Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 7 по дисциплине «Основы искусственного интеллекта»

Тема Кластеризация

Студент Сапожков А.М.,

Группа ИУ7-13М

Преподаватель Строганов Ю.В.

Москва, 2024

Содержание

ВВЕДЕНИЕ

Кластеризация является одной из важнейших задач в области анализа данных. Она используется для группировки объектов в такие подмножества (кластеры), в которых объекты внутри каждого кластера максимально схожи, а объекты из разных кластеров максимально различны. Задача кластеризации широко применяется в различных областях, включая обработку текстов, анализ изображений, биоинформатику и многие другие.

Целью данной лабораторной работы является сравнение алгоритмов кластеризации векторов методами K-средних и C-средних.

Задачи данной лабораторной работы:

1) провести кластеризацию заданный набор данных;
2) привести результаты работы при различном указываемом количестве кластеров;
3) определить среднее внутрикластерное расстояние и среднее межкластерное расстояние для каждого рассматриваемого случая.

1 Аналитическая часть

1.1 Кластеризация методом K-средних

Алгоритм K-средних является одним из самых популярных методов кластеризации. Он основан на итеративном процессе, в ходе которого определяется k центроидов кластеров. В каждой итерации алгоритм присваивает каждому объекту (в нашем случае, каждому документу) метку, соответствующую ближайшему центроиду. Затем обновляются координаты центроидов, которые рассчитываются как среднее значение всех объектов, принадлежащих каждому кластеру. Процесс продолжается до тех пор, пока центроиды не перестанут изменяться, либо не будет достигнут лимит по числу итераций. Преимущества метода K-средних:

— подходит для случаев, когда кластеры имеют форму, приближенную к круглой.

Недостатки:

— требуется заранее задавать количество кластеров k, что не всегда возможно без предварительного анализа данных;

— алгоритм чувствителен к выбору начальных центроидов, что может привести к искажению результатов кластеризации;

— не работает хорошо в случае сложных или эллиптических форм кластеров.

1.2 Кластеризация методом C-средних

Алгоритм Fuzzy C-Means (FCM) — один из классических методов кластеризации. В отличие от K-Means, объекты данных в FCM могут принадлежать нескольким кластерам одновременно, причем каждому кластеру сопоставляется степень принадлежности объекта (значение из интервала [0, 1]). Этот метод широко применяется, когда данные имеют нечёткие границы или не подходят для традиционной жёсткой кластеризации.

FCM нацелен на минимизацию расстояния между данными (объектами) и центроидами кластеров, при этом каждому объекту фиксируется нечеткая степень принадлежности к каждому кластеру. В результате вместо жёсткого разделения на кластеры (как, например, в K-Means) объект может принадлежать сразу нескольким кластерам с различными коэффициентами степени принадлежности. Преимущества метода FCM:

— размытие границ кластеров: алгоритм может быть полезен для данных, у которых границы между группами нечёткие;

— наличие возможности регулирования степени размытости (нечёткости) кластеров.

Недостатки:

— необходимость подбора большего количества параметров, чем для алгоритма K-средних;

— уязвимость к выбросам: малые изменения входных данных могут повлиять на местоположение центроидов кластеров;

— производительность как правило ниже, чем у алгоритма K-средних, по причине пе-

ерасчёта степеней принадлежности кластерам на каждой итерации.

2 Конструкторская часть

2.1 Векторизация текстов методом TF-IDF

TF-IDF представляет собой статистический показатель, который используется для оценки важности слова в контексте документа и всего корпуса. Он состоит из следующих составляющих.

— **TF** (Term Frequency) — частота термина в документе. Это количество раз, которое слово появляется в данном документе.

— **IDF** (Inverse Document Frequency) — обратная частота документа. Этот показатель уменьшает вес часто встречающихся слов в корпусе, таких как стоп-слова (например, «и» на «с»).

Итоговый показатель TF-IDF вычисляется как произведение этих двух величин:

$$TF - IDF(t, d) = TF(t, d) \times IDF(t), \tag{2.1}$$

где t — это слово, d — документ, а IDF определяется по формуле

$$IDF(t) = log\frac{N}{df(t)}, \tag{2.2}$$

где N — общее количество документов в корпусе, а $df(t)$ — количество документов, содержащих слово t. Метод TF-IDF позволяет выделить важные слова и фразы, которые будут использоваться в качестве признаков для кластеризации.

