Практика 2.3.1. Основные понятия компьютерной лингвистики. Лингвистические данные. Сложности обработки и анализа естественного языка

Первым шагом в автоматической обработке текста обычно становится **токенизация** – разбиение текста на более мелкие части – **токены**. К токенам относятся как слова, так и знаки пунктуации.

Обработка естественного языка используется для создания приложений, таких как классификация текста, интеллектуальный чат-бот, сентиментальный анализ, языковой перевод и т.д. Для достижения вышеуказанной цели становится жизненно важным понять закономерность в тексте. Эти токены очень полезны для нахождения таких паттернов, а также рассматриваются как базовый шаг для дальнейшего анализа.

После токенизации обычно создается словарь, в который заносятся уникальные лексемы, встретившиеся в корпусе или тексте. На этих этапах исследователи сталкиваются с несколькими проблемами.

Проблема 1 – языки с богатой морфологией. Это языки с развитыми системами склонений и спряжений слов. При работе с текстами на этих языках сложность возникает при составлении словаря, когда нужно найти и объединить все словоформы одной и той же лексемы. Пример – русский язык, в котором есть падежи. При переводе слова в векторное пространство нужно учитывать, что *стол, столу и столом* – это одно слово в разных падежных формах, а не 3 уникальных лексемы. Чтобы решить эту задачу, текст можно предварительно лемматизировать, или применить стемминг (от английского *stem* – стебель), то есть просто отрезать у слов окончания.

Проблема 2 – языки с продуктивным сложением основ. В германских языках (в английском, немецком, шведском и т.д.) очень продуктивно образуются новые сложные слова. Значения таких слов выводятся из значения их элементов, их можно создавать бесконечно долго, и большинство из них не зафиксировано в «бумажном» словаре.

Пример шведского названия гаечного ключа для колеса мотоцикла

При работе с этими языками сложность также возникает на этапе составления словаря. При составлении словаря модели ориентируются на частотность (например, сохраняем слово, если оно встретилось чаще пяти раз), поэтому не будут запоминать такое длинное и сложное слово.

Проблема 3 – определение границ слова. Современные лингвисты до сих пор не могут придумать универсальное определение понятию слово и в каждой конкретной ситуации объясняют его по-разному. Для нас, привыкших к языкам европейского типа, слово – это набор букв между пробелами и знаками препинания. По таким разделителям компьютер тоже может легко найти слово.

Но в английском языке многие сложные слова пишутся раздельно, а в японском, наоборот, между словами вообще нет пробелов. Поэтому универсальный токенизатор создать нелегко.

Пример. Исследовать роль половой принадлежности предложений в тексте на английском языке для определения проявлений мужчин и женщин в разных контекстах.

Рассмотрим простую модель, использующую лингвистические признаки для выявления преобладающего рода (мужское/женское) во фрагменте текста. В 2013 г. Нил Карен (Neal Caren), доцент кафедры социологии в Университете города Чаппел-Хилл (штат Северная Каролина), опубликовал в блоге статью, где исследовал роль половой принадлежности в новостях для определения проявлений мужчин и женщин в разных контекстах. Он применил гендерный анализ к текстам статей, опубликованных в New-York Times, и выяснил, что слова в мужском и женском роде появляются в совершенно разных контекстах, что способно усиливать гендерные предубеждения. Особенно интересно, в этом анализе, слова в женском и мужском роде использовались для создания частотной оценки мужских и женских признаков. Реализацию подобного анализа на Python можно начать с определения наборов слов, различающих предложения, в которых речь идет о мужчинах и женщинах. Для простоты допустим, что всякое предложение может классифицироваться как рассказывающее о мужчинах, о женщинах, о мужчинах и женщинах и неизвестно о ком (потому, что предложение может рассказывать о чем-то другом, не о мужчинах и не о женщинах, а также потому, что наши множества слов не будут являться исчерпывающими).

Возьмем для анализа следующий текст:

Оригинал	**Перевод**
With apologies to James Brown, the hardest working people in show business may well be ballet dancers. And at New York City Ballet, none work harder than the dancers in its lowest rank, the corps de ballet. During the first week of the company's winter season, Claire Kretzschmar, 24, a rising corps member,	*Приносим извинения Джеймсу Брауну, но самые трудолюбивые люди в шоу-бизнесе вполне могут быть артистами балета. А в New York City Ballet никто не работает усерднее, чем танцоры самого низкого ранга, кордебалета. В течение первой недели зимнего сезона труппы 24-летняя Клэр*

