Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Воронежский государственный университет»

Лабораторная работа №13

Предмет: «Введение в компьютерную лингвистику»

Доклад по теме: «Графовые методы для анализа текста»

Работу выполнила:
Обучающаяся Искорнева А.Р.
Математический факультет
Направление: Информационно-аналитические
системы безопасности
1 курс, группа 5, подгруппа 2
Преподаватель: Донина О.В.

Воронеж 2025г.

1. Введение в понятие графовых методов для анализа текста

Анализ текстовых данных — одна из ключевых задач современной обработки естественного языка (Natural Language Processing). В последние годы всё большее внимание уделяется методам представления и анализа текста с использованием графов. Графовые методы для анализа текста представляют собой подходы, в которых текст моделируется как графовая структура, а затем анализируется с помощью алгоритмов теории графов и машинного обучения.

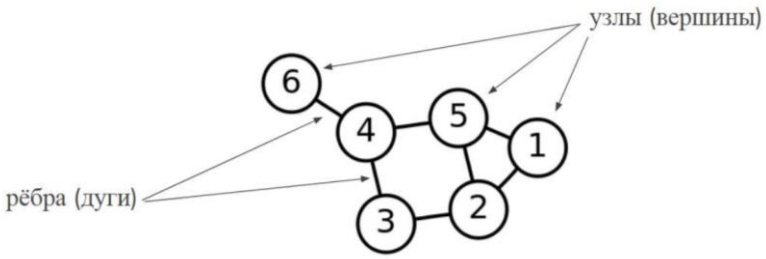


Рисунок 1

Идея заключается в том, чтобы представить элементы текста — такие как слова, фразы, предложения или абзацы — в виде узлов графа, а связи между ними — в виде рёбер. Эти связи могут быть основаны на различных принципах: частоте совместного появления слов, семантической близости, синтаксических зависимостях, логической связности предложений и других характеристиках. Таким образом, текст преобразуется в формат, который позволяет применять мощные инструменты графового анализа.

Применение графов в анализе текста имеет ряд преимуществ. Во-первых, графы позволяют наглядно представить сложные взаимосвязи внутри текста. Во-вторых, они дают возможность использовать количественные метрики, такие как степень узла, центральность, плотность подграфов, которые помогают выделять важные элементы текста, находить скрытые закономерности и строить более точные модели. В-третьих, графовый подход может служить основой для применения современных методов машинного обучения, таких как Graph Neural Networks (GNN), что особенно актуально в эпоху развития искусственного интеллекта.

Эти методы находят широкое применение в различных задачах: от автоматического резюмирования и извлечения ключевых слов до классификации текстов и анализа тональности. Они также эффективны при работе с большими объемами информации, например, при анализе научных публикаций, социальных сетей или юридических документов.

Таким образом, графовые методы становятся важным инструментом в области анализа текста, обеспечивая гибкость, интерпретируемость и глубину анализа, недоступную традиционным подходам. Их развитие открывает новые возможности для исследования структуры языка и построения интеллектуальных систем, способных эффективно обрабатывать и понимать текстовую информацию.

2. Строительство графов из текстовых данных

Рассмотрим основные этапы по созданию графов на основе текстовых данных:

1 этап - предобработка текста

Перед построением графа текст проходит стандартную предобработку:

- Токенизация (разделение на слова или предложения)
- Лемматизация или стемминг
- Удаление стоп-слов
- Извлечение именованных сущностей (NER)

2 этап - выбор узлов графа:

Узлами графа могут быть различные элементы текста:

- Слова или леммы — наиболее распространённый вариант.
- Фразы или n-граммы — позволяют учитывать контекст и составные выражения.
- Предложения или абзацы — подходят для анализа структуры текста на более высоком уровне.
- Сущности (имена собственные, термины) — особенно полезно при работе с фактологическими текстами и онтологиями.