2.2 Метрики оценки качества кластеризации

Для оценки качества кластеризации предлагается использовать следующие метрики.

— **Среднее внутрикластерное расстояние** — мера того, насколько объекты внутри одного кластера похожи друг на друга. Чем меньше это расстояние, тем более компактными являются кластеры.

— **Среднее межкластерное расстояние** — расстояние между центроидами различных кластеров. Чем больше это расстояние, тем лучше разделены кластеры.

Данные метрики позволяют оценить, насколько качественно проведена кластеризация, и дают возможность сравнивать различные алгоритмы кластеризации.

2.3 Визуализация кластеров

Для наглядной оценки результатов кластеризации используется метод главных компонент (PCA). PCA позволяет снизить размерность данных до двух или трёх компонентов, что позволяет представить многомерные данные в двумерном или трёхмерном пространстве. С помощью этого метода можно визуализировать кластеры и оценить их разделимость. Визуали-

зация помогает лучше понять, как алгоритм классифицирует объекты и насколько хорошо он справляется с поставленной задачей.

3 Технологическая часть

3.1 Средства реализации

В качестве языка программирования для реализации выбранных алгоритмов был выбран язык программирования Python [1] ввиду наличия библиотек для кластеризации (sklearn [2]), подсчёта метрик её качества, а также для лемматизации текстов (pymorphy3 [3]).

3.2 Реализация алгоритма

На листинге 3.1 представлена реализация программы для сравнения алгоритмов кластеризации текстов методами K-средних и C-средних.

