NLP

Где можно увидеть эти записи?

«Выкл», «Стоп», «Ограничение скорости 60», «Дорожные работы» «Позвонить домой», «Включить классическую музыку» «Где находится ближайшая заправка?».

Или Вы получаете сообщение от клиента на испанском языке и переводите программой-переводчиком. И Вы отвечаете на английском языке, зная, что клиент может легко перевести его на испанский. Вы не уверены в том, на каком языке написано полученное сообщение, но программа моментально определяет это за вас и переводит текст на английский. Далее можно перевести на тот язык, которую Вы хорошо знаете.

Все перечисленное — примеры общения на естественном языке в форме текста, голоса, видео, знака жестов и в других формах на разных языках: английском, испанском, русском и других.

В Data Science имеются ряд средств обработки естественного языка (NLP). Обработка естественного языка может применяться к текстовым коллекциям, состоящим из сообщений в Facebook, диалогов, рецензий на фильмы, исторических документов, новостей, протоколов собраний и т. д.

Текстовая коллекция называется корпусом.

TextBlob

TextBlob библиотека построена на базе NLP-библиотек NLTK и pattern. С ее помощи можно:

- разбить текст на содержательные блоки (например, слова и числа);
- пометить части речи — идентификация части речи каждого слова (существительное, глагол, прилагательное и т. д.);
- извлечь именные конструкции — обнаружение групп слов, представляющих имена существительные;
- анализировать эмоциональную окраску — определение положительной, отрицательной или нейтральной окраски текста;
- выполнить перевод на другие языки и распознавание языка на базе Google Translate;
- выполнить формообразование — образование множественного и единственного числа.
- проверить орфографию и исправить ошибки;

- выделить основы — исключение приставок, суффиксов и т. д.; например, при выделении основы из слова «varieties» будет получен результат «varieti»;
- лемматизацию — аналог выделения основы, но с формированием реальных слов на основании контекста исходных слов; например, результатом лемматизации «varieties» является слово «variety»;
- определить частоты слов — определение того, сколько раз каждое слово встречается в корпусе;
- выполнить поиск слов, синонимов и антонимов;
- устранить игнорируемые слова — исключение таких слов, как a, an, the, I, we, you и т. д., с целью анализа важных слов в корпусе;
- выполнить n-граммы — построение множеств последовательно идущих слов в корпусе для выявления слов, часто располагающихся

Чтобы установить TextBlob, выбрать Anaconda Prompt в меню Пуск и выполните команду:

conda install -c conda-forge textblob

После завершения установки выполните загрузку корпусов NLTK, используемых TextBlob:

ipython -m textblob.download_corpora

In [ ]:
```
# [Jupyter Notebook]
import sys
!{sys.executable} -m pip install textblob
!{sys.executable} -m textblob.download_corpora
```

https://guides.library.upenn.edu/penntdm/python/textblob#:~:text=Installation,package%20index%20and%20setup%20tools.&text=If%20you%20on
(https://guides.library.upenn.edu/penntdm/python/textblob#:~:text=Installation,package%20index%20and%20setup%20tools.&text=If%20you%20or

Какие корпуса будут установлены?

- Brown Corpus (создан в Университете Брауна1) — для пометки частей речи.
- Punkt — для разбиения английских предложений на лексемы.
- WordNet — для определений слов, синонимов и антонимов.
- Averaged Perceptron Tagger — для пометки частей речи.
- conll2000 — для разбиения текста на компоненты (существительные, глаголы и т. д.). Имя conll2000 происходит от конференции, на которой были созданы эти данные (Conference on Computational Natural Language Learning).
- Movie Reviews — для анализа эмоциональной окраски.

Проект «Гутенберг»

Для обработки текста можно воспользоваться бесплатными электронными книгами на сайте проекта «Гутенберг»: https://www.gutenberg.org (https://www.gutenberg.org)

Сайт содержит свыше 57 000 электронных книг различного формата, включая простые текстовые файлы.