3 этап - определение рёбер графа:

Рёбра отражают связи между узлами. Существует несколько способов их установления:

- Ко-оккуррентность (совместное появление): два слова считаются связанными, если они встречаются в одном контексте (например, в одном предложении или окне из N слов).
- Синтаксические зависимости: связи между словами в предложении, основанные на грамматических отношениях (подлежащее – сказуемое и т.д.).
- Семантическая близость: устанавливается на основе векторных представлений слов (word embeddings) или косинусной меры схожести.
- Логическая связь: используется в задачах анализа дискурса, где предложения соединяются на основе причинно-следственных или временных связей.

4 этап - взвешивание рёбер:

Рёбра могут быть невзвешенными (просто наличие/отсутствие связи) или взвешенными. Вес может отражать: частоту совместного появления, степень семантической близости, силу синтаксической зависимости и др.

5 этап - определение типа графа:

В зависимости от задачи можно использовать:

- Ориентированные графы — если связи имеют направление (например, в синтаксических зависимостях).
- Неориентированные графы — когда важна только сама связь без направления.

- Мультиграфы — если допускается наличие нескольких рёбер между одними и теми же узлами.
- Гиперграфы — для моделирования сложных многокомпонентных взаимодействий.

6 этап – визуализация графа

Отображения узлов и рёбер графа в графическом виде, чтобы сделать структуру наглядной и понятной для человека.

3. Методы вычисления и анализа графов для текста

После того как текстовые данные преобразованы в граф, наступает этап его анализа — вычисления метрик, выявления структурных особенностей, поиска ключевых элементов и извлечения полезной информации. В этом процессе используются методы теории графов, алгоритмы машинного обучения и инструменты сетевого анализа.

Рассмотрим основные методы вычисления и анализа графов в задачах обработки текста:

1. Метрики центральности (Centrality Measures)

Эти метрики помогают определить наиболее "важные" узлы графа — слова, фразы или предложения, играющие ключевую роль в структуре текста.

- *Степень узла* (Degree Centrality): количество связей у узла. Чем выше степень, тем более связано слово с другими.
- *Центральность по близости* (Closeness Centrality): отражает, насколько быстро узел может достичь других узлов графа. Полезно для поиска информативных слов.
- *Центральность по посредничеству* (Betweenness Centrality): показывает, насколько часто узел лежит на кратчайших путях между другими узлами. Может указывать на "перекрестья" смыслов.
- *Векторная центральность* (Eigenvector Centrality): учитывает не только количество соседей, но и их важность. Например, популярна в PageRank.

2. Обнаружение сообществ (Community Detection)

Обнаружение сообществ — это процесс выявления групп узлов графа, которые более плотно связаны между собой внутри группы, чем с узлами из других групп. Такие группы часто называют сообществами или кластерами.

Основные алгоритмы обнаружения сообществ:

- *Алгоритм Лувена* (Louvain Method) - один из самых популярных алгоритмов. Работает быстро и эффективно даже на больших графах.

Принцип работы:

Проходится по узлам и пытается присоединить их к соседним сообществам, если это увеличивает *модулярность* графа (*модулярность* (modularity): мера, показывающая, насколько хорошо граф разбит на сообщества. Чем выше значение, тем лучше разбиение).

Повторяется до тех пор, пока улучшений не будет.

- *Label Propagation Algorithm* (LPA) - простой и быстрый алгоритм, основанный на распространении меток среди узлов.

Принцип работы:

Каждому узлу случайно присваивается метка (номер кластера).

Узлы последовательно меняют свои метки на ту, которая чаще встречается среди соседей.

Процесс продолжается до стабилизации.

- *Spectral Clustering* - метод, использующий собственные значения матрицы смежности графа.

Принцип работы:

Строится матрица Лапласа графа.

Вычисляются собственные векторы этой матрицы.

На основе этих векторов применяется классический алгоритм кластеризации K-means.

3. Поиск кратчайших путей и связности

Изучение топологии графа позволяет понять, насколько он целостен и как связаны его части.

Связность графа: определение, является ли граф связным или состоит из нескольких компонент.