danced in all seven performances, appearing in five ballets, sometimes changing costumes at intermission to dance two roles in a night. But her work onstage did not even begin to capture the stamina required to be in the corps. Spending a week shadowing Ms. Kretzschmar was exhausting – she gave new meaning to the idea of being on your feet all day. Twelve-hour days at the David H. Koch Theater, the company's Lincoln Center home, were hardly unusual: Company class each morning was followed by back-to-back-to-back rehearsals, with occasional breaks for costume fittings or physical therapy, and then by the hair-makeup-costume-dance routine of daily performances.	*Кречмар, восходящая участница труппы, танцевала во всех семи спектаклях, появлялась в пяти балетах, иногда меняя костюмы в антракте, чтобы танцевать две роли за ночь. Но работа на сцене никак не сказывалась на ее состоянии, что позволяло оставаться в форме. Провести неделю, следя за мисс Кречмар, было утомительно – она придала новый смысл идее быть на ногах весь день. Двенадцатичасовой рабочий день в Театре Дэвида Х. Коха, доме компании Lincoln Center, вряд ли был чем-то необычным: каждое утро занятия компании сопровождались репетициями спина к спине, с редкими перерывами на примерку костюмов или физиотерапию, а затем ежедневными выступлениями с прической, макияжем, костюмами и танцами.*

1) Определим по тексту наборы слов, которые будем использовать для распознавания предложений, в которых речь идет о мужчинах или женщинах, т.е. создадим словари, характеризующие принадлежность предложения к мужскому или женскому роду.

Оригинал	**Перевод**
`MALE_WORDS = set(['guy','spokesman', 'chairman',"men's",'men','him', "he's", 'his','boy','boyfriend', 'boyfriends', 'boys','brother','brothers','dad', 'dads','dude','father','fathers', 'fiance','gentleman','gentlemen', 'god', 'grandfather','grandpa','grandson', 'groom','he','himself','husband', 'husbands','king','male','man','mr', 'nephew','nephews','priest', 'prince','son','sons','uncle', 'uncles', 'waiter','widower','widowers']) # Множество мужских слов`	MALE_WORDS = {парень, представитель, председатель, мужской, мужчина, он, ему, его, мальчик, бойфренд, бойфренды, мальчики, брат, братья, папа, папы, чувак, отец, отцы, жених, джентльмен, джентльмены, бог, дедушка, старик, внук, жених, он, сам, муж, мужья, король, мужчины, мистер, господин, племянник, племянники, священник, принц, сын, сыновья, дядя, дяди, официант, вдовец, вдовцы}
`FEMALE_WORDS = set([ 'heroine', 'spokeswoman','chairwoman',`	FEMALE_WORDS = {героиня,

"women's", 'actress','women', "she's",'her','aunt','aunts', 'bride', 'daughter','daughters','female', 'fiancee','girl','girlfriend', 'girlfriends','girls','goddess', 'granddaughter','grandma', 'grandmother', 'herself','ladies', 'lady', 'mom','moms','mother','mothers', 'mrs','ms','niece','nieces', 'priestess','princess','queens', 'she', 'sister','sisters','waitress', 'widow','widows','wife','wives', 'woman']) # Множество женских слов	представительница, председательница, женщина, актриса, женщины, она, ее, тетя, тети, невеста, дочь, дочери, женщина, невеста, девочка, подруга, подруги, девочки, богиня, внучка, бабка, бабушка, сама, дамы, леди, мама, мамы, мать, матери, миссис, мисс, племянница, племянницы, жрица, принцесса, королевы, она, сестра, сестры, официантка, вдова, вдовы, жена, жены, женщина}

Теперь, когда у нас есть множества слов-признаков, характеризующих половую принадлежность, нам нужен метод определения гендерного класса предложения.

2) Создадим функцию `genderize`, которая подсчитывает количество слов в предложении, попадающих в наши списки MALE_WORDS и FEMALE_WORDS. Если предложение содержит только слова из MALE_WORDS, оно классифицируется как мужское. Предложение, содержащее только слова из FEMALE_WORDS, классифицируется как женское. Если предложение содержит мужские и женские слова, отнесем его к категории двуполых; а если в нем нет ни мужских, ни женских слов, определим его как имеющее неизвестный род.

```
def genderize(words): #Функция подсчета количества слов в преложении
  mwlen = len(MALE_WORDS.intersection(words)) #Запишем в переменную
mwlen количество мужских слов, используя функцию пересечения множест
в
  fwlen = len(FEMALE_WORDS.intersection(words)) #В переменную fwlen
количество женских слов
  if mwlen > 0 and fwlen == 0: #Если количество мужских слов больше
нуля и женских слов нет совсем, то функция вернет параметр "male" -
 мужской
    return "male"
  elif mwlen == 0 and fwlen > 0: #Аналогично, если женских слов не н
уль, а мужских слов нет, то вернет параметр "female" - женский
    return 'female'
  elif mwlen > 0 and fwlen > 0: #Если количество женских и мужских с
лов отлично от нуля, то предложение будет считать двуполым
    return 'both'
  else:
```

```
    return 'unknow' #В ином случае - неизвестно.
```

3) Нам также нужно подсчитать частоту слов, признаков рода и предложений во всем тексте статьи. Для этой цели можно использовать встроенный в Python класс `collections.Counters`.