«Ромео и Джульетта» Шекспира можно загрузить по ссылке: https://www.gutenberg.org/ebooks/1513 (https://www.gutenberg.org/ebooks/1513)

Сохраните этот файл с именем RomeoAndJuliet.txt в каталоге ch4, чтобы наши примеры правильно работали. Для анализа удалите текст перед строкой "THE TRAGEDY OF ROMEO AND JULIET", а также текст в конце файла, начиная с текста: End of the Project Gutenberg EBook of Romeo and Juliet, by William Shakespeare

Создание TextBlob

В модуле textblob имеется класс **TextBlob** для NLP-операций. С помощью данного класса создадим объект TextBlob с двумя предложениями:

In [2]:
```
from textblob import TextBlob
text = 'The weather is not  rainy today. Tomorrow looks like sunny.'
blob = TextBlob(text)
blob
```

Out[2]:
```
TextBlob("The weather is not  rainy today. Tomorrow looks like sunny.")
```

Объекты TextBlob (объекты Sentence и Word) поддерживают строковые методы и могут сравниваться со строками.

Разбиение текста на предложения и слова

Обработка естественного языка часто требует разбиения текста на лексемы перед выполнением других NLP-операций. TextBlob имеет удобные свойства для обращения к предложениям и словам. Используем свойство sentence для получения списка объектов Sentence:

```
In [3]: blob.sentences
```

```
Out[3]: [Sentence("The weather is not  rainy today."),
         Sentence("Tomorrow looks like sunny.")]
```

Свойство words возвращает объект WordList со списком объектов Word, представляющим каждое слово в TextBlob после удаления знаков препинания:

```
In [4]: blob.words
```

```
Out[4]: WordList(['The', 'weather', 'is', 'not', 'rainy', 'today', 'Tomorrow', 'looks', 'like', 'sunny'])
```

Пометка частей речи

В английском языке существуют восемь основных частей речи — существительные, местоимения, глаголы, прилагательные, наречия, предлоги, союзы и междометия (слова, выражающие эмоции, за которыми обычно следует знак препинания, например «Yes!» или «Ha!»). В каждой категории существует множество подкатегорий.

Некоторые слова имеют несколько значений — так, у слов «set» или «run» возможные значения исчисляются сотнями! Взглянув на определения слова «run» на сайте dictionary.com, вы увидите, что оно может быть глаголом, существительным, прилагательным или частью глагольной группы. Одно из важных применений пометки частей речи — определение смысла слова из множества возможных вариантов. Эта операция играет важную роль в «понимании» естественных языков компьютерами.

Свойство tags возвращает список кортежей, каждый из которых содержит слово и строку, представляющую его пометку части речи:

In [7]:
```
blob.tags
```

Out[7]:
```
[('The', 'DT'),
 ('weather', 'NN'),
 ('is', 'VBZ'),
 ('not', 'RB'),
 ('rainy', 'JJ'),
 ('today', 'NN'),
 ('Tomorrow', 'NNP'),
 ('looks', 'VBZ'),
 ('like', 'IN'),
 ('sunny', 'NN')]
```

В выводе приведенного фрагмента:

- The имеет пометку DT — определитель
- weather, today, sunny имеют пометку NN — признак существительного
- is, looks имеют пометку VBZ — признак глагола третьего лица единственного числа
- rainy имеет пометку JJ — признак прилагательного
- Tomorrow имеет пометку NNP — признак имени собственного единственного числа
- like имеет пометку IN — признак подчинительного союза или предлога
- partly, not имеет пометку RB - признак наречия

In [ ]:

```
List of the tags:
DT-determiner
EX-existential there (like: “there is” … think of it like “there exists”)
FW-foreign word
IN-preposition/subordinating conjunction
JJ-adjective ‘big’
JJR-adjective, comparative ‘bigger’
JJS-adjective, superlative ‘biggest’
LS-list marker 1)
MD-modal could, will
NN-noun, singular ‘desk’
RBR-adverb, comparative better
RBS-adverb, superlative best
NNS-noun plural ‘desks’
NNP-proper noun, singular ‘Harrison’
NNPS-proper noun, plural ‘Americans’
PDT-predeterminer ‘all the kids’
POS-possessive ending parent‘s
PRP- personal pronoun I, he, she
PRP$-possessive pronoun my, his, hers
RB-adverb very, silently,
RP-particle give up
TO- to go ‘to‘ the store.
UH-interjection
VB-verb, base form take
VBD-verb, past tense, took
CC-coordinating conjunction
CD-cardinal digit
VBG-verb, gerund/present participle taking
VBN-verb, past participle taken
VBP-verb, sing. present, non-3d take
VBZ-verb, 3rd person sing. present takes
WDT-wh-determiner which
WP-wh-pronoun who, what
WP$-possessive wh-pronoun whose
WRB-wh-adverb where, when
```