Расстояние между узлами: может служить мерой семантической или контекстуальной близости.

4. PageRank и его аналоги

PageRank — один из самых известных алгоритмов, используемых в графовом анализе. Он оценивает важность узла на основе количества и качества входящих ссылок.

- TextRank: адаптация PageRank для задач NLP — используется в извлечении ключевых слов и автоматическом резюмировании.
- HITS (Hyperlink-Induced Topic Search): определяет авторитетные узлы.

5. Меры плотности и связности графа

Для оценки общих характеристик графа применяются следующие меры:

- Плотность графа (Graph Density): отношение числа рёбер к максимально возможному числу.
- Коэффициент кластеризации (Clustering Coefficient): мера, показывающая, насколько вероятно объединение узлов в треугольники.
- Количество компонент связности: говорит о том, насколько текст представляет собой единую структуру.

6. Графовые нейронные сети (Graph Neural Networks, GNN)

С развитием глубокого обучения всё чаще используются методы, которые работают напрямую с графами.

- GCN (Graph Convolutional Networks)
- GAT (Graph Attention Networks)
- GraphSAGE — генерация эмбеддингов для новых узлов без переобучения всей сети

4. Примеры построения графов из текстовых данных

1. Граф ко-оккуррентности слов (Ко-оккурентность в лингвистике — это повышенная частота упорядоченного появления двух соседних терминов в корпусе текста)

Граф ко-оккуррентности (co-occurrence graph) — один из самых простых и популярных способов представления текста в виде графа. Узлы — это слова, а рёбра — частота совместного появления слов в определённом контексте (например, в окне из N слов).

Метод:

- Токенизация текста
- Построение контекстных окон
- Подсчёт пар слов, встречающихся вместе
- Построение графа с весами рёбер = количеству совпадений

Пример кода:

```python
import networkx as nx
from nltk import word_tokenize
from itertools import combinations
from collections import defaultdict

text = "Графовые методы анализа текста позволяют выявлять скрытые структуры в тексте."

# Токенизация
words = word_tokenize(text.lower())

# Создание графа
G = nx.Graph()

# Размер окна
window_size = 3

# Скользящее окно
for i in range(len(words)):
    window = words[i:i+window_size]
    for w1, w2 in combinations(window, 2):
        G.add_edge(w1, w2)

# Визуализация
nx.draw(G, with_labels=True)
```

Рисунок 2

Результат: граф, где узлы — слова, а ребра соединяют те, которые встречались рядом.

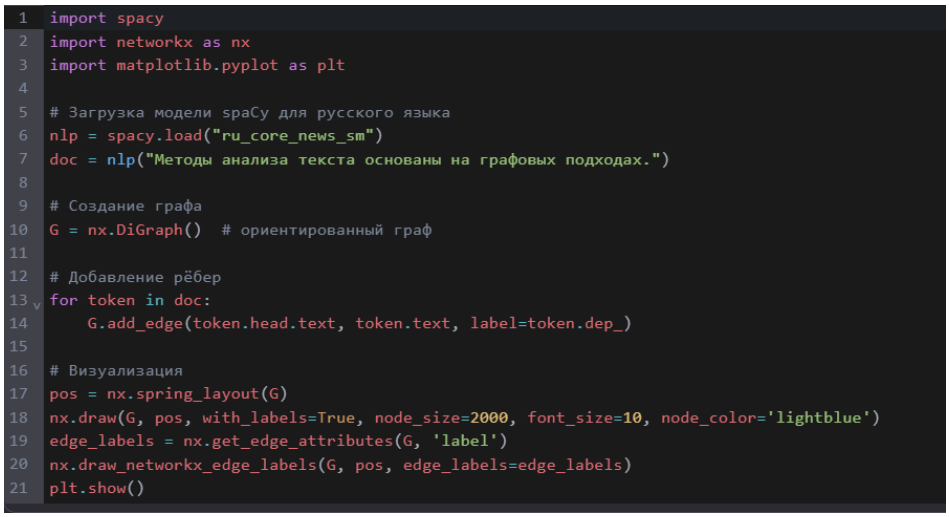
1. Граф зависимостей предложения

Синтаксический граф зависимостей строится на основе грамматического разбора предложений. Каждое слово — узел, а ребро отражает грамматическую связь (например, подлежащее — сказуемое).

