**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет: Гуманитарный

Кафедра: Прикладная лингвистика

Дисциплина: Основы программирования

**КУРСОВАЯ РАБОТА**

**Тема**: «Автоматический анализ текстов на естественном языке средствами Python (на примере произведений Джека Лондона)»

**Выполнил**: Шарифзода Ф. О

**Научный руководитель:**

старший преподаватель,Куликова А. А

Ульяновск 2024 г.

**Введение**

Цель работы - изучить и применить методы анализа текстов на естественном языке с использованием языка программирования Python для обработки и анализа произведений Джека Лондона.

Джек Лондон (1876-1916) был американским писателем, известным своими приключенческими рассказами и романами. Родился в Сан-Франциско, Лондон начал свою трудовую жизнь еще в раннем возрасте и испытал множество профессий, включая работу на фабрике, матроса, устричного пирата и золотоискателя на Аляске.

Его первый литературный успех пришел с публикацией рассказа "Тайфун у берегов Японии" в 1893 году. С тех пор Лондон продолжал писать и опубликовал множество произведений, включая романы "Зов предков", "Белый клык", "Мартин Иден" и "Люди бездны". Он путешествовал по разным странам и использовал свой опыт и приключения в своих произведениях.

Лондон был известен своим ярким и реалистичным описанием природы и жизни в экстремальных условиях. Он вдохновлялся идеями социализма и активно выступал за социальную справедливость и равенство.

Джек Лондон умер в 1916 году в возрасте 40 лет. Его литературное наследие продолжает вдохновлять и волновать читателей по всему миру.

**"Love of Life"** (Любовь к жизни) - короткий рассказ Джека Лондона, опубликованный в 1907 году. История рассказывает о выживании одного человека в суровых условиях Арктики.

Весь рассказ сосредоточен на борьбе героя с природой и его внутренней силе воли. Он преодолевает физические и эмоциональные испытания, чтобы найти пищу и укрытие. В ходе своего путешествия он также сталкивается с дикими животными и другими опасностями Арктики.

"Love of Life" иллюстрирует силу человеческого выживания и способность человека преодолевать экстремальные условия. Рассказ подчеркивает важность надежды, оптимизма и любви к жизни в суровых обстоятельствах.

**"The Call of the Wild"** (Зов предков) - роман Джека Лондона, опубликованный в 1903 году. Произведение рассказывает историю пса по имени Бак, который переживает серию приключений во время золотой лихорадки.

Роман "The Call of the Wild" исследует темы природы, выживания, адаптации и примитивных инстинктов. Он показывает превращение Бака от домашнего питомца в дикого существа, восстанавливая связь с его предками - волками. В процессе Бак находит свое истинное предназначение в мире дикой природы.

Роман Лондона прославился своими яркими описаниями природы Аляски и своими глубокими размышлениями о природе человека и животных. "The Call of the Wild" стал одним из самых известных произведений Джека Лондона и продолжает вдохновлять читателей своей сильной и эмоциональной историей о выживании и свободе.

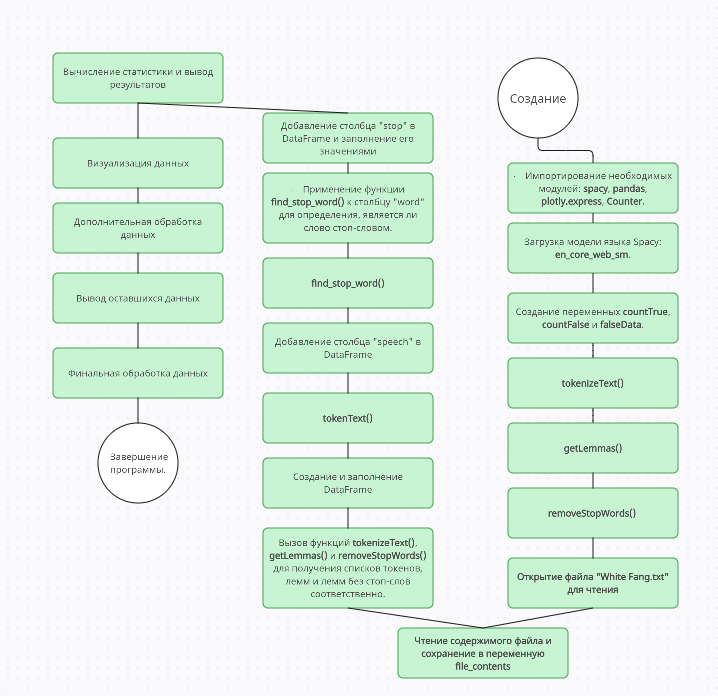
**"White Fang"** (Белый клык) - роман Джека Лондона, опубликованный в 1906 году. Произведение рассказывает историю жизни волка по имени Белый клык, который переживает серию трудностей и переходов от жестокого мира дикой природы к миру людей.

Произведение исследует темы выживания, природы и взаимодействия между животными и людьми. Роман подчеркивает значимость окружения и воспитания в формировании характера и поведения животного. Он также вызывает размышления о влиянии окружающей среды и взаимодействия с людьми на развитие животного инстинкта.

Роман является одним из наиболее известных произведений Лондона и пользуется популярностью своими глубокими образами, напряженным сюжетом и проникновенной тематикой, касающейся отношений между людьми и животными в условиях суровой природы.

**Основная часть**

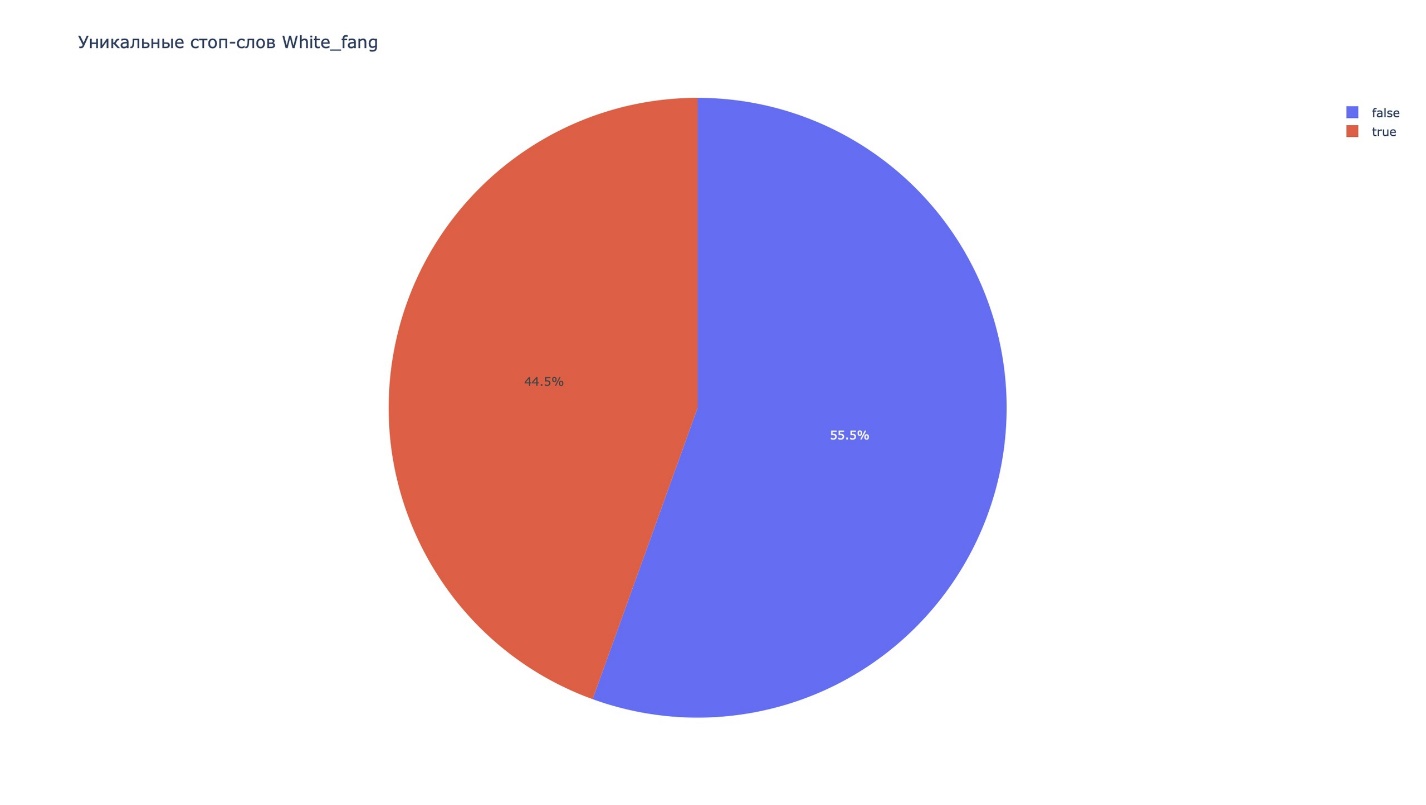
**Алгоритм расчета в формате блок-схемы и сопроводительного описания**



