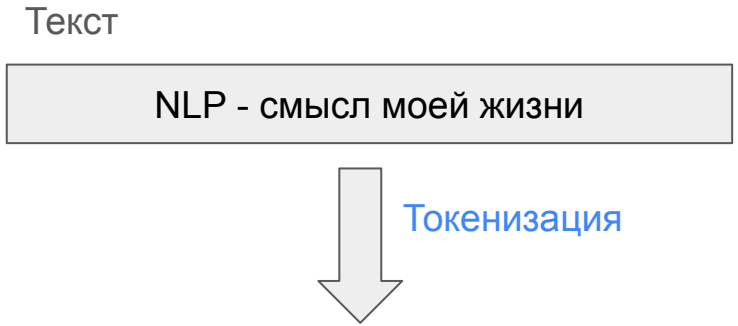
- Текст - это последовательность некоторых языковых единиц (символов, слов, токенов, …)
- Его необходимо представить в виде числовых признаков для использования в языковом моделировании
- Как мы можем это сделать?

Текст

NLP - смысл моей жизни

# Представление текста в виде числовых признаков: Токенизация

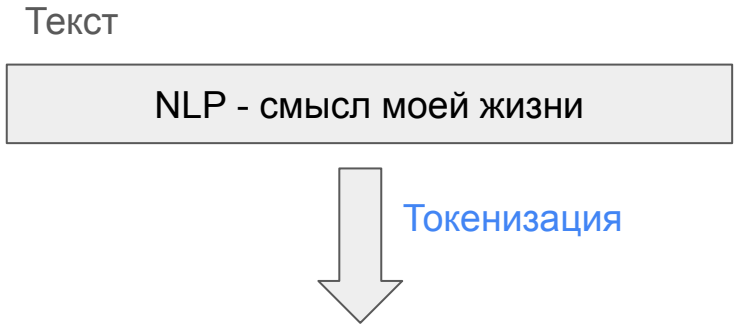
- Мы всегда можем “**сегментировать**” текст - разбить его на отдельные части (например, слова)
- Выберем те слова, которые могут встретиться в интересующих нас текстах, и составим из них **словарь**
- Присвоим каждому слову в словаре свой **индекс**
- В результате мы можем представить текст как **последовательность индексов** слов в словаре




| NLP | - | смысл | моей | жизни |
|---|---|---|---|---|
| 101 | 32 | 476 | 491 | 244 |

# Токенизация

- Токенизация — это процесс разбиения текста на более мелкие единицы (токены), которые удобнее анализировать и обрабатывать
- В нашем случае токенами будут отдельные слова
- В зависимости от задачи, токенами могут быть отдельные символы, сочетания из нескольких букв, словосочетания, предложения, и даже абзацы



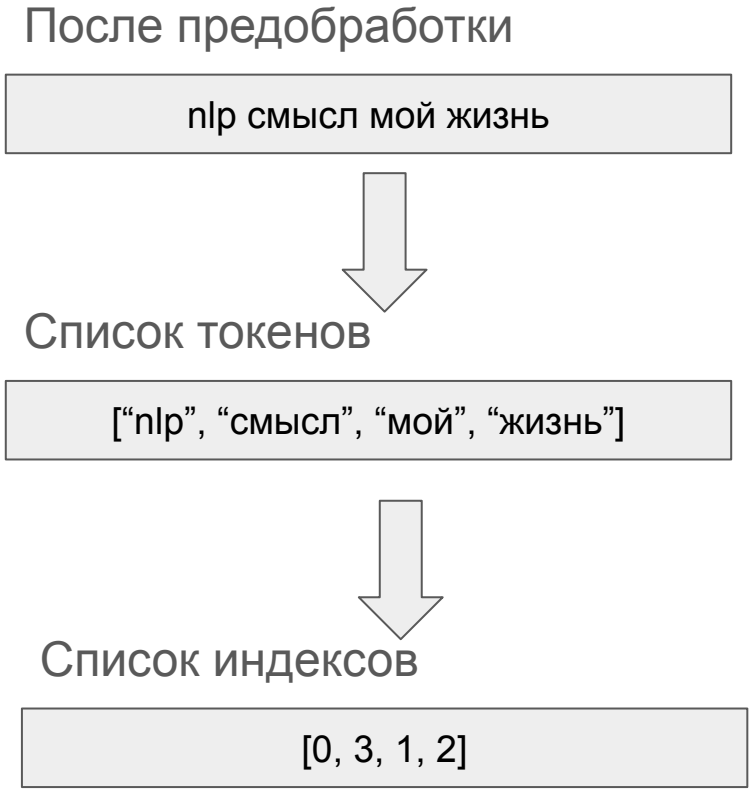

| NLP | - | смысл | моей | жизни |
|---|---|---|---|---|
| 101 | 32 | 476 | 491 | 244 |