Метод:

- Использование парсера зависимостей (например, SpaCy или StanfordNLP)
- Извлечение отношений между словами

Пример кода:



```python
import spacy
import networkx as nx
import matplotlib.pyplot as plt

# Загрузка модели spaCy для русского языка
nlp = spacy.load("ru_core_news_sm")
doc = nlp("Методы анализа текста основаны на графовых подходах.")

# Создание графа
G = nx.DiGraph()  # ориентированный граф

# Добавление рёбер
for token in doc:
    G.add_edge(token.head.text, token.text, label=token.dep_)

# Визуализация
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=2000, font_size=10, node_color='lightblue')
edge_labels = nx.get_edge_attributes(G, 'label')
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
plt.show()
```

Рисунок 3

Результат: ориентированный граф с отношениями типа nsubj (подлежащее), amod (определение), case и т. д.

2. Тематический граф / семантическая сеть

В этом графе узлы представляют ключевые термины, а рёбра — семантические связи между ними. Такие графы часто используются в тематическом моделировании и анализе знаний.

Метод:

- Извлечение ключевых слов (TextRank, RAKE и др.)
- Вычисление семантической близости (BERT, Word2Vec)
- Построение графа с весами на основе схожести

Пример кода:

```python
from sentence_transformers import SentenceTransformer
import numpy as np
import networkx as nx
from sklearn.metrics.pairwise import cosine_similarity

# Загрузка модели BERT
model = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')

terms = ["граф", "метод", "текст", "анализ", "данные"]
embeddings = model.encode(terms)

# Вычисление матрицы схожести
sim_matrix = cosine_similarity(embeddings)

# Построение графа
G = nx.Graph()

threshold = 0.5
for i in range(len(terms)):
    for j in range(i+1, len(terms)):
        if sim_matrix[i][j] > threshold:
            G.add_edge(terms[i], terms[j], weight=sim_matrix[i][j])

# Визуализация
pos = nx.spring_layout(G)
weights = [G[u][v]['weight']*10 for u,v in G.edges()]
nx.draw(G, pos, with_labels=True, width=weights, node_size=800, font_size=10)
plt.show()
```

Рисунок 4

Результат: граф, в котором узлы связаны, если они семантически близки.

Применение графов для разрешения сущностей в тексте

Разрешение сущностей (Entity Resolution, иногда называемое Entity Disambiguation) — это задача обработки естественного языка (NLP), заключающаяся в определении того, какая именно реальная сущность имеется в виду при упоминании в тексте. Например, слово "Apple" может относиться к компании Apple Inc., к фрукту или одноимённой группе.

Графовые методы всё чаще применяются для решения этой задачи благодаря своей способности моделировать сложные связи между сущностями и их контекстами. Ниже рассмотрим, как графы используются в разрешении сущностей.

Разрешение сущностей (Entity Disambiguation) — выбор правильной сущности из возможных вариантов по её упоминанию в конкретном контексте.

Например:

- Упоминание "Барселона" может означать город в Испании или футбольный клуб.
- Слово "Москва" — столица России или одноимённый город в Беларуси.

Представление знаний в виде графа позволяет эффективно моделировать связи между сущностями, их атрибутами и контекстом. В основе подхода лежит идея, что правильная сущность лучше соответствует своему окружению в графе знаний.

Основные компоненты графового подхода:

компонент	описание
узлы графа	представляют сущности (например, "альберт эйнштейн", "теория относительности")
рёбра графа	отражают отношения между сущностями (например, "родился в", "создал")
контекст упоминания	предложение или абзац, где встречается неоднозначное имя

веса ребер	могут отражать силу связи между сущностями или частоту совместного появления

Примеры графовых структур:

1. Knowledge Graphs (Графы знаний)

Наиболее известны такие графы, как:

Wikidata / DBpedia / YAGO — содержат миллионы сущностей и связей между ними.