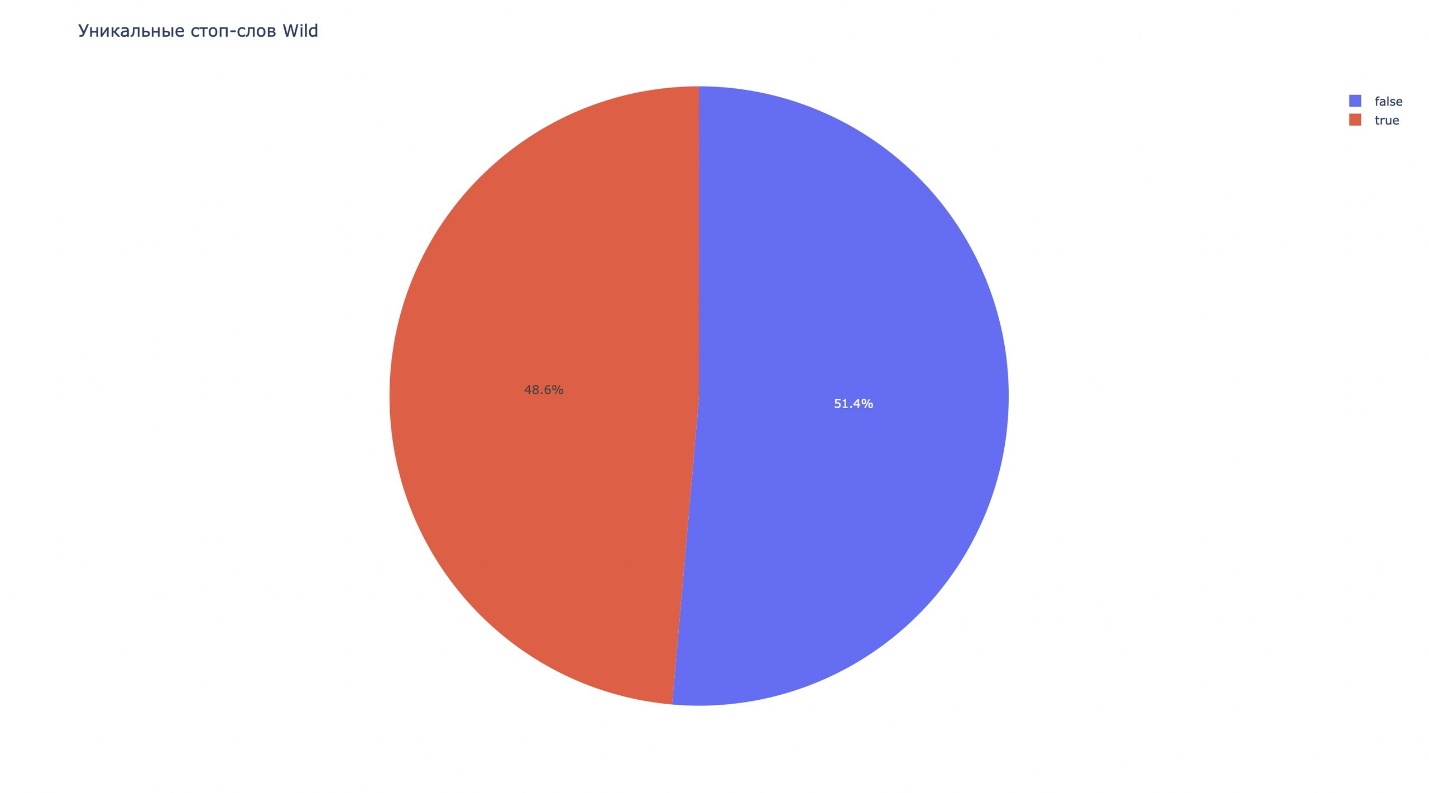
1. Создание:
   * Импортирование необходимых модулей: **spacy**, **pandas**, **plotly.express**, **Counter**.
   * Загрузка модели языка Spacy: **en\_core\_web\_sm**.
   * Инициализация переменных **countTrue**, **countFalse** и **falseData**.
2. Функция **tokenizeText()**:
   * Чтение текста из файла.
   * Использование Spacy для токенизации текста.
   * Создание списка токенов и возврат его.
3. Функция **getLemmas()**:
   * Чтение текста из файла.
   * Использование Spacy для получения лемм (нормализованных форм) слов.
   * Создание списка лемм и возврат его.
4. Функция **removeStopWords()**:
   * Чтение текста из файла.
   * Использование Spacy для удаления стоп-слов (часто встречающихся, но не несущих смысловой нагрузки).
   * Создание списка лемм без стоп-слов и возврат его.
5. Чтение текста из файла и вызов функций:
   * Открытие файла "White Fang.txt" для чтения.
   * Чтение содержимого файла и сохранение в переменную **file\_contents**.
   * Вызов функций **tokenizeText()**, **getLemmas()** и **removeStopWords()** для получения списков токенов, лемм и лемм без стоп-слов соответственно.
6. Создание и заполнение DataFrame:
   * Создание пустого DataFrame **data**.
   * Добавление столбцов "word" и "lemma" в DataFrame и заполнение их значениями из списков токенов и лемм соответственно.
7. Функция **tokenText()**:
   * Использование Spacy для определения части речи каждого слова.
   * Возврат первой части речи слова.
8. Добавление столбца "speech" в DataFrame:
   * Применение функции **tokenText()** к столбцу "word" для определения части речи каждого слова.
   * Добавление столбца "speech" в DataFrame и заполнение его значениями.
9. Функция **find\_stop\_word()**:
   * Использование Spacy для определения, является ли слово стоп-словом.
   * Возврат булевого значения (True или False).
10. Добавление столбца "stop" в DataFrame:
    * Применение функции **find\_stop\_word()** к столбцу "word" для определения, является ли слово стоп-словом.
    * Добавление столбца "stop" в DataFrame и заполнение его значениями.
11. Вычисление статистики и вывод результатов:
    * Подсчет количества True и False в столбце "stop".
    * Подсчет процента True и False относительно общего количества слов.
    * Подсчет частоты встречаемости стоп-слов и сохранение результатов в переменную **falseData**.
    * Вывод результатов подсчета и частоты встречаемости стоп-слов.
12. Визуализация данных:
    * Создание круговой диаграммы с использованием библиотеки Plotly Express для визуализации частоты встречаемости стоп-слов.