Листинг 3.1 — Сравнение алгоритмов кластеризации K-Means и K-Medoids

```
import os
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
import pymorphy3
from nltk.corpus import stopwords
import nltk
import skfuzzy as fuzz

# Make sure stopwords are downloaded
nltk.download('stopwords')

def read_file(file_path):
  try:
    if file_path.endswith('.docx'):
      from docx import Document
      doc = Document(file_path)
      return ' '.join([para.text for para in doc.paragraphs])
    elif file_path.endswith('.odt'):
      from odf.opendocument import load
      from odf.text import P
      doc = load(file_path)
      paragraphs = doc.getElementsByType(P)
```

```
      return " ".join([str(p) for p in paragraphs if p and p.firstChild
         ])
    elif file_path.endswith('.txt'):
      with open(file_path, 'r', encoding='latin-1') as file:
        return file.read()
    elif file_path.endswith('.html'):
      from bs4 import BeautifulSoup
      with open(file_path, 'r', encoding='latin-1') as file:
        soup = BeautifulSoup(file, 'html.parser')
        return soup.get_text()
    else:
      return ''
  except Exception as e:
    print(f"Error reading file {file_path}: {e}")
  return ''

def load_data(folder_path):
  texts, filenames = [], []
  for root, _, files in os.walk(folder_path):
    for file in files:
      filepath = os.path.join(root, file)
      text = read_file(filepath)
      if text:
        texts.append(text)
        filenames.append(file)
  return texts, filenames

folder_path = "./"  # Specify the folder path
texts, filenames = load_data(folder_path)

# Vectorization
# 1. Vectorization by word forms
vectorizer_forms = TfidfVectorizer(max_features=5000)
X_forms = vectorizer_forms.fit_transform(texts)

# 2. Lemmatization of the text
morph = pymorphy3.MorphAnalyzer()
lemmatized_texts = [' '.join([morph.parse(word)[0].normal_form for word
    in text.split()]) for text in texts]
vectorizer_lemmas = TfidfVectorizer(max_features=5000)
X_lemmas = vectorizer_lemmas.fit_transform(lemmatized_texts)
```

```
def fuzzy_cmeans_clustering(X, n_clusters):
  # Transpose for skfuzzy (skfuzzy expects shape (features, samples))
  X = X.T.toarray() if not isinstance(X, np.ndarray) else X.T

  # Apply Fuzzy C-Means
  cntr, u, _, _, _, _, _ = fuzz.cluster.cmeans(
    X, c=n_clusters, m=1.1, error=0.01, maxiter=10000, init=None
  )

  # Get cluster labels based on maximum membership
  cluster_labels = np.argmax(u, axis=0)  # Shape (samples,)
  return cluster_labels

# Clustering
n_clusters = 7
labels = {}

# 1. K-means for word forms
kmeans_forms = KMeans(n_clusters=n_clusters, random_state=None)
labels["kmeans_forms"] = kmeans_forms.fit_predict(X_forms)

# 2. K-means for lemmas
kmeans_lemmas = KMeans(n_clusters=n_clusters, random_state=None)
labels["kmeans_lemmas"] = kmeans_lemmas.fit_predict(X_lemmas)

# 3. C-means for word forms
labels["cmeans_forms"] = fuzzy_cmeans_clustering(X_forms, n_clusters)

# 4. C-means for lemmas
labels["cmeans_lemmas"] = fuzzy_cmeans_clustering(X_lemmas, n_clusters)

print(labels)

def calculate_distances(X, labels):
  # Convert X to a dense array if it's not already
  X = X.toarray() if not isinstance(X, np.ndarray) else X

  n_clusters = len(set(labels))
  cluster_distances = []
  intra_cluster_distances = []
```

```
  for cluster in range(n_clusters):
    cluster_points = X[labels == cluster]
    if len(cluster_points) > 1:
      # Average intra-cluster distance
      intra_dist = np.mean(pairwise_distances(cluster_points))
      intra_cluster_distances.append(intra_dist)

    # Centroid or median of the current cluster
    cluster_center = np.mean(cluster_points, axis=0)
    cluster_distances.append(cluster_center)

  # Average inter-cluster distance
  inter_cluster_distances = np.mean(pairwise_distances(
     cluster_distances))

  return np.mean(intra_cluster_distances), inter_cluster_distances

# Calculating metrics
metrics = {
  "kmeans_forms": calculate_distances(X_forms, labels["kmeans_forms"]),
  "kmeans_lemmas": calculate_distances(X_lemmas, labels["kmeans_lemmas"
     ]),
  "cmeans_forms": calculate_distances(X_forms, labels["cmeans_forms"]),
  "cmeans_lemmas": calculate_distances(X_lemmas, labels["cmeans_lemmas"
     ]),
}

def print_clusters(labels):
  for cluster in range(len(set(labels))):
    print(f"    {cluster}: {[filenames[i] for i in range(len(filenames)
       ) if labels[i] == cluster]}")

def plot_clusters(X, labels, title):
  from sklearn.decomposition import PCA
  pca = PCA(n_components=2)
  X_pca = pca.fit_transform(X.toarray())
  plt.figure(figsize=(8, 6))
  plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap="viridis", s=50)
  plt.colorbar()
  plt.title(title)
```

```
  plt.xlabel("PCA1")
  plt.ylabel("PCA2")
  plt.show()

# Display results
for method, (intra_dist, inter_dist) in metrics.items():
print(f"Method: {method}")
print(f"  Average intra-cluster distance: {intra_dist}")
print(f"  Average inter-cluster distance: {inter_dist}")
print(f"  Clusters:")
print_clusters(labels[method])
print("\n")

# Visualization for each method
plot_clusters(X_forms, labels["kmeans_forms"], "K-means (word forms)")
plot_clusters(X_lemmas, labels["kmeans_lemmas"], "K-means (lemmas)")
plot_clusters(X_forms, labels["cmeans_forms"], "C-means (word forms)")
plot_clusters(X_lemmas, labels["cmeans_lemmas"], "C-means (lemmas)")
```