`Counter` – это подкласс `dict`, который используется для подсчета объектов. Элементы хранятся в качестве ключей словаря, а количество объектов сохранено в качестве значения.

Функция `count_gender` принимает список предложений и использует функцию `genderize` для определения общего количества слов-признаков и классификации предложений. Для каждого предложения определяется его класс, а все слова в предложении считаются принадлежащими к этой категории:

```
from collections import Counter #Из встроенной библиотеки Python под
ключим функцию Counter для подсчета частоты слов
def count_gender(sentences):
    sents = Counter() #Задаем пустую переменную sents для подсчета к
оличества предложений определенного рода
    words = Counter() #Задаем пустую переменную words для подсчета к
оличества слов в предложении
    for sentence in sentences:
      gender = genderize(sentence) #Вызываем ранее созданную функцию
      sents[gender] += 1 #Считаем количество предложений определенно
го рода
      words[gender] += len(sentence) #Считаем количество слов в пред
ложении
    return sents, words
```

Инструменты для анализа текста. Модули и библиотеки для NLP

4) Наконец, чтобы задействовать функции определения гендерной принадлежности, нам нужен некоторый механизм, преобразующий исходный текст статьи в составляющие его предложения и слова, но для его создания необходимо познакомится с некоторыми библиотеками обработки и анализа естественного языка. Для Python существует множество библиотек и расширений, предоставляющих необходимые инструменты для анализа текста, однако большая их часть предназначена для работы с английским языком. Это одна из распространённых проблем обработки и анализа естественного языка. В наше время не существует достойных уникальных расширений или библиотек, способных работать с любым языком.

Одна из популярнейших библиотек, позволяющая проводить обработку естественного языка – это библиотека NLTK. Библиотека NLTK – пакет библиотек и программ для символьной и статистической обработки естественного языка, написанных на языке программирования Python. Содержит графические

представления и примеры данных. Сопровождается обширной документацией, включая книгу с объяснением основных концепций, стоящих за теми задачами обработки естественного языка, которые можно выполнять с помощью данного пакета. Обладающая огромным арсеналом функций, она подходит для решения почти любой задачи, связанной с анализом текста, а именно:

- Лексемизация – процесс разбиения предложений на изолированные лексемы.
- Векторизация – процесс преобразования нечисловых данных (например, текста, изображений и т.д.) в векторное представление, к которому можно применить методы машинного обучения.
- Лемматизация – это процесс преобразования слова в его базовую форму.
- Стемминг – это частица алгоритма поиска, чтобы найти основу словоформы. Такой поиск предназначен искать слова в морфологическом изменении.

К сожалению, не смотря на свою популярность и широкий функционал, NLTK в первую очередь предназначена для работы с английским языком, а следовательно все методы этой библиотеки работают с наибольшей отдачей только с англоязычными словами. В качестве примера рассмотрим небольшую программу для стемминга слов:

```
from nltk.stem import SnowballStemmer

words = ["game","gaming","gamed","games"]

ps = SnowballStemmer(language="english")

for word in words:
    print(ps.stem(word))
```

[Вывод] game
game
game
game

Данный программный код позволяет определить основу слова «game» – игра. Как мы видим, в какой бы форме это слово не стояло, функция `SnowballStemmer` справляется с ней на отлично, однако если сделать что-то подобное для русского языка:

```
from nltk.stem import SnowballStemmer
words = ["игра","игровой","игры","играющий"]
ps = SnowballStemmer(language="russian")

for word in words:
    print(ps.stem(word))
```

[Вывод] игр
игров
игр
игра

то результат будет не такой впечатляющий. Несомненно, это связанно с особенностями языка, что является еще одной проблемой для естественной обработки. Каждый язык, существующий у нас в мире, уникален.