# Токенизация

- На вход подаётся текст
- На выходе получаем список токенов и список их индексов в словаре

Словарь

| Токен | Индекс |
|---|---|
| nlp | 0 |
| мой | 1 |
| жизнь | 2 |
| смысл | 3 |
| кот | 4 |



# Векторизация

- Переводим токены из списка в пространство векторов признаков (feature vectors) для дальнейшего извлечения признаков моделями
- Можно кодировать как категориальные переменные (пример справа)
- А можно сопоставить каждому слову вектор в некотором пространстве признаков и обучать его
- Вектор текста может быть функцией от векторов токенов (напрмер, усреднение)

| Токен | Индекс | Вектор |
|---|---|---|
| nlp | 0 | [1, 0, 0, 0, 0] |
| мой | 1 | [0, 1, 0, 0, 0] |
| жизнь | 2 | [0, 0, 1, 0, 0] |
| смысл | 3 | [0, 0, 0, 1, 0] |
| кот | 4 | [0, 0, 0, 0, 1] |

# Векторизация: One-hot encoding

- One-hot encoding - простейший способ кодирования последовательности токенов
- Каждый токен представляется как вектор из нулей длины V, где V - количество слов в словаре, с единственной единицей
- Эта единица расположена в месте, соответствующем номеру данного токена в словаре
- Учитываем только факт вхождения слова

| Токен | Индекс | Вектор |
|---|---|---|
| nlp | 0 | [1, 0, 0, 0, 0] |
| мой | 1 | [0, 1, 0, 0, 0] |
| жизнь | 2 | [0, 0, 1, 0, 0] |
| смысл | 3 | [0, 0, 0, 1, 0] |
| кот | 4 | [0, 0, 0, 0, 1] |

# Векторизация: One-hot encoding

- Получим разреженную матрицу (это плохо)
- Не учитываем частоту вхождения слов
- Не учитываем контекст и порядок слов

Словарь

Текст

| | nlp | мой | жизнь | смысл | кот |
|---|---|---|---|---|---|
| nlp | 1 | 0 | 0 | 0 | 0 |
| смысл | 0 | 0 | 0 | 1 | 0 |
| мой | 0 | 1 | 0 | 0 | 0 |
| жизнь | 0 | 0 | 1 | 0 | 0 |

# Векторизация: One-hot encoding

- Можем использовать частотный подход - вместо 1 указывать число вхождений слова
- Слишком частые слова: как правило, стоп-слова
- Слишком редкие слова: могут быть опечатками
- “Коты - смысл моей жизни с котами”.

Словарь

Текст

| | nlp | мой | жизнь | смысл | кот |
|---|---|---|---|---|---|
| кот | 0 | 0 | 0 | 0 | 2 |
| смысл | 0 | 0 | 0 | 1 | 0 |
| мой | 0 | 1 | 0 | 0 | 0 |
| жизнь | 0 | 0 | 1 | 0 | 0 |

# Векторизация: Мешок слов (Bag of Words)

- **Мешок слов**: как one-hot encoding, но матрицы “схлопываются” - каждый документ представляется одним вектором
- В строке на месте слова либо бинарный признак (0 или 1) как факт вхождения, либо частота вхождения слова в документ
- На практике: сумма векторов частот

| | nlp | мой | жизнь | смысл | кот |
|---|---|---|---|---|---|
| “Коты - смысл моей жизни с котами” | 0 | 1 | 1 | 1 | 2 |
| “NLP - смысл моей жизни” | 1 | 1 | 1 | 1 | 0 |

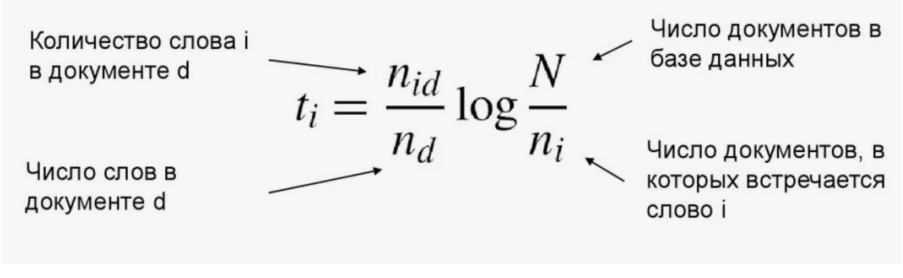
# Векторизация: Мешок слов (Bag of Words)