4 Исследовательская часть

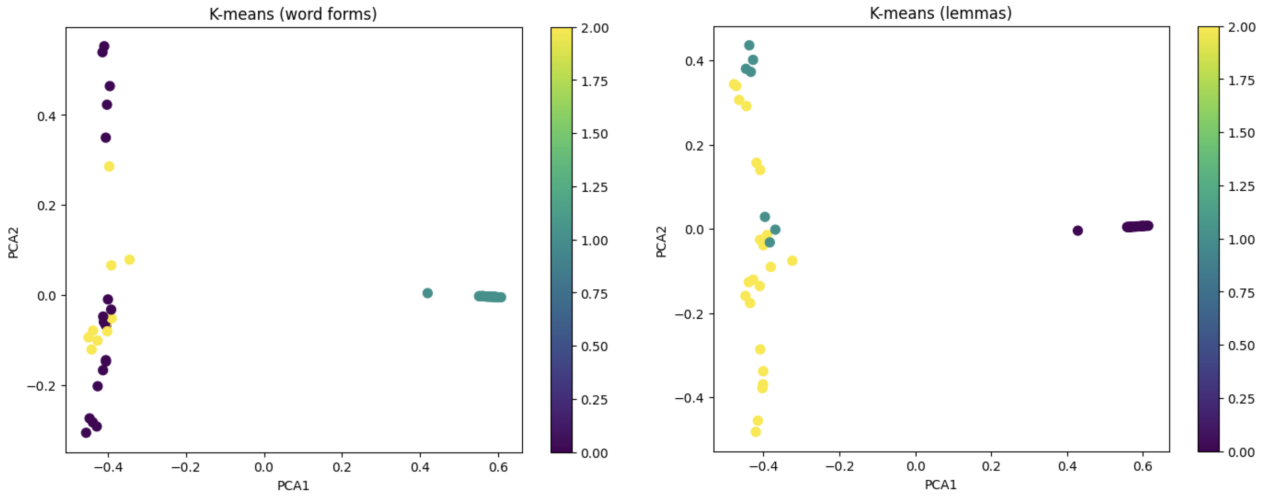
4.1 Среда для тестирования

Для тестирования разработанного алгоритма применялась облачная платформа Google Colab [4], не требующая установки ПО на локальный компьютер.

4.2 Тестирование кластеризации

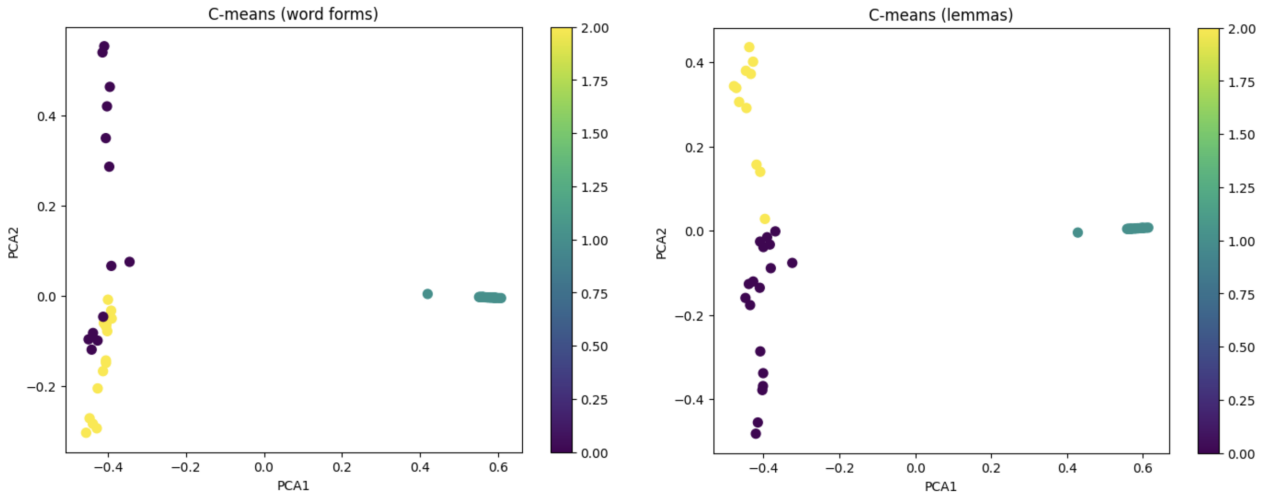
Для кластеризации использовались размеченные тексты, разделение которых было известно заранее. Кластеризуемые тексты имели одну из следующих тем: комментаторы, ставки на спорт, спортивная медицина, органическая химия, неорганическая химия, биохимия крови, анамнез.