Извлечение именных конструкций

Допустим, вы собираетесь купить водные лыжи и ищете информацию о них в интернете — скажем, по строке «best water ski». В данном случае группа «water ski» является именной конструкцией. Если поисковая система не сможет правильно выделить именную конструкцию, то вероятно, что полученные результаты не будут оптимальными. Попробуйте поискать информацию по строкам «best water», «best ski» и «best water ski» и посмотрите, что вы найдете. Свойство noun_phrases класса TextBlob возвращает объект WordList со списком объектов Word — по одному для каждой именной конструкции в тексте:

```
In [5]: blob.noun_phrases
```

```
Out[5]: WordList(['tomorrow'])
```

Анализ эмоциональной окраски

Некоторые компании могут использовать их для определения того, как потребители отзываются в интернете об их продуктах — положительно или отрицательно. Будем считать слово «good» положительным, а слово «bad» — отрицательным. Но сам факт присутствия слова «good» или «bad» в предложении еще не означает, что предложение в целом эмоционально окрашено положительно или отрицательно. Например, предложение

The food is not good.

обладает отрицательной эмоциональной окраской. Аналогичным образом предложение

The movie was not bad.

явно обладает положительной эмоциональной окраской, хотя, возможно, и не настолько положительной, как предложение вида

The movie was excellent!

Анализ эмоциональной окраски — сложная задача. Тем не менее TextBlob содержит предварительно обученные модели для ее выполнения.

In [9]:
```
blob.sentiment
```

Out[9]:
```
Sentiment(polarity=0.0, subjectivity=0.0)
```

В показанном выводе полярность (показатель polarity) означает эмоциональную окраску. Значение 0.0 соответствует нейтральной эмоциональной окраске.

In [10]:
```
for sentence in blob.sentences:
    print(sentence.sentiment)
```

```
Sentiment(polarity=0.0, subjectivity=0.0)
Sentiment(polarity=0.0, subjectivity=0.0)
```

In [6]:
```
from textblob.sentiments import NaiveBayesAnalyzer
blob = TextBlob(text, analyzer=NaiveBayesAnalyzer())
blob
```

Out[6]:
```
TextBlob("The weather is not  rainy today. Tomorrow looks like sunny.")
```

In [7]:
```
blob.sentiment
```

Out[7]:
```
Sentiment(classification='pos', p_pos=0.7542783346449297, p_neg=0.24572166535506929)
```

В данном случае общая эмоциональная окраска классифицируется как положительная (classification='pos'.) Свойство p_pos объекта Sentiment показывает, что текст TextBlob положителен на 66,34%, а свойство p_neg показывает, что текст TextBlob отрицателен на 33,65%.

А теперь получим данные об эмоциональной окраске каждого объекта Sentence:

In [8]:
```
for sentence in blob.sentences:
    print(sentence.sentiment)
```

```
Sentiment(classification='pos', p_pos=0.5015554841425046, p_neg=0.4984445158574952)
Sentiment(classification='pos', p_pos=0.7531233178323515, p_neg=0.24687668216764802)
```

Распознавание языка и перевод

Перевод на другой язык — довольно сложная задача из области обработки естественного языка и искусственного интеллекта. Благодаря новейшим достижениям в области машинного обучения, искусственного интеллекта и обработки естественных языков такие сервисы, как Google Translate (100+ языков) и DeepL могут мгновенно переводить тексты на другие языки.

Автоматический перевод также очень удобен для людей, посещающих зарубежные страны. Они могут использовать приложение-переводчик для перевода меню, дорожных знаков и т. д. Также ведутся исследования в области перевода речи, чтобы вы могли общаться в реальном времени с людьми, которые не знают ваш основной язык. Некоторые современные смартфоны в сочетании с наушниками-вкладышами обеспечивают практически мгновенный перевод со многих языков.