**Круговые диаграммы с использованием библиотеки Plotly Express**

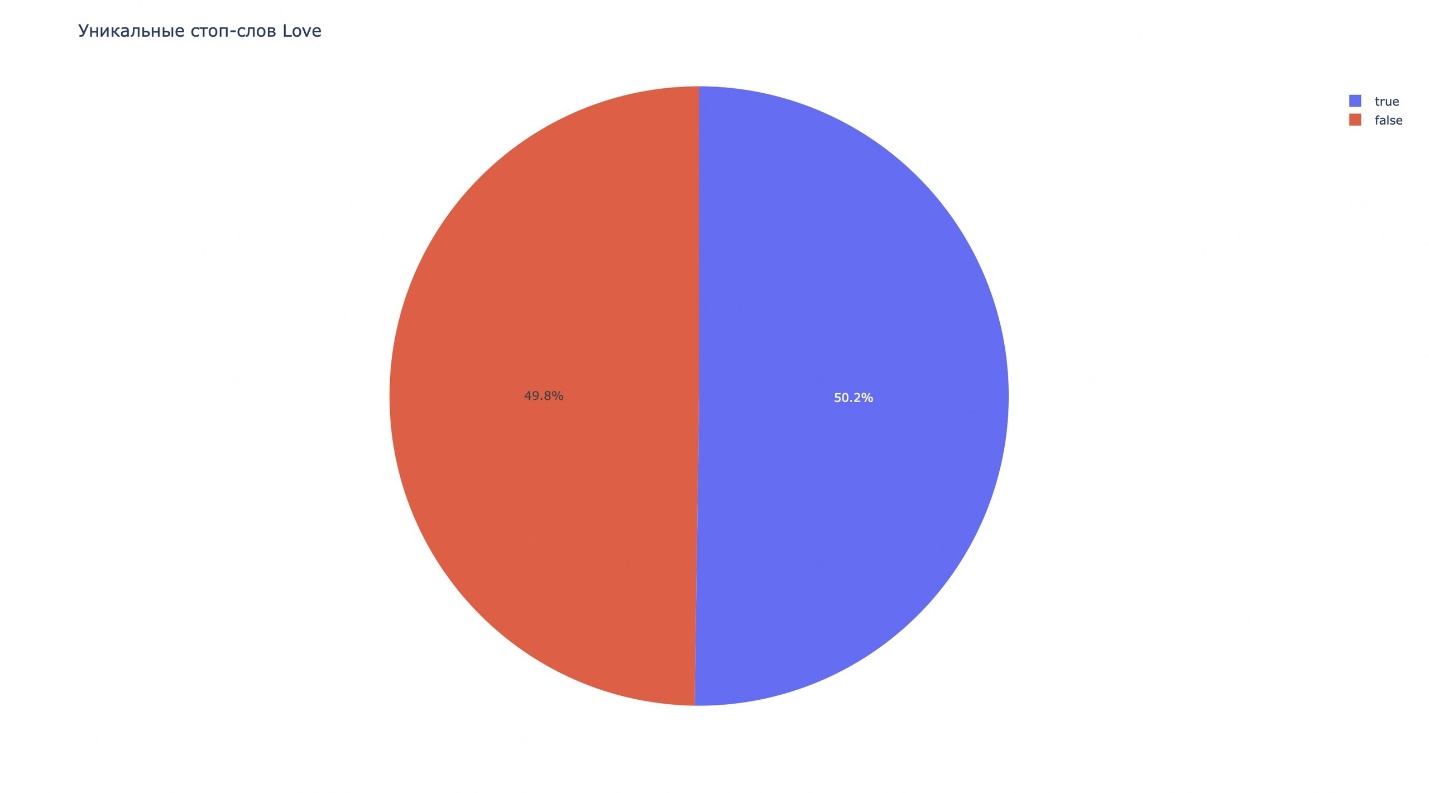
**"White Fang"**



**"The Call of the Wild"**



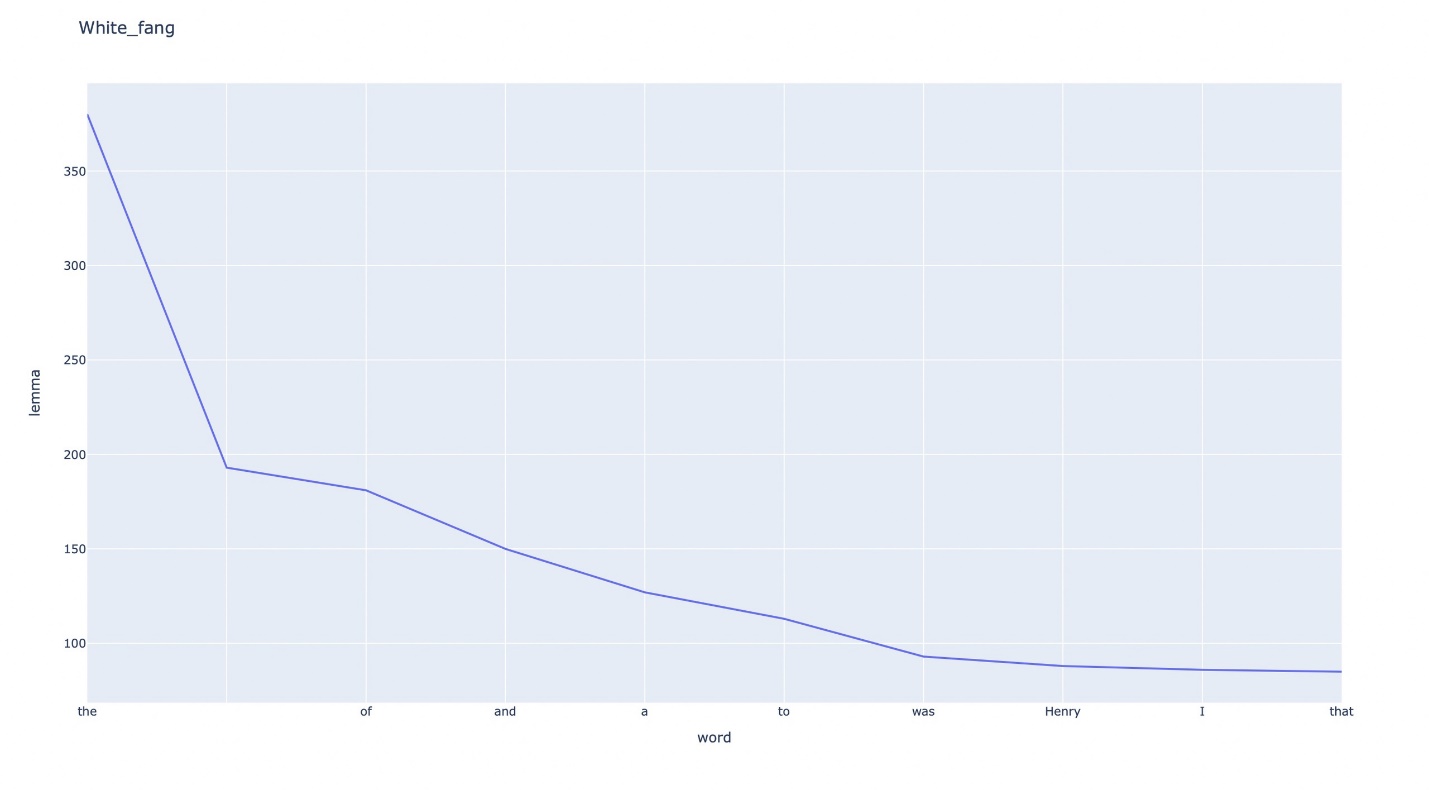
