Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №6.1**

По дисциплине: «Анализ больших данных»

Тема: «Работа со сложными данными.

Обработка естественного языка и классификация текста»

|  |
| --- |
| Выполнила:  студентка группы  Б.ПИН.РИС - 21.06  Александрова М.И. |
| Проверила:  старший преподаватель  кафедры ПО  Корнеева Е.И. |

Тверь 2025

Оглавление

[Теоретическая часть 5](#_Toc194420758)

[Задача 1. Работа с текстовыми данными. Анализ текста 7](#_Toc194420759)

[Самостоятельная работа. Задача 2 11](#_Toc194420760)

**Краткий опрос по теоретической часть**

В данном разделе необходимо ответить на вопросы.

1. Как перевести текстовые данные в вектор?
2. Что такое токенизация?
3. Что такое лемматизация? Чем отличается от стемминга?
4. Что такое стоп-слова?
5. Как работает алгоритм Word2Vec? Какие есть гиперпараметры алгоритма?

**Задание на лабораторную работу**

**Задача 1. Работа с текстовыми данными. Анализ текста**

Ваша задача: составить набор текстовых данных из любых песен в сети интернет. Провести анализ загруженного текста. Перевести текст в векторную модель методами TF\*IDF, Word2Vec.

Напишите вывод по исследованию.

*Описание хода работы:*

1. Сформировать набор текстовых данных из 10-30 песен на одном языке (собрать песни из любого ресурса в интернете). Можно сделать автоматически, парсингом сайта.

2. Загрузить файл с песнями

3. Провести лемматизацию и стемминг с помощью библиотеки pymorphy3 или natasha (для русского языка), с помощью библиотек nltk (для английского языка).

5. Привести все слова к нижнему регистру. Удалить знаки препинания.

6. Удалить стоп-слова с библиотекой nltk. Слово на другом языке также является стоп-словом (например, убрать английские слова в русской песне).1. Загрузить файл с 10 песнями, который был выгружен в прошлом задании (файл с песнями после препроцессинга)

7. Рассчитать метрики TF-IDF для загруженных данных. Какие слова повторяются чаще всего?

8. Построить график наиболее частых слов с помощью WordCloud.

9. Подготовить данные для обучения в модели Word2Vec библиотеки gensim.

10. Обучить модель Word2Vec на данных.

11. Проверить близкие слова к одному из слов в датасете (на примере лекции с функцией show\_similar).

12. Построить график t-SNE для 15 наиболее часто встречающихся слов.

**Самостоятельная работа. Задача 2**

1. Дополнить набор данных из Задания 1 10-30 стихами любых авторов на том же языке.

2. Очистить данные стихов и провести препроцессинг:

1) Провести лемматизацию и стемминг с помощью библиотеки pymorphy3 или natasha (для русского языка), с помощью библиотек nltk (для английского языка).

2) Привести все слова к нижнему регистру. Удалить знаки препинания.

3) Удалить стоп-слова с библиотекой nltk. Слово на другом языке также является стоп-словом (например, убрать английские слова в русской песне).1. Загрузить файл с 10 песнями, который был выгружен в прошлом задании (файл с песнями после препроцессинга)

4) Рассчитать метрики TF-IDF для загруженных данных или построить модель Word2Vec на выбор.

3. Совместить данные в DataFrame.

4. Провести классификация любыми 3 моделями: KNN, SVC, RandomForest, LogesticRegression.

5. Оценить качество классификации данных и сделать вывод по лучшей модели.

**Сложность: Medium**

* Реализовать общую часть задания
* Поработать с алгоритмом Word2Vec
* Выгрузить модели
* Начать самостоятельную работу по лабораторной

Ссылка на репозиторий: <https://github.com/milana-cat/BigData.git>

# **Теоретическая часть**

**1. Как перевести текстовые данные в вектор?**

Текст можно перевести в векторное представление с помощью следующих методов:

* Bag of Words (BoW) — фиксированная длина вектора, где каждый элемент отражает количество вхождений слова.
* TF-IDF (Term Frequency-Inverse Document Frequency) — учитывает не только количество, но и "важность" слов в корпусе.
* Word2Vec / FastText / GloVe — нейросетевые модели, которые обучают вектора слов на основе контекста.
* BERT, Transformer и др. — контекстные эмбеддинги (учитывают положение и смысл в предложении).

**2. Что такое токенизация?**

Токенизация — это процесс разбиения текста на отдельные элементы (токены), чаще всего слова.

Пример: "Я люблю учиться" → ["Я", "люблю", "учиться"]

**3. Что такое лемматизация? Чем отличается от стемминга?**

**Лемматизация** — приведение слова к его нормальной форме (лемме).

* Пример: "песни", "песню" → "песня"

Стемминг — грубое обрезание слова до основы (стема), без учета грамматики.