Но вернемся к нашему коду и продолжим выполнение практической задачи. Используя библиотеку NLTK, разобьем текст на предложения. Выделив отдельные предложения, мы сможем разбить их на лексемы, чтобы выявить отдельные слова и знаки пунктуации, и передать размеченный текст нашим функциям классификации для вывода процентов предложений и слов, относящихся к категории мужских, женских, двуполых и неизвестной принадлежности:

```
import nltk              #Подключаем библиотеку NLTK
nltk.download('punkt') #Скачиваем для нее нужное для работы расширен
ие

def parse_gender(text): #На основе этой библиотеки, создадим функции
 для разделения предложения на отдельные слова.
  sentences = [[word.lower() for word in nltk.word_tokenize(sentence
)] #Для этого в двух циклах разобьем наш текст на предложения, а
  for sentence in nltk.sent_tokenize(text)] #каждое предложения на с
лова.
  sents, words = count_gender(sentences) #Вызов ранее созданной функ
ции
  total = sum(words.values()) #В переменную total запишем все слова
из предложений определенного рода.
  for gender, count in words.items(): #И для каждой категории посчит
аем частоту слов в процентах.
    pcent = (count / total) * 100
    nsents = sents[gender]
    print("{:.3f}% {} ({} sentences)".format(pcent, gender, nsents))
```

5) В конце зададим строку, содержащую наш фрагмент текста, и вызовем основную функцию `parse_gender` для вывода результатов анализа:

```
text = '''With apologies to James Brown, the hardest working people
in show business may well be ballet dancers.
And at New York City Ballet, none work harder than the dancers in it
s lowest rank, the corps de ballet.
During the first week of the company’s winter season, Claire Kretzsc
hmar, 24, a rising corps member, danced in all seven performances,
appearing in five ballets, sometimes changing costumes at intermissi
on to dance two roles in a night.
```

```
But her work onstage did not even begin to capture the stamina requi
red to be in the corps.
Spending a week shadowing Ms. Kretzschmar was exhausting –
 she gave new meaning to the idea of being on your feet all day.
Twelve-hour days at the David H. Koch Theater, the company's Lincoln
Center home, were hardly unusual:
company class each morning was followed by back-to-back-to-
back rehearsals, with occasional breaks for costume
fittings or physical therapy, and then by the hair-makeup-costume-
dance routine of daily performances.'''

parse_gender(text)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
76.374% unknow (4 sentences)
23.626% female (2 sentences)
```

Таким образом, мы получили следующий результат: среди 6 предложений – 4 предложения не удалось отнести к какому-либо роду, что составило 76% от общего числа предложений; 2 предложения отнеслись к женскому роду, что составило 24% от общего числа предложений.

Функция оценки определяет длину предложений по количеству слов в них. Поэтому, даже при том, что чисто женских предложений меньше, чем неизвестной принадлежности, содержимое статьи относится к женской категории.

==========

Таким образом, мы с вами провели исследование текста на английском языке для определения половой принадлежности предложений в плане отнесения их к мужской и женской категориям.

Задание. Написать простую модель, использующую лингвистические признаки текста (любого объема на ваше усмотрение, но не менее 10 предложений), для выявления преобладающего рода (мужского или женского) во фрагменте различных произведений и статей, представленных ниже:

a) Гарри Поттер и философский камень
b) Приключения Шерлока Холмса
c) Путешествие к центру Земли
d) Елизавета II – царствующая королева Великобритании (перед обработкой нужно удалить надстрочный и подстрочный текст!)
e) Любое другое произведение или статья на ваш выбор

Аналогично разобранному примеру написать модель, использующую лингвистические признаки текста. Для этого:

1. Определите по тексту наборы слов, которые будут использованы для распознавания предложений. Для этого из выбранного вами текста создайте 2

множества с именами MALE_WORDS и FEMALE_WORDS, содержащих ключевые слова, относящиеся к мужским и женским родам соответственно. Например: он, она, парень, девушка и т.д.

2. Создайте функцию `genderize`, которая подсчитывает количество слов в предложении, попадающих в списки MALE_WORDS и FEMALE_WORDS. Если предложение содержит только слова из MALE_WORDS, оно классифицируется как мужское. Предложение, содержащее только слова из FEMALE_WORDS, классифицируется как женское. Если предложение содержит мужские и женские слова, отнесите его к категории двуполых; а если в нем нет ни мужских, ни женских слов, определите его как имеющее неизвестный род. **Функция возвращает русские наименования категорий!**

3. Напишите функцию, которая будет подсчитывать частоту слов, признаков рода и предложений во всем тексте статьи.

4. Используя библиотеку NLTK, разбейте абзацы на предложения. Выделив отдельные предложения, разбейте их на лексемы, чтобы выявить отдельные слова и знаки пунктуации, и передайте размеченный текст функциям классификации для вывода процентов предложений и слов, относящихся к категории мужских, женских, двуполых и неизвестной принадлежности.