- https://www.skype.com/en/features/skype-translator/ (https://www.skype.com/en/features/skype-translator/)
- https://www.microsoft.com/en-us/translator/business/live/ (https://www.microsoft.com/en-us/translator/business/live/)
- https://www.telegraph.co.uk/technology/2017/10/04/googles-new-headphones-can-translate-foreignlanguages-real/ (https://www.telegraph.co.uk/technology/2017/10/04/googles-new-headphones-can-translate-foreignlanguages-real/)
- https://store.google.com/us/product/google_pixel_buds?hl=en-US (https://store.google.com/us/product/google_pixel_buds?hl=en-US)
- http://www.chicagotribune.com/bluesky/originals/ct-bsi-google-pixel-buds-review-20171115-story.html (http://www.chicagotribune.com/bluesky/originals/ct-bsi-google-pixel-buds-review-20171115-story.html)

```
blob.detect_language()

spanish = blob.translate(to='es')

chinese = blob.translate(to='zh')

chinese = blob.translate(from_lang='en', to='zh')
```

Формообразование: образование единственного и множественного числа

У одного слова может быть несколько разных форм — например, единственное или множественное число («person» и «people») или разное время глагола («run» и «ran»). Возможно, при вычислении частот слов в тексте вы захотите сначала преобразовать все формы слов к одной форме для получения более точных данных частот. Word и WordList поддерживают преобразование слов к форме единственного или множественного числа. Попробуем выполнить это преобразование с парой объектов Word:

In [9]:
```
from textblob import Word
index = Word('index')
index.pluralize()
```

Out[9]:
```
'indices'
```

In [10]:
```
cats = Word('cats')
cats.singularize()
```

Out[10]:
```
'cat'
```

In [11]:
```
from textblob import TextBlob
animals = TextBlob('dog cat fish bird').words
animals.pluralize()
```

Out[11]:
```
WordList(['dogs', 'cats', 'fish', 'birds'])
```

Проверка орфографии и исправление ошибок

Для задач обработки естественного языка очень важно, чтобы текст был свободен от орфографических ошибок. Программные пакеты для ввода и редактирования текста, такие как Microsoft Word, Google Docs и им подобные, автоматически проверяют орфографию во время ввода текста и обычно подчеркивают неправильные слова красной волнистой линией. Другие инструменты позволяют выполнить проверку орфографии вручную. Вы можете проверить орфографию текста в Word методом spellcheck этого объекта. Этот метод возвращает список кортежей, содержащих возможные варианты написания слова и уровень достоверности. Предположим, вы хотели написать слово «they», но случайно ввели его в виде «theyr». Результаты проверки предлагают два возможных исправления, при этом вариант 'they' имеет наивысший уровень достоверности:

In [12]:
```
from textblob import Word
word = Word('theyr')
%precision 2
```

Out[12]:
```
'%.2f'
```

In [13]:
```
word.spellcheck()
```

Out[13]:
```
[('they', 0.57), ('their', 0.43)]
```

Объекты TextBlob, Sentence и Word содержат метод correct, который можно вызвать для исправления ошибки. Вызов correct для Word возвращает правильно написанное слово с наибольшим уровнем достоверности (по данным проверки орфографии):

In [14]:
```
word.correct() # Выбирает слово с наибольшим уровнем достоверности
```

Out[14]:
```
'they'
```

Вызов correct для объекта TextBlob или Sentence проверяет орфографию каждого слова. Для каждого неправильного слова correct заменяет его правильно написанным вариантом с наибольшим уровнем достоверности:

In [2]:
```
from textblob import TextBlob
from textblob import Word
sentence = TextBlob('Ths sentense has missplled wrds.')
sentence.correct()
```

Out[2]:
```
TextBlob("The sentence has misspelled words.")
```

Нормализация: выделение основы и лемматизация

В результате выделения основы из слова удаляется префикс или суффикс и остается только основа, которая может быть реальным словом. Лемматизация выполняется аналогичным образом, но ее результатом является осмысленная часть речи, то есть реальное слово. Выделение основы и лемматизация относятся к операциям нормализации, готовящим слова для анализа. Например, перед вычислением статистики вхождения слов в корпусе текста все слова могут быть преобразованы к нижнему регистру, чтобы слова, начинающиеся с букв нижнего и верхнего регистра, обрабатывались одинаково. Иногда для представления разных форм слова приходится использовать только корень. Например, в некотором приложении все следующие слова могут рассматриваться как слово «program»: program, programs, programmer, programming и programmed (и, возможно, английские варианты написания — скажем, programmes).