На рисунках 4.1-4.6 приведены графики, показывающие результаты кластеризации методами K-средних и методом C-средних для текстовых данных при заданном количестве кластеров (3, 7 и 10). При визуализации кластеров был использован метод главных компонент (PCA) для уменьшения размерности.



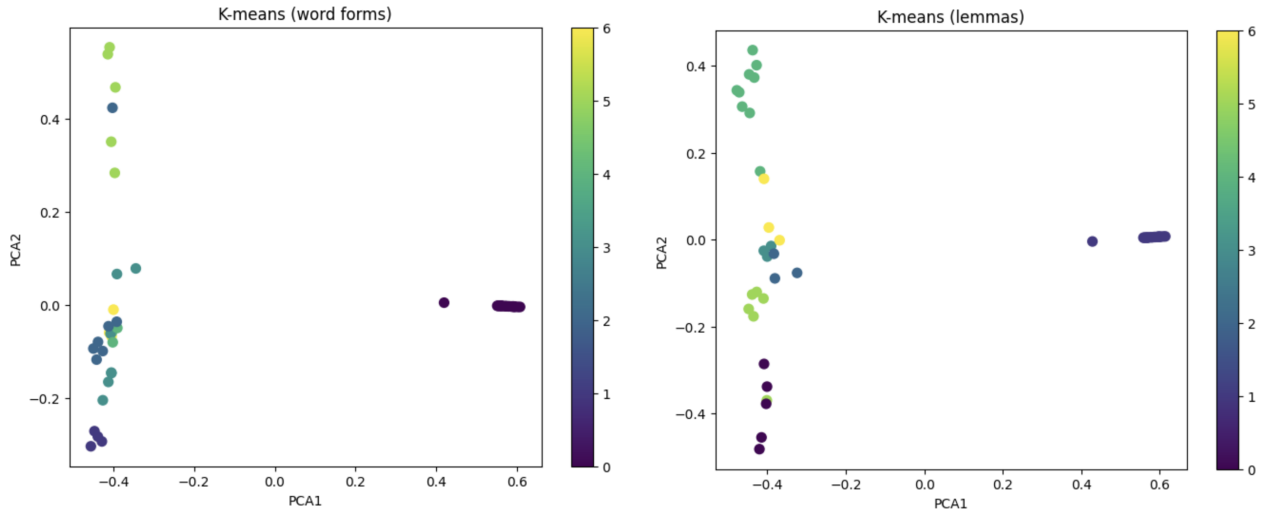
а) словоформы б) леммы

Рисунок 4.1 — Результаты кластеризации методом K-средних, 3 кластера



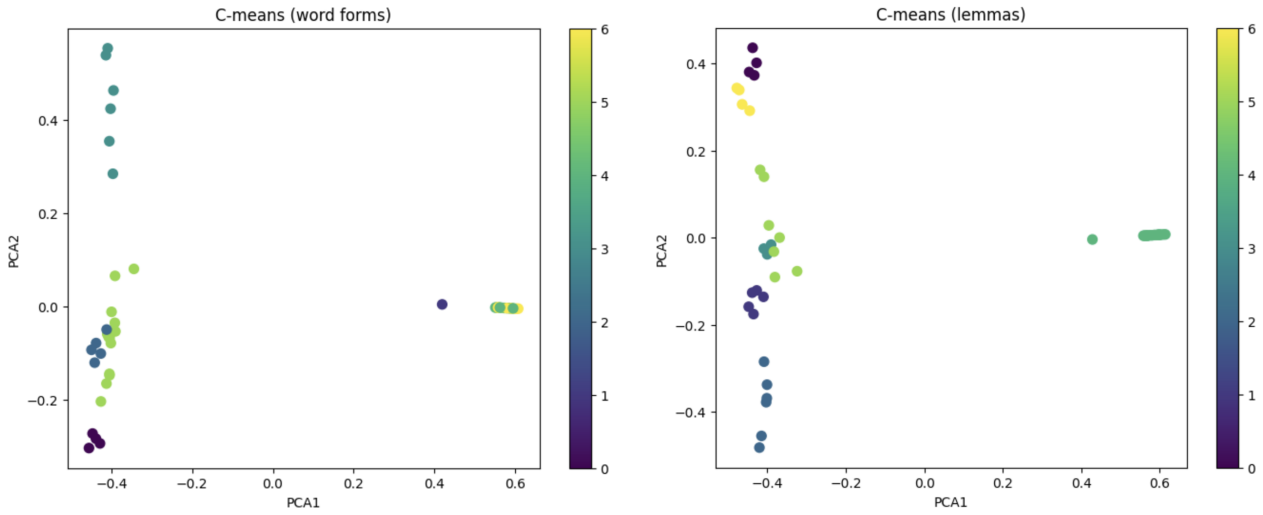
а) словоформы б) леммы

Рисунок 4.2 — Результаты кластеризации методом C-средних, 3 кластера



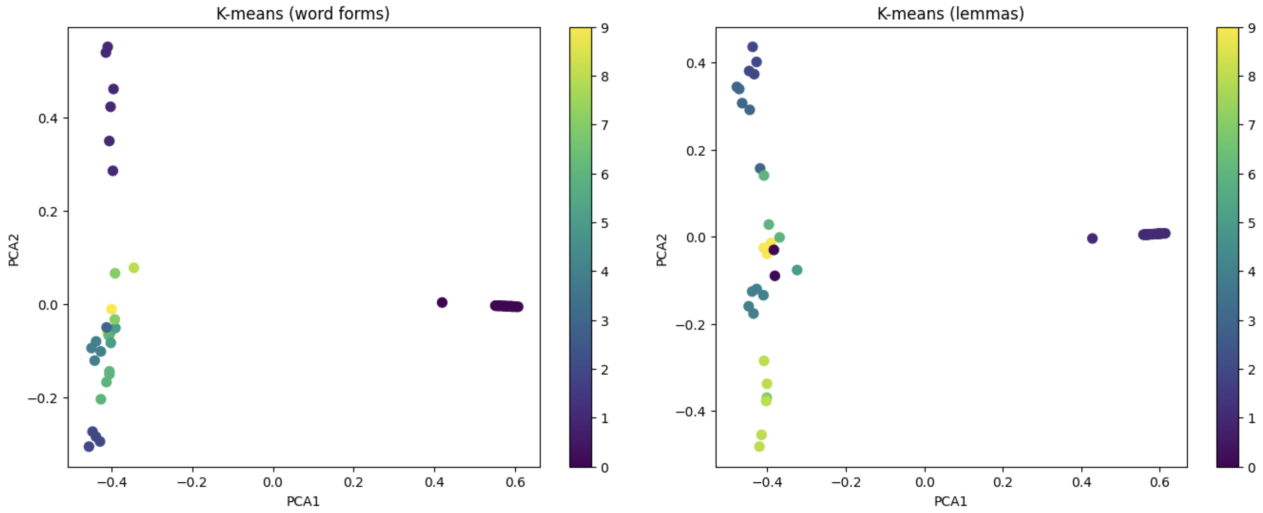
а) словоформы б) леммы

Рисунок 4.3 — Результаты кластеризации методом K-средних, 7 кластеров



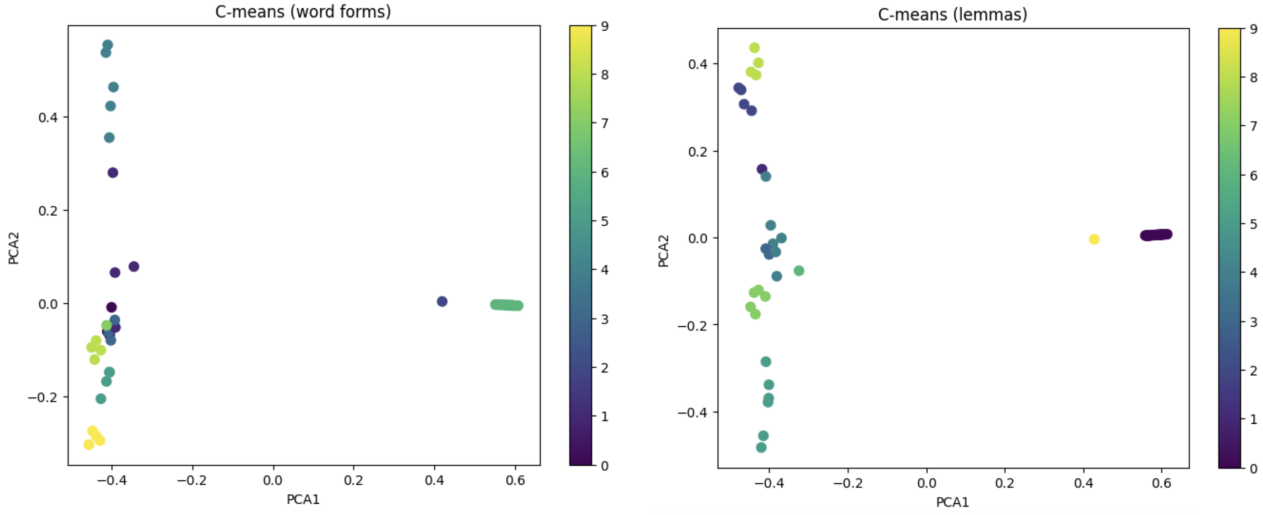
а) словоформы б) леммы

Рисунок 4.4 — Результаты кластеризации методом C-средних, 7 кластеров



а) словоформы б) леммы

Рисунок 4.5 — Результаты кластеризации методом K-средних, 10 кластеров



а) словоформы б) леммы

Рисунок 4.6 — Результаты кластеризации методом C-средних, 10 кластеров

4.3 Сравнение метрик кластеризации

Таблица 4.1 — Сравнение метрик кластеризации для различных подходов

Метод	Ср. внутрикластерное расст-е			Ср. межкластерное расст-е		
Количество кластеров	3	7	10	3	7	10