**"Love of Life"**

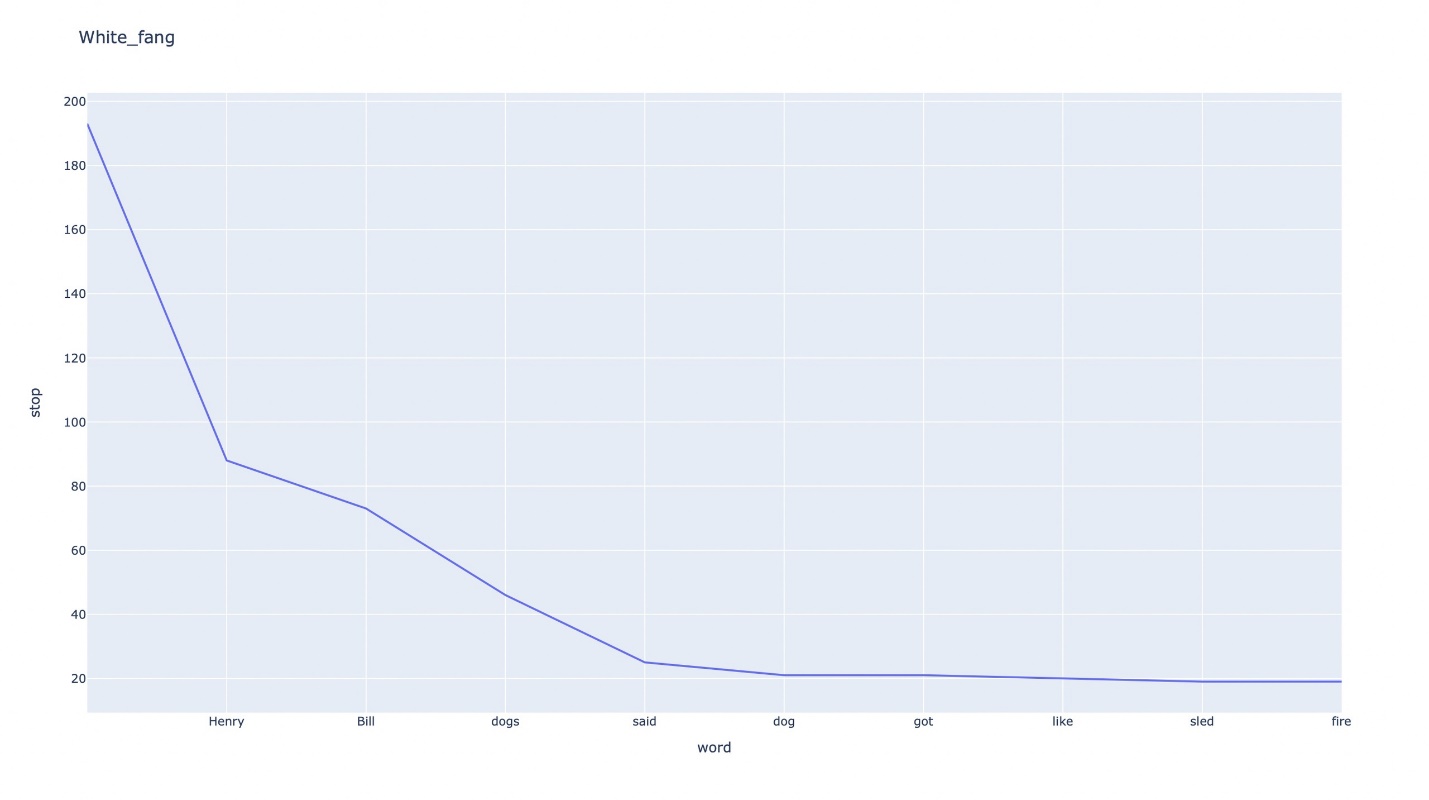


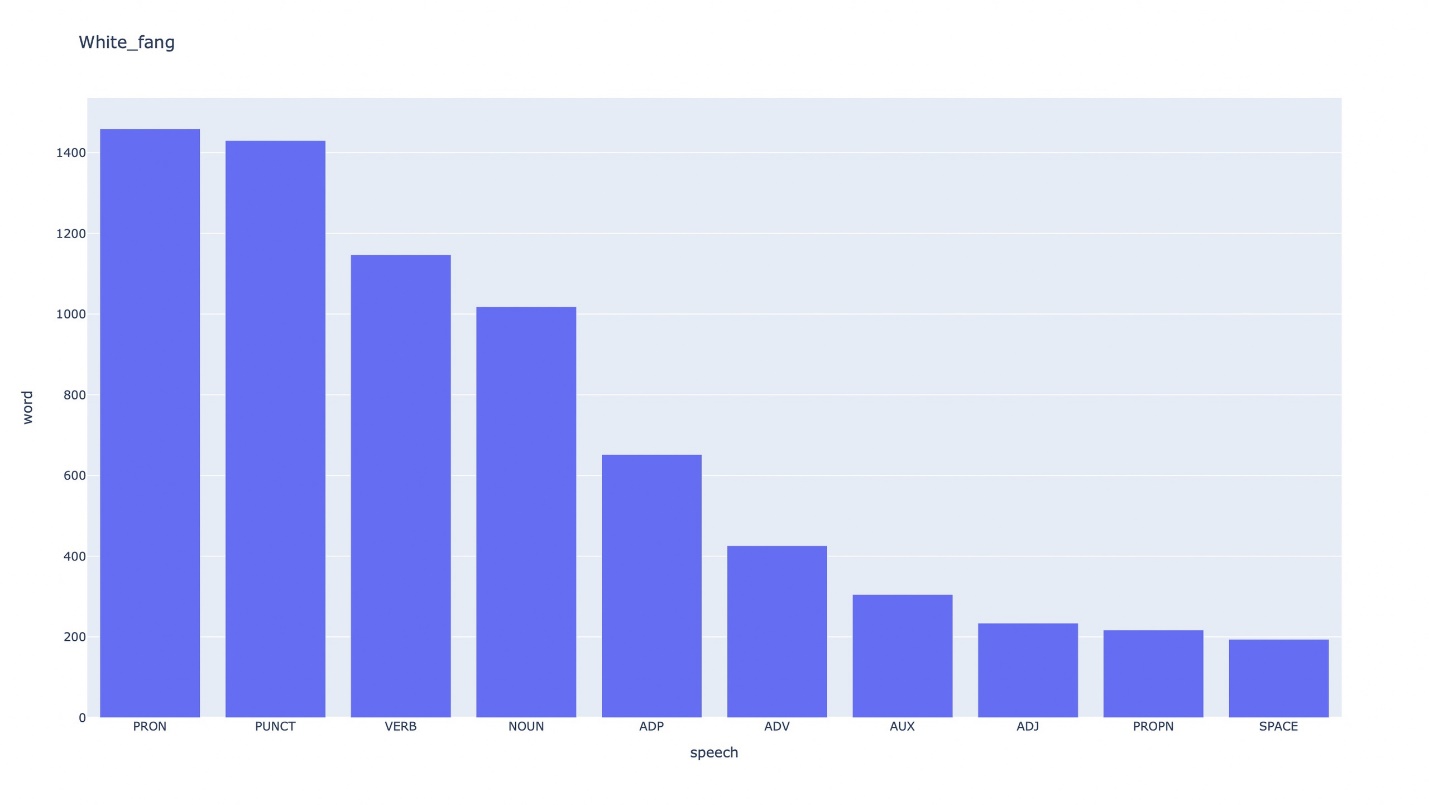
* + Создание графика линии и столбчатой диаграммы для визуализации других статистических данных.