У объектов Word и WordList для выделения основы и лемматизации используются методы stem и lemmatize. Попробуем использовать их с Word:

In [17]:
```
from textblob import Word
word = Word('varieties')
word.stem()
```

Out[17]:
```
'varieti'
```

In [18]:
```
word.lemmatize()
```

Out[18]:
```
'variety'
```

Частоты слов

Различные методы выявления сходства между документами основаны на частотах вхождения слов. TextBlob подсчитывает частоты автоматически. Начнем с загрузки электронного текста пьесы Шекспира «Ромео и Джульетта».

In [19]:
```
import codecs
from pathlib import Path
import codecs
file = codecs.open( "RomeoAndJuliet.txt", "r", "utf-8" )
data = file.read()
blob = TextBlob(data)
```

Частоты вхождения слов в тексте TextBlob хранятся в словаре word_counts. Подсчитаем вхождения некоторых слов в пьесе:

In [20]:
```
blob.word_counts['juliet']
```

Out[20]:
```
190
```

In [21]:
```
 blob.word_counts['romeo']
```

Out[21]:
```
315
```

In [22]:
```
blob.word_counts['thou']
```

Out[22]:
```
278
```

Если вы уже разобрали TextBlob в список WordList, то для подсчета вхождений конкретных слов в список можно воспользоваться методом count:

In [23]:
```
blob.words.count('joy')
```

Out[23]: 14

In [24]:
```
blob.noun_phrases.count('lady capulet')
```

Out[24]: 46

Получение определений, синонимов и антонимов из WordNet

WordNet– база данных слов, созданная в Принстонском университете. Библиотека TextBlob использует интерфейс WordNet библиотеки NLTK, позволяющий искать определения слов, синонимы и антонимы.

https://www.nltk.org/api/nltk.corpus.reader.html#module-nltk.corpus.reader.wordnet (https://www.nltk.org/api/nltk.corpus.reader.html#module-nltk.corpus.reader.wordnet)

In [25]:
```
# Получение определений
# создания объекта Word:
from textblob import Word
happy = Word('happy')
```

Свойство definitions класса Word возвращает список всех определений слова в базе данных WordNet. Также имеется метод define, который позволяет передать в аргументе часть речи, с тем чтобы вы могли получить определения, соответствующие только этой части слова.

In [26]:
```
happy.definitions
```

Out[26]:
```
['enjoying or showing or marked by joy or pleasure',
 'marked by good fortune',
 'eagerly disposed to act or to be of service',
 'well expressed and to the point']
```

Получение синонимов

Наборы синонимов Word доступны в свойстве synsets. Результат представляет собой список объектов Synset:

In [27]:
```
happy.synsets
```

Out[27]:
```
[Synset('happy.a.01'),
 Synset('felicitous.s.02'),
 Synset('glad.s.02'),
 Synset('happy.s.04')]
```

Каждый объект Synset представляет группу синонимов. В записи happy.a.01:

- happy — лемматизированная форма исходного объекта Word (в данном случае они совпадают).
- a — часть речи: a — прилагательное, n — существительное, v — глагол, r — наречие, s — прилагательное-сателлит. Многие наборы синонимов прилагательных в WordNet имеют сателлитные наборы синонимов, представляющие похожие прилагательные.
- 01 — индекс, начинающийся с 0. Многие слова обладают несколькими смыслами; значение является индексом соответствующего смысла в базе данных WordNet

In [28]:
```
synonyms = set()
for synset in happy.synsets:
    for lemma in synset.lemmas():
        synonyms.add(lemma.name())
synonyms
```