- **Документ** - отдельный текст (например, у нас сейчас это предложение)
- **Корпус** - множество документов
- **Мешок слов**: как one-hot encoding, но матрицы “схлопываются” - каждый документ представляется одним вектором
- Не учитываем контекст и порядок слов

*Банк купил компанию* и *Компания купила банк* - это одинаковые векторы для разных по смыслу документов

# TF-IDF - для информационного поиска

- **TF-IDF = TF * IDF** - количество вхождений токена в документ, нормированное на частоту встречаемости токена в документах корпуса
- **N** = количество документов в корпусе
- **Term Frequency** TF(t, d) = частота появления слова t в документе d
- **Document Frequency** DF(t) = количество документов, содержащих слово t
- **Inverse Document Frequency** IDF(t) = обратная доля документов, содержащих слово t



# TF-IDF - для информационного поиска

- Принцип: документ определяется словами, которые часто встречаются в этом документе, но редко встречаются в других документах
- Цель: определение токенов, специфичных для данного текста
- Большой вес в TF-IDF получат слова с высокой частотой в пределах конкретного документа и с низкой частотой в других документах
- Мы рискуем получить завышенный показатель TF-IDF, если мы имеем корпус с документами, сильно отличающимися по размеру
  - если слово встречается только в маленьких текстах (TF)
  - если слово часто встречается только в одном большом документе (IDF)
- Можно использовать для инициализации более сложных эмбеддингов

# Задача классификации текстов

- Одна из типичных задач анализа текстовых данных
- Дано:
  - Документ d
  - Множество классов (например, темы обращений граждан)
- Задача:
  - Определить, к какой теме относится документ d
- Другие примеры:
  - Бинарная классификация: спам / не спам
  - Многоклассовая классификация: к какой теме относится данная новость?

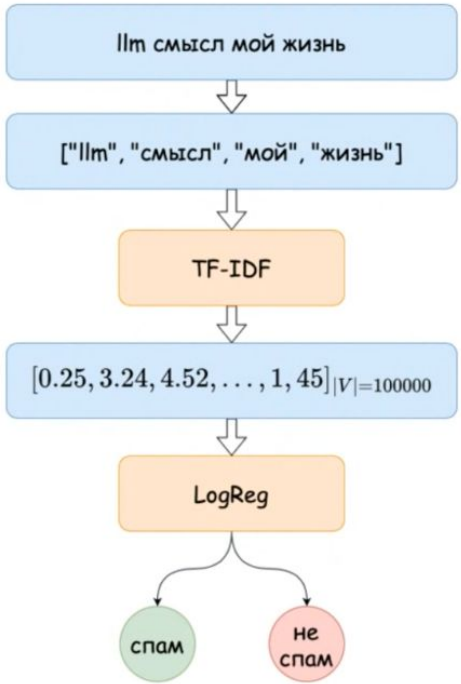
# Задача классификации текстов

- Обучение с учителем
- Нужен большой корпус текстовых документов
- Выполнить разметку данных - для каждого документа указать тему, к которой он принадлежит (класс)
- Представить документы векторами признаков
- Обучить классификатор на размеренном датасете
- kNN, логистическая регрессия, дерево решений, случайный лес, нейронная сеть,…

# Задача классификации текстов

## Как использовать

- Задача: классификация текста (спам / не спам)
- Вход: корпус текстов, метки, текст для теста
  - тест не содержится в трейне
- По корпусу составили словарь
- Для каждого текста: через OHE или TF-IDF получили вектор текста
  - размерность – размер словаря
- Выбрали и обучили классификатор
- Проверили на тестовом тексте – какие могут быть проблемы?



# Языковое моделирование

$$P(y_1, y_2, \ldots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \cdots \cdot P(y_n|y_1, \ldots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t}).$$

- Задача: научиться генерировать следующий токен по предыдущим
- Глобально решение задачи выглядит так
  - Представить текст в векторном виде
    - разобьем текст на токены
    - сформируем словарь
    - как-нибудь закодируем каждый токен из словаря
  - Модель => внутренние свойства и взаимосвязи токенов в последовательности
  - Внутреннее представление => для каждого токена из словаря оценить вероятность появления этого токена на следующей позиции
  - Вероятностное распределение для следующего токена => семплируем токен