**Графики линии и столбчатые диаграммы для визуализации других статистических данных**

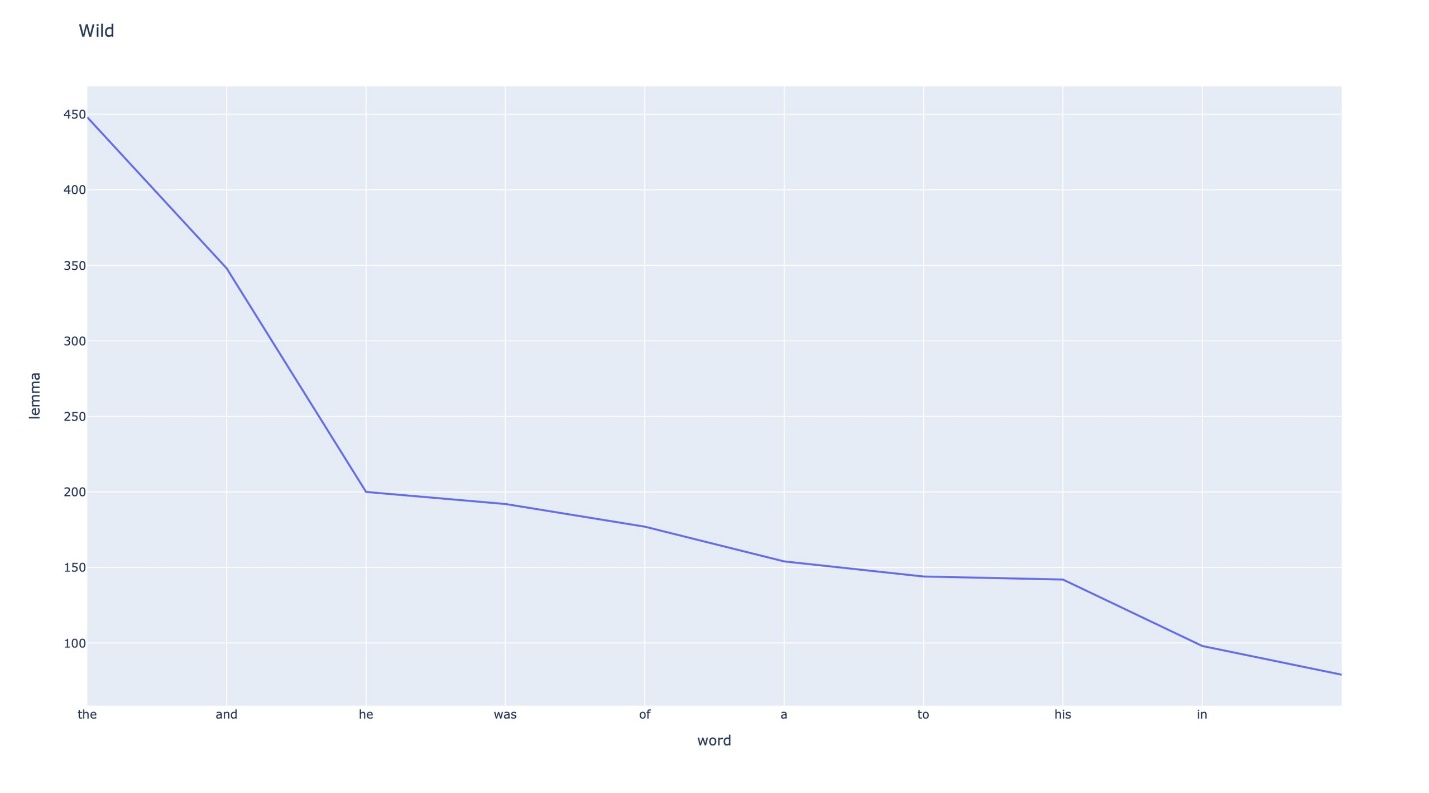
**"White Fang"**

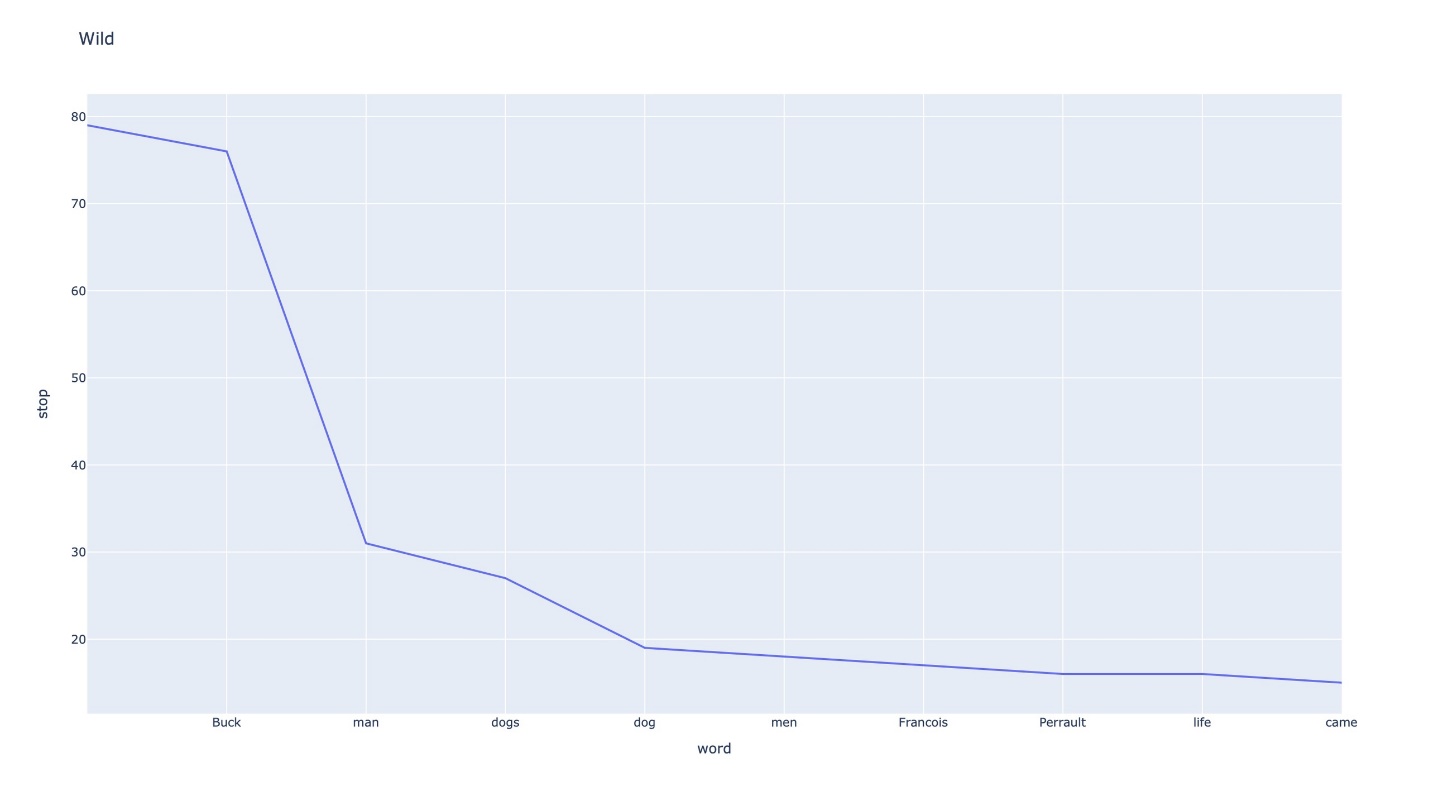


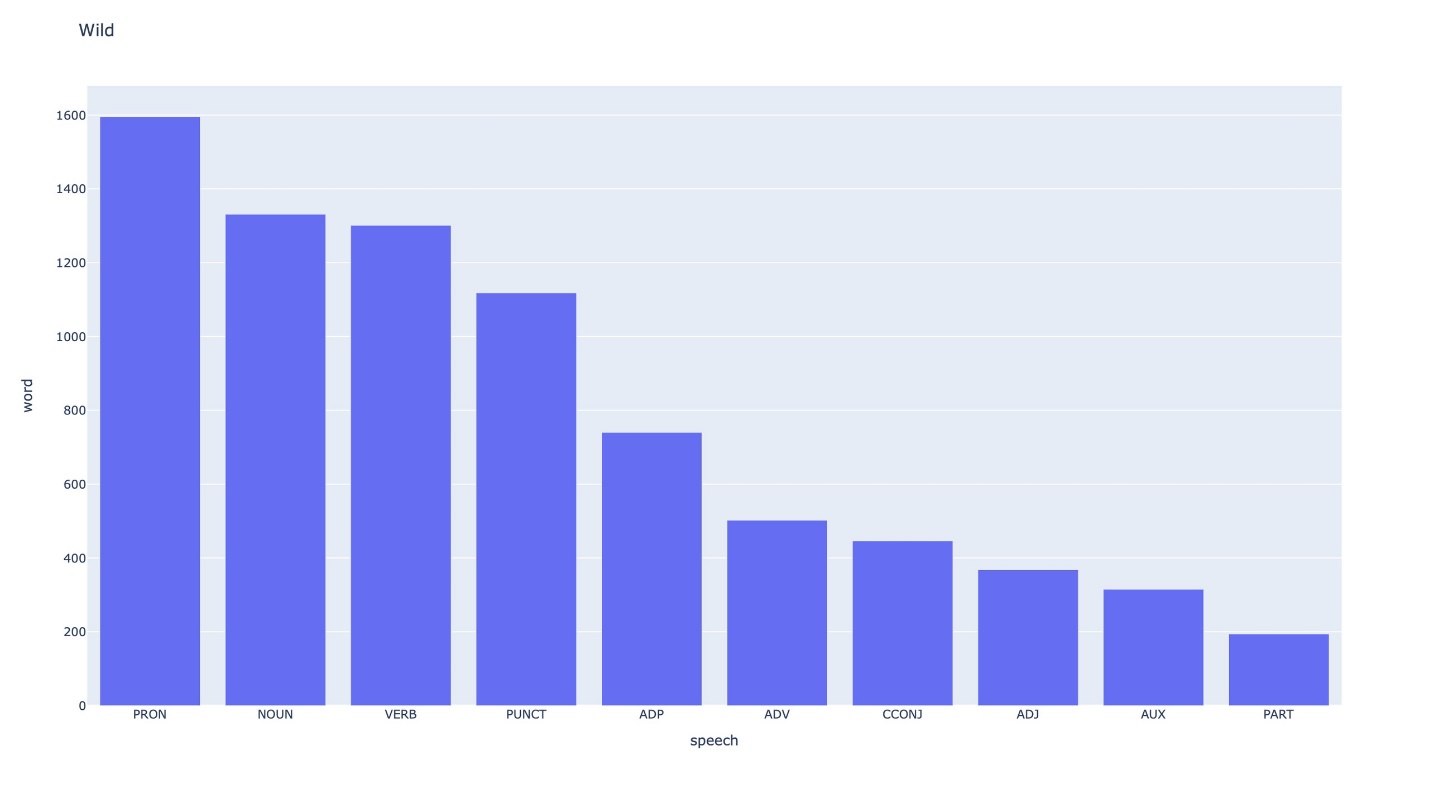




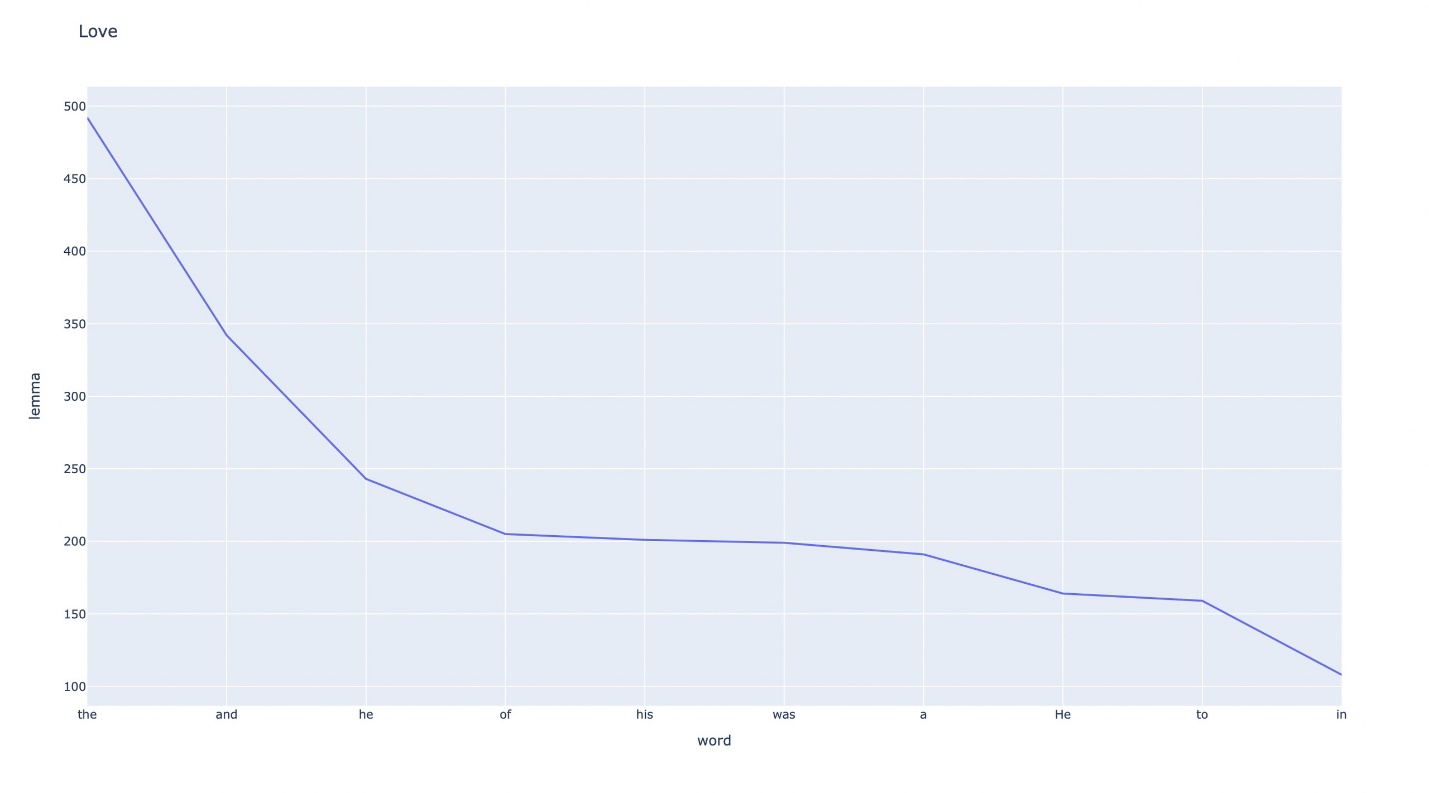
**"The Call of the Wild"**

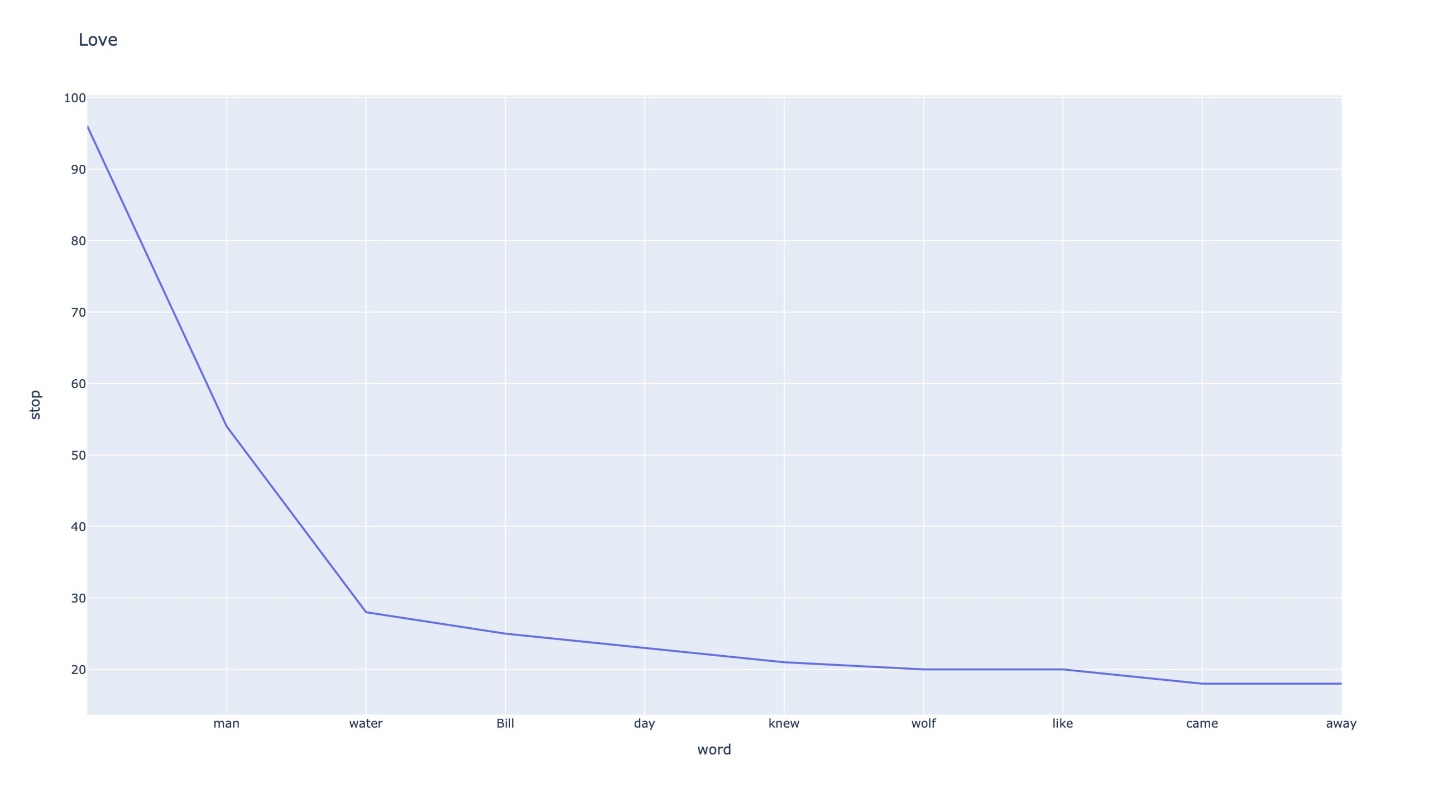


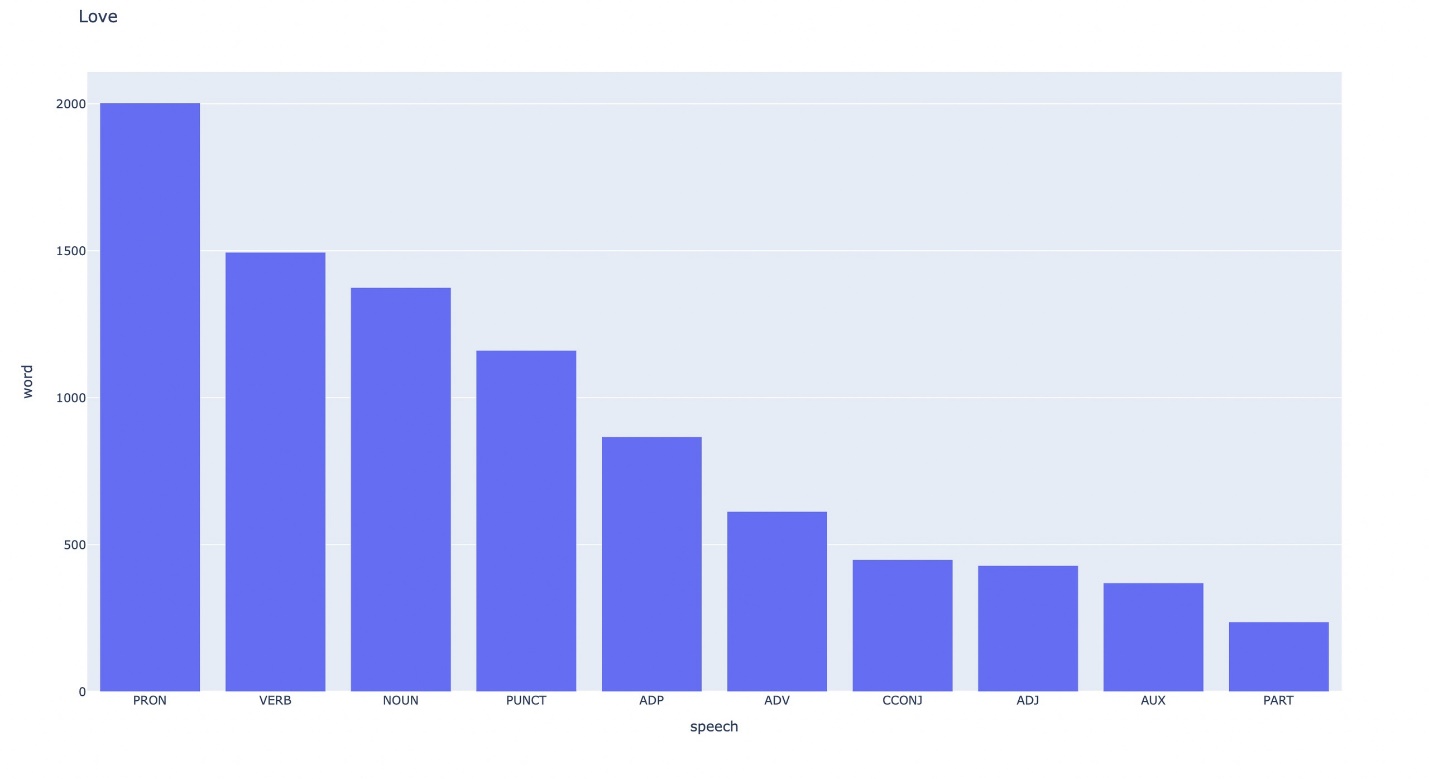




**"Love of Life"**







1. Дополнительная обработка данных:
   * Создание нового DataFrame **data\_without\_punctuation**, исключающего знаки пунктуации из столбца "speech".
   * Группировка данных по словам и подсчет частоты встречаемости слов.
   * Визуализация полученных результатов с использованием графика линии и столбчатой диаграммы.
2. Вывод оставшихся данных:
   * Группировка данных по частям речи и подсчет частоты встречаемости каждой части речи.
   * Визуализация полученных результатов с использованием столбчатой диаграммы.
3. Финальная обработка данных:
   * Создание нового DataFrame **falseData**, содержащего только неправильные (не стоп-слова) данные.
   * Группировка данных по словам и подсчет частоты встречаемости неправильных слов.
   * Визуализация полученных результатов с использованием графика линии.
4. Завершение программы.