Out[28]:
```
{'felicitous', 'glad', 'happy', 'well-chosen'}
```

Получение антонимов

Если слово, представленное объектом Lemma, имеет антонимы в базе данных WordNet, то вызов метода antonyms объекта Lemma возвращает список объектов Lemma, представляющих антонимы (или пустой список, если в базе данных нет ни одного антонима). Во фрагменте показано, что для 'happy' было найдено четыре объекта Synset. Сначала найдем объекты Lemma для Synset с индексом 0 в списке synsets:

In [29]:
```
lemmas = happy.synsets[0].lemmas()
lemmas
```

Out[29]:
```
[Lemma('happy.a.01.happy')]
```

В данном случае lemmas возвращает список из одного элемента Lemma. Мы можем проверить, содержит ли база данных какие-либо антонимы для этого объекта Lemma:

In [30]:
```
lemmas[0].antonyms()
```

Out[30]:
```
[Lemma('unhappy.a.01.unhappy')]
```

Удаление игнорируемых слов

Игнорируемые слова (стоп-слова) — часто встречающиеся в тексте слова, которые часто удаляются из текста перед анализом, поскольку обычно не несут полезной информации. Ниже приведен список игнорируемых слов английского языка из NLTK, возвращаемый функцией words1 модуля stopwords (которой мы вскоре воспользуемся на практике):

Список игнорируемых слов английского языка из NLTK

Прежде чем использовать списки игнорируемых слов NLTK, их необходимо загрузить при помощи функции download модуля nltk

In [44]:
```
pip install nltk
```

```
Requirement already satisfied: nltk in c:\programdata\anaconda3\envs\example\lib\site-packages (3.8.1)
Requirement already satisfied: regex>=2021.8.3 in c:\programdata\anaconda3\envs\example\lib\site-packages (from nltk)
(2024.4.16)
Requirement already satisfied: tqdm in c:\programdata\anaconda3\envs\example\lib\site-packages (from nltk) (4.66.4)
Requirement already satisfied: joblib in c:\programdata\anaconda3\envs\example\lib\site-packages (from nltk) (1.2.0)
Requirement already satisfied: click in c:\programdata\anaconda3\envs\example\lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: colorama in c:\programdata\anaconda3\envs\example\lib\site-packages (from click->nltk)
(0.4.5)
Requirement already satisfied: importlib-metadata in c:\programdata\anaconda3\envs\example\lib\site-packages (from cl
ick->nltk) (4.11.3)
Requirement already satisfied: zipp>=0.5 in c:\programdata\anaconda3\envs\example\lib\site-packages (from importlib-m
etadata->click->nltk) (3.8.0)
Requirement already satisfied: typing-extensions>=3.6.4 in c:\programdata\anaconda3\envs\example\lib\site-packages (f
rom importlib-metadata->click->nltk) (4.3.0)
Note: you may need to restart the kernel to use updated packages.
```