Google Knowledge Graph — используется в поиске Google для уточнения значений запросов.

Как используется:

При упоминании “Солнечная система” алгоритм проверяет, какие соседние сущности связаны с этим понятием в графе.

Если рядом упоминаются "планеты", "Земля", "Сатурн", то скорее всего имеется в виду астрономическая система, а не название фильма.

2. Семантические сети

Моделируют сущности и их семантические связи. Применяются в системах искусственного интеллекта для понимания контекста.

3. Графы ко-оккуррентности

Строятся на основе анализа текстов: если две сущности часто встречаются вместе, они соединяются ребром. Это помогает выявлять тематические связи.

Алгоритмы и подходы:

1. Random Walk / Personalized PageRank

Алгоритмы случайного блуждания по графу позволяют оценить, насколько вероятно, что та или иная сущность связана с другими упомянутыми в тексте.

Пример:

- В тексте встретились "Эйнштейн", "физика", "теория".
- Запускается случайное блуждание по графу знаний, начиная с этих узлов.
- Сущность "Альберт Эйнштейн" получает наибольший вес → выбирается как правильный вариант.

2. Graph Neural Networks (GNN)

Современные методы машинного обучения, работающие с графами:

GCN (Graph Convolutional Networks)

GAT (Graph Attention Networks)

Как работает:

Каждая сущность кодируется вектором, учитывающий не только собственные признаки, но и связь с соседними узлами. На основе этих эмбеддингов делается предсказание, какая сущность наиболее корректна в текущем контексте.

3. Entity Embeddings

Сущности кодируются в числовые векторы, основываясь на их расположении в графе. Чем ближе векторы, тем выше вероятность, что сущности связаны.

Применение на практике:

- Поисковые системы — точное понимание запросов
- Чат-боты и ассистенты — корректная интерпретация реплик
- Юридический и научный анализ текста — точное сопоставление терминов
- Системы рекомендаций — понимание контента
- Фактчекинг и детекция дезинформации — проверка утверждений через графы знаний

Визуализация и интерпретация графовых структур

Графовые модели широко используются в анализе текста, машинном обучении и обработке естественного языка (NLP), поскольку позволяют представить сложные взаимосвязи между элементами данных. Однако построение графа — лишь половина дела. Чтобы извлечь полезную информацию, необходимо визуализировать граф и интерпретировать его структуру.

Визуализация графа — это процесс отображения узлов и рёбер графа в графическом виде, чтобы сделать структуру наглядной и понятной для человека. Это особенно важно при работе с большими и сложными графами, например, графами ко-оккуррентности слов, семантическими сетями или онтологиями.

Основные задачи визуализации:

- Выделение центральных узлов
- Обнаружение сообществ и кластеров
- Понимание связей между элементами
- Поиск аномалий или выбросов
- Представление результатов анализа в доступной форме

Методы визуализации графов:

1. Force-directed layout (алгоритм пружин)

- Узлы притягиваются к связанным узлам и отталкиваются от несвязанных.
- Пример: алгоритмы Fruchterman-Reingold, Kamada-Kawai.

Плюсы: интуитивно понятная структура
Минусы: может быть медленным на больших графах

2. Круговая / радиальная раскладка

- Узлы размещаются по окружности или радиально вокруг центрального узла.

Используется: когда есть явный центральный узел (например, ключевое слово)

3. Слоистая (hierarchical) раскладка

- Используется для ориентированных ациклических графов (DAG).
- Полезна для визуализации причинно-следственных связей, деревьев решений и т.п.

4. Спектральная раскладка

- Основана на собственных векторах матрицы смежности графа.
- Хорошо выделяет кластеры.

5. Слои и фильтрация

- Можно отображать только подграфы, соответствующие определённым условиям (например, только самые важные узлы).