**Результаты анализа – для каждой метрики**

1. Круговая диаграмма "Уникальные стоп-слова":
   * Позволяет увидеть долю стоп-слов от общего количества слов в тексте.
   * Можно сделать вывод о том, насколько текст содержит стоп-слова, которые могут не нести смысловую нагрузку.
2. График линии "Частота встречаемости слов":
   * Отображает топ-10 наиболее часто встречающихся слов в тексте.
   * Позволяет определить наиболее часто используемые слова и их частоту встречаемости.
   * Можно сделать вывод о наиболее значимых или повторяющихся словах в тексте.
3. Столбчатая диаграмма "Частота частей речи":
   * Показывает распределение слов по частям речи (существительные, глаголы, прилагательные и т.д.).
   * Позволяет определить наиболее частую часть речи в тексте и ее частоту встречаемости.
   * Можно сделать вывод о структуре и характере текста на основе преобладающих частей речи.
4. График линии "Частота неправильных слов":
   * Отображает топ-10 наиболее часто встречающихся неправильных (не стоп-слов) слов в тексте.
   * Позволяет определить наиболее значимые или повторяющиеся неправильные слова в тексте.
   * Можно сделать вывод о наличии особенных тематических слов или ключевых понятий в тексте.

Эти выводы помогут лучше понять структуру и содержание текста, выделить наиболее важные слова и части речи, а также оценить присутствие стоп-слов и их влияние на текст.

**Заключение: основные выводы по работе.**

В ходе работы были получены следующие основные выводы:

Стоп-слова: Анализ текстов позволил выделить наличие стоп-слов, которые представляют собой часто встречающиеся, но не несущие смысловой нагрузки слова. Их процентное соотношение к общему количеству слов в трех текстах составило 52.37% в среднем. Это может указывать на особенности стиля или жанра произведений Джека Лондона.

Лемматизация: применение алгоритма лемматизации позволило получить нормализованные формы слов. Это упрощает анализ и сравнение словных форм, помогает выделить семантические связи и определить ключевые понятия в тексте.

Части речи: анализ частей речи позволил определить распределение слов по различным категориям (существительные, глаголы, прилагательные и т.д.). Наиболее часто встречающейся частью речи в произведениях Джека Лондона являются местоимения. Это может свидетельствовать о специфике его стиля или описывать основные элементы его произведений.

Неправильные слова: выделены наиболее часто встречающиеся неправильные слова, их частота встречаемости и распределение. Чаще всего встречаются слова обозначающие главного героя или же “man” Это может указывать на особенности лексического состава произведений или выявлять наиболее употребляемые понятия.

В целом, анализ текстов на естественном языке с использованием средств Python позволяет получить информацию о структуре и содержании произведений Джека Лондона, исследовать лексические и семантические аспекты его текстов. Это может быть полезным для лингвистических исследований, анализа литературных произведений или извлечения информации из больших текстовых корпусов.

**Приложение: исходный текст программы**

**import** spacy  
**import** pandas **as** pd  
**import** plotly.express **as** px  
**from** collections **import** Counter  
nlp = spacy.load(**"en\_core\_web\_sm"**)  
  
countTrue = 0  
countFalse = 0  
falseData = []  
  
**def** tokenizeText():  
 doc = nlp(file\_contents)  
 token\_list = []  
 **for** token **in** doc:  
 token\_list.append(token.text)  
  
 **return** token\_list  
  
**def** getLemmas():  
 doc = nlp(file\_contents)  
 token\_list = []  
 **for** token **in** doc:  
 token\_list.append(token.lemma\_)  
  
 **return** token\_list  
  
**def** removeStopWords():  
 doc = nlp(file\_contents)  
 token\_list = []  
 **for** token **in** doc:  
 **if** token.is\_stop == **False**:  
 token\_list.append(token.lemma\_)  
  
 **return** token\_list  
  
text = open(**'White Fang.txt'**, **'r'**)  
file\_contents = text.read()  
  
resultText = tokenizeText()  
*#print(resultText)*resultLemma = getLemmas()  
*#print(resultLemma)*resultToken = removeStopWords()  
*#print(resultToken)*data = pd.DataFrame()  
data[**'word'**] = resultText  
  
data[**'lemma'**] = resultLemma  
print(data)  
  
**def** tokenText(one\_text):  
 doc = nlp(one\_text)  
  
 **return** doc[0].pos\_  
  
data[**'speech'**] = data[**'word'**].apply(tokenText)  
print(data)  
  
**def** find\_stop\_word(word):  
 doc = nlp(word)  
  
 **return** doc[0].is\_stop  
  
data[**'stop'**] = data[**'word'**].apply(find\_stop\_word)  
print(data)  
  
**for** i **in** data[**'stop'**]:  
 **if** i == **True**:  
 countTrue += 1  
 **else**:  
 countFalse += 1  
  
falseData = data[data[**'stop'**] == **True**]  
print(countTrue/len(data) \* 100)  
print(countFalse/len(data) \* 100)  
falseData = Counter(falseData)  
print(falseData)  
  
fig = px.pie(data, values=falseData.get(**"countTrue"**), names=**'stop'**, title=**'Уникальные стоп-слов White\_fang'**)  
fig.show()  
  
*#data.to\_excel('data\_frame.xlsx')*data\_without\_punctuation = data[data[**'speech'**] != **'PUNCT'**]  
print(data\_without\_punctuation)  
  
pos\_data = data\_without\_punctuation.groupby(**'word'**).count().sort\_values(**'lemma'**, ascending=**False**)  
print(pos\_data.head(10))  
fig = px.line(pos\_data.head(10), x=pos\_data.get(**"word"**), y=**"lemma"**, title=**'White\_fang'**)  
fig.show()  
  
new\_data = data.groupby(**'speech'**).count().sort\_values(**'lemma'**, ascending=**False**)  
print(new\_data.head(10))  
fig = px.bar(new\_data.head(10), x=new\_data.get(**"speech"**), y=**'word'**, title=**'White\_fang'**)  
fig.show()  
  
falseData = data\_without\_punctuation.loc[data[**'stop'**] == **False**]  
falseData = falseData.groupby(**'word'**).count().sort\_values(**'stop'**, ascending=**False**)  
print(falseData.head(10))  
fig = px.line(falseData.head(10), x=falseData.get(**"word"**), y=**'stop'**, title=**'White\_fang'**)  
fig.show()  
  
trueData = data.loc[data[**'stop'**] == **True**]  
trueData = trueData.groupby(**'word'**).count().sort\_values(**'stop'**, ascending=**False**)