In [46]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Error loading stopwords: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
```

Out[46]:
```
False
```

Импортируйте stopwords из модуля nltk.corpus, а затем используйте метод words класса stopwords для загрузки списка игнорируемых слов 'english':

In [2]:
```
from nltk.corpus import stopwords
stops = stopwords.words('english')
```

Затем создадим объект TextBlob, из которого будут удаляться игнорируемые слова:

In [32]:
```
from textblob import TextBlob
blob = TextBlob('Today is a beautiful day.')
```

Наконец, чтобы удалить игнорируемые слова, используйте слова TextBlob в трансформации списка, которая добавляет каждое слово в полученный список только в том случае, если слово не входит в stops:

In [33]:
```
[word for word in blob.words if word not in stops]
```

Out[33]:
```
['Today', 'beautiful', 'day']
```

n-граммы

N-грамма представляет собой последовательность из n текстовых элементов, например букв в словах или слов в предложении. В обработке естественных языков n-граммы могут использоваться для выявления букв или слов, часто располагающихся рядом друг с другом. Для текстового ввода это поможет предсказать следующую букву или слово, вводимое пользователем, например при завершении элементов в IPython по нажатии клавиши Tab или при вводе текстового сообщения в вашем любимом месседжере для смартфона. При преобразовании речи в текст n-граммы могут использоваться для повышения качества текста. N-граммы представляют собой форму лексической солидарности, то есть расположения букв или слов рядом друг с другом.

Метод ngrams объекта TextBlob выдает список WordList n-грамм, которые по умолчанию имеют длину 3 (они называются триграммами). Передавая ключевой аргумент n, вы сможете получать n-граммы любой длины по своему усмотрению.

In [34]:
```
from textblob import TextBlob
text = 'Today is a beautiful day. Tomorrow looks like bad weather.'
blob = TextBlob(text)
blob.ngrams()
```

Out[34]:
```
[WordList(['Today', 'is', 'a']),
 WordList(['is', 'a', 'beautiful']),
 WordList(['a', 'beautiful', 'day']),
 WordList(['beautiful', 'day', 'Tomorrow']),
 WordList(['day', 'Tomorrow', 'looks']),
 WordList(['Tomorrow', 'looks', 'like']),
 WordList(['looks', 'like', 'bad']),
 WordList(['like', 'bad', 'weather'])]
```

In [35]:
```
blob.ngrams(n=2)
```

Out[35]:
```
[WordList(['Today', 'is']),
 WordList(['is', 'a']),
 WordList(['a', 'beautiful']),
 WordList(['beautiful', 'day']),
 WordList(['day', 'Tomorrow']),
 WordList(['Tomorrow', 'looks']),
 WordList(['looks', 'like']),
 WordList(['like', 'bad']),
 WordList(['bad', 'weather'])]
```

Визуализация частот вхождения слов средствами Pandas

In [36]:
```
# Загрузка данных
import codecs
from pathlib import Path
import codecs
file = codecs.open( "RomeoAndJuliet.txt", "r", "utf-8" )
data = file.read()
blob = TextBlob(data)
# Загрузка списка игнорируемых слов NLTK
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
