**Токенизация**

1)Токенизация (иногда – сегментация) по предложениям – это процесс разделения письменного языка на предложения-компоненты. Идея выглядит довольно простой. В английском и некоторых других языках мы можем вычленять предложение каждый раз, когда находим определенный знак пунктуации – точку.

2) Токенизация (иногда – сегментация) по словам – это процесс разделения предложений на слова-компоненты. В английском и многих других языках, использующих ту или иную версию латинского алфавита, пробел – это неплохой разделитель слов

3) 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.

Python:

nltk.sent\_tokenize – токенизация по предложениям(когда на конце .)

nltk.word\_tokenize – токенизация по словам

**Лемматизация и стемминг**

Лемматизация – процесс приведения слова к нормальной форме(используется контекст и тд)

Стемминг – поиск основы слова, но более грубый(не использует контекст, зато быстро)

**Примеры:**  
dog, dogs, dog’s, dogs’ => dog

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

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.

**Стоп-слова**

Стоп-слова – это слова, которые выкидываются из текста до/после обработки текста. Когда мы применяем машинное обучение к текстам, такие слова могут добавить много шума, поэтому необходимо избавляться от нерелевантных слов.  
  
Стоп-слова это обычно понимают артикли, междометия, союзы и т.д., которые не несут смысловой нагрузки. При этом надо понимать, что не существует универсального списка стоп-слов, все зависит от конкретного случая.

Python:

nltk.download(“stopwords”)

import nltk

from nltk.corpus import stopwords

set(stopwords.words('english'))

**Регулярные выражения**

Регулярное выражение (регулярка, regexp, regex) – это последовательность символов, которая определяет шаблон поиска. Например:

* . – 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.

**Мешок слов**

Документ разбивается на предложения, предложения на слова. Берется по одному экземпляру слова и составляется словарь. Индикатором 1 или 0 помечается соответственно, есть слово в предложении или нет.

Когда создан словарь, следует оценить наличие слов. Мы уже рассматривали простой, бинарный подход (1 – есть слово, 0 – нет слова).  
  
Есть и другие методы:

1. Количество. Подсчитывается, сколько раз каждое слово встречается в документе.
2. Частотность. Подсчитывается, как часто каждое слово встречается в тексте (по отношению к общему количеству слов).

**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](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B5_%D0%9A%D1%83%D0%BB%D1%8C%D0%B1%D0%B0%D0%BA%D0%B0_%E2%80%94_%D0%9B%D0%B5%D0%B9%D0%B1%D0%BB%D0%B5%D1%80%D0%B0) 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](https://ru.wikipedia.org/wiki/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.)

Скоринг по TF-IDF растет пропорционально частоте появления слова в документе, но это компенсируется количеством документов, содержащих это слово.

Graphical user interface, text, application

Description automatically generated

TF (term frequency — частота слова) – отношение числа вхождений слова к общему числу слов документа.

A picture containing text, tool

Description automatically generated

IDF (inverse document frequency — обратная частота документа) — инверсия частоты, с которой некоторое слово встречается в документах коллекции.

A picture containing text

Description automatically generated

Text

Description automatically generated

vectorizer=TfidfVectorizer() – метод на питоне

**ПОЧЕМУ TF-IDF ИМЕЕТ СМЫСЛ**:

Речь идет о важности некоторого слова в конкретном документе. Обычная частота слова в документе(TF) может быть завышена, если слово встречается очень часто, значит информация о нем не особо важна. Однако, если добавлять IDF, может выявиться, что это слово встречается всего в одном документе, а значит оно для него важно(IDF имеет место за счет величины (кол-во всех документов/кол-во документов, в которых содержится данное слово). Но стоить заметить, что это чисто математическая величина, и на коротких дистанциях она ведет себя странно (в примере из нескольких никак не связанных между собой предложений, слово this встретилось в предложении “I like this movie, it's funny”; его оценка в этом предложении всего 0.38, однако если его убрать, смысл этого предложения теряется.)

**Заключение**

* NLP позволяет применять алгоритмы машинного обучения для текста и речи;
* NLTK (Natural Language Toolkit) – ведущая платформа для создания NLP-программ на Python;
* токенизация по предложениям – это процесс разделения письменного языка на предложения-компоненты;
* токенизация по словам – это процесс разделения предложений на слова-компоненты;
* лемматизация и стемминг преследуют цель привести все встречающиеся словоформы к одной, нормальной словарной форме;
* стоп-слова – это слова, которые выкидываются из текста до/после обработки текста;
* регулярное выражение (регулярка, regexp, regex) – это последовательность символов, которая определяет шаблон поиска;
* мешок слов – это популярная и простая техника извлечения признаков, используемая при работе с текстом. Она описывает вхождения каждого слова в текст.

**Алгоритм решения NLP задач**

1)Сегментация (деление текста на предложения- nltk.sent\_tokenize(прочитанный файл)

2)Токенизация (деление предложений на токены, то есть отдельные слова)

3) Дальше нужно вычислить признаки каждого токена. Вычислить контекстно-независимые и контекстно-завимые признаки токена. Это набор признаков, которые никак не зависят от окружающих наш токен других слов. Обычные **контекстно-независимые** признаки – это:

* эмбеддинги
* символьные признаки
* дополнительные признаки, специальные для конкретной задачи или языка

Примеры дополнительных признаков:

1)Часть речи или POS-тег (part of speech).

2)Позиция токена в тексте(Мы можем априори знать, что какая-то сущность чаще встречается в начале текста или наоборот в конце.)

Все вместе – эмбеддинги, символьные и дополнительные признаки – формируют вектор признаков токена, который не зависит от контекста.

**Контекстно – зависимые** признаки:

Контекстно-зависимые признаки токена — это набор признаков, который содержит информацию не только про сам токен, но и про его соседей. В классических алгоритмах люди часто просто шли «окном»: брали несколько (например, три) токенов до исходного и несколько токенов после, а затем вычисляли все признаки в таком окне. Такой подход ненадежен, так как важная информация для анализа может находиться на расстоянии, превышающем окно, соответственно, мы можем что-то пропустить.

Поэтому сейчас все контекстно-зависимые признаки вычисляются на уровне предложения стандартным образом: с помощью **двухсторонних рекуррентных нейросетей LSTM или GRU**. Чтобы получить контекстно-зависимые признаки токена из контекстно-независимых, контекстно-независимые признаки всех токенов предложения подаются в Bidirectional RNN (одно- или несколько- слойный). Выход Bidirectional RNN в i-ый момент времени и является контекстно-зависимым признаком i-того токена, который содержит информацию как о предыдущих токенах (т.к. эта информация содержится в i-м значении прямого RNN), так и о последующих (т.к. эта информация содержится в соответствующем значении обратного RNN).

Такой способ получения признаков называется **Pipeline NLP.**