K-средние (словоформы)	1.00	0.92	0.85	0.54	0.76	0.93
K-средние (леммы)	0.96	0.89	0.82	0.57	0.78	0.95
C-средние (словоформы)	1.00	0.84	0.89	0.55	0.79	0.90
C-средние (леммы)	0.97	0.88	0.83	0.57	0.79	0.98

Вывод

Время выполнения программы в среде Google Colab составило приблизительно 1 минуту. Данный эксперимент был проведён с использованием стандартных вычислительных ресурсов, включая графические ускорители, повышающие производительность обучения.

Качество кластеризации можно оценить как функцию, прямо зависящую от межкластерного расстояния и обратно зависящую от внутрикластерного расстояния. Так, алгоритм K-средних показал наилучшие результаты кластеризации текстовых данных. Лемматизация повышает компактность кластеров и улучшает их разделение, также повышая качество кластеризации. В то же время алгоритм C-средних демонстрирует более размытые границы кластеров, особенно при работе с леммами. Визуализация дополнительно подчёркивает, что выбор метода кластеризации и предварительной обработки текста (лемматизация или её отсутствие) существенно влияет на качество результатов.

ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы была проведено сравнение алгоритмов кластеризации векторов методами K-средних и C-средних. Все поставленные задачи были выполнены.

1) Проведена кластеризация заданного набора данных.
2) Определены результаты работы при различном указываемом количестве кластеров.
3) Определено среднее внутрикластерное расстояние и среднее межкластерное расстояние для каждого рассматриваемого случая.

Качество кластеризации текстовых данных методами K-средних и C-средних зависит от выбора метода предварительной обработки текстов, а также от близости количества выбранных кластеров к реальному. Алгоритм K-средних показал наилучшие результаты кластеризации текстовых данных, особенно при использовании лемматизации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Python [Электронный ресурс]. — Режим доступа, URL: https://www.python.org/ (дата обращения: 19.11.2024).

2. scikit-learn: Machine Learning in Python [Электронный ресурс]. — Режим доступа, URL: https://scikit-learn.org/stable/ (дата обращения: 25.12.2024).

3. pymorphy3: Morphological analyzer (POS tagger + inflection engine) for Russian language [Электронный ресурс]. — Режим доступа, URL:https://pypi.org/project/pymorphy3/ (дата обращения: 25.12.2024).

4. Google Colab [Электронный ресурс]. — Режим доступа, URL: https://colab.research.google.com/drive/1TTJqgHALWtEPX255K11s8Bgb_1Jq2mKU?usp=sharing (дата обращения: 25.12.2024).