```

Чтобы построить визуализацию частот 20 слов, необходимо знать все слова и их частоты. Вызовем метод items словаря blob.word_counts, чтобы получить список кортежей «слово-частота»:

In [37]:
```
items = blob.word_counts.items()
```

Воспользуемся трансформацией списка для удаления кортежей, содержащих стоп-слова:

In [38]:
```
items = [item for item in items if item[0] not in stop_words]
```

Выражение item[0] получает слово из каждого кортежа, чтобы проверить, входит ли оно в список stop_words.

Чтобы определить 20 наиболее частых слов, отсортируем кортежи в items по убыванию частоты. Мы можем воспользоваться встроенной функцией sorted с аргументом key для сортировки кортежей по элементу частоты в каждом кортеже. Чтобы задать элемент кортежа для выполнения сортировки, используйте функцию itemgetter из модуля operator стандартной библиотеки Python:

In [39]:
```
from operator import itemgetter
sorted_items = sorted(items, key=itemgetter(1), reverse=True)
```

В процессе упорядочения элементов items функция sorted обращается к элементу с индексом 1 в каждом кортеже с использованием выражения itemgetter(1). Ключевой аргумент reverse=True означает, что кортежи должны сортироваться по убыванию.

In [40]:
```
top20 = sorted_items[1:21]
```

Затем сегмент используется для получения 20 самых частых слов из sorted_ items. Когда объект TextBlob проводит разбиение корпуса на лексемы, он разбивает все сокращенные формы по апострофам и подсчитывает общее количество апострофов как одно из «слов». Текст «Ромео и Джульетты» содержит множество сокращенных форм. Если вывести sorted_items[0], вы увидите, что это самое часто встречающееся «слово» с 867 вхождениями. Выводиться должны только слова, поэтому мы игнорируем элемент 0 и получаем сегмент с элементами с 1 по 20 списка sorted_items

Затем преобразуем список кортежей top20 в коллекцию DataFrame библиотеки Pandas для удобства его представления в визуальном виде:

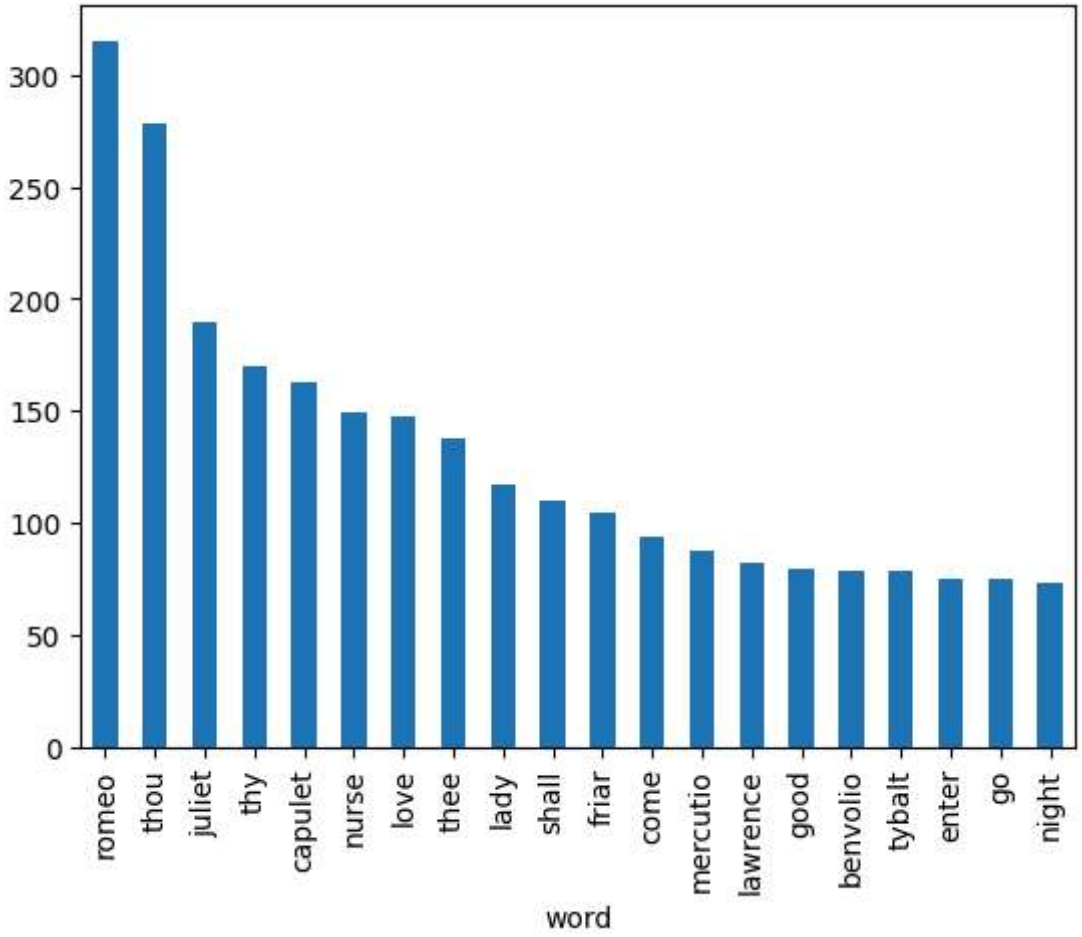
In [41]:
```
import pandas as pd
df = pd.DataFrame(top20, columns=['word', 'count'])
df
```

Out[41]:

	word	count
0	romeo	315
1	thou	278
2	juliet	190
3	thy	170
4	capulet	163
5	nurse	149
6	love	148
7	thee	138
8	lady	117
9	shall	110
10	friar	105
11	come	94
12	mercutio	88
13	lawrence	82
14	good	80
15	benvolio	79
16	tybalt	79
17	enter	75
18	go	75
19	night	73

Визуализация DataFrame

```
In [42]: axes = df.plot.bar(x='word', y='count', legend=False)
```



Метод bar создает и выводит гистограмму Matplotlib. Взглянув на исходную гистограмму, которая появляется на экране, можно заметить, что некоторые из ее столбцов усечены. Чтобы решить эту проблему, используйте функцию gcf (Get Current Figure) для получения рисунка Matlpotlib, выведенного pandas, а затем вызовите метод tight_layout. Вызов «сжимает» гистограмму, чтобы все ее компоненты поместились в окне:

In [58]:
```
import matplotlib.pyplot as plt
plt.gcf().tight_layout()
```

```
<Figure size 640x480 with 0 Axes>
```