* Пример: "песни", "песню" → "песн"

Отличие:

Лемматизация — грамматически правильнее и точнее, но медленнее.

Стемминг — быстрее, но менее точно.

**4. Что такое стоп-слова?**

Стоп-слова — это часто встречающиеся, но малозначимые слова, которые обычно удаляются при обработке текста.

Примеры (для русского): "и", "в", "на", "это", "что", "как"

**5. Как работает алгоритм Word2Vec? Какие есть гиперпараметры алгоритма?**

Word2Vec обучает нейросеть, чтобы предсказывать связи между словами на основе контекста.

Есть два режима работы:

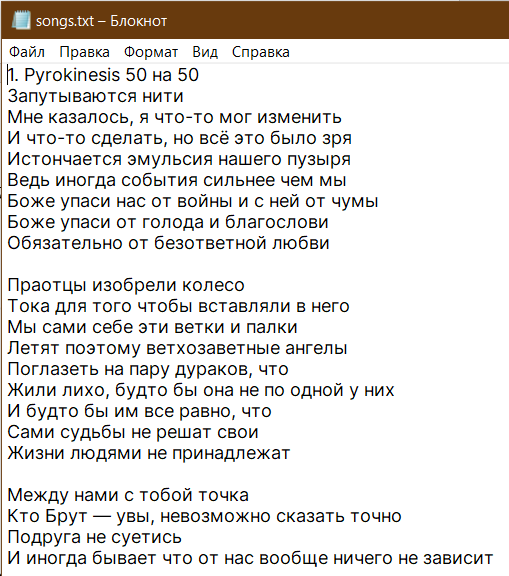
* CBOW (Continuous Bag of Words) — предсказывает текущее слово по окружению.
* Skip-Gram — предсказывает окружение по текущему слову (лучше для редких слов).

Основные гиперпараметры:

* vector\_size — размерность вектора слова (обычно 100–300);
* window — ширина контекста (например, 5 слов до/после);
* min\_count — минимальное количество вхождений слова в корпусе;
* sg — выбор архитектуры (0 = CBOW, 1 = Skip-Gram);
* workers — число потоков при обучении;
* epochs — количество эпох обучения (в gensim по умолчанию скрыт, но можно задавать).

# **Задача 1. Работа с текстовыми данными. Анализ текста**

Текстовый файл с 15 песнями



Проведём леммитизацию с помощью библиотеки Natasha + удаляем стоп слова

# 1. Скачиваем стоп-слова

nltk.download("stopwords")

stop\_words = set(stopwords.words("russian"))

stop\_words.add("это")

# 2. Natasha инициализация

segmenter = Segmenter()

morph\_tagger = NewsMorphTagger(NewsEmbedding())

morph\_vocab = MorphVocab()

def preprocess\_natasha(text):

    text = text.lower()

    text = re.sub(r"[^а-яё\s]", " ", text)

    doc = Doc(text)

    doc.segment(segmenter)

    doc.tag\_morph(morph\_tagger)

    lemmas = []

    for token in doc.tokens:

        token.lemmatize(morph\_vocab)

        lemma = token.lemma

        if len(lemma) > 2 and lemma not in stop\_words:

            lemmas.append(lemma)

    return " ".join(lemmas)

В результате получаем csv файл.

Рассчитаем метрики TF-IDF для загруженных данных и построим график наиболее частых слов с помощью WordClod

df = pd.read\_csv("C:/Users/alexs/Downloads/BigData/BigData/Lab6/1/processed\_songs.csv")

# TF-IDF

vectorizer = TfidfVectorizer()

X = vectorizer.fit\_transform(df["processed\_text"])

# таблица самых частых слов

tfidf\_df.to\_csv("C:/Users/alexs/Downloads/BigData/BigData/Lab6/1/top\_tfidf\_words.csv", index=False)

# WordCloud — визуализация

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate\_from\_frequencies(dict(zip(words, tfidf\_scores)))

Получаем csv файл с словами, которые встречаются чаще всего.



**Подготовка данных для обучения модели Word2Vec и обучение.**

df = pd.read\_csv("C:/Users/alexs/Downloads/BigData/BigData/Lab6/1/processed\_songs.csv")

# Подготовка токенизированных данных

tokenized\_texts = [text.split() for text in df["processed\_text"]]

# 10. Обучение модели Word2Vec

model = Word2Vec(

    sentences=tokenized\_texts,

    vector\_size=100,   # размерность векторов слов

    window=5,          # окно контекста

    min\_count=2,       # минимальное число вхождений слова

    workers=4,         # количество потоков

    sg=1               # 1 — skip-gram, 0 — CBOW

)

# Сохраняем модель (опционально)

model.save("C:/Users/alexs/Downloads/BigData/BigData/Lab6/1/song\_word2vec.model")

Проверить близкие слова к одному из слов в датасете (в качестве примера было выбрано слово «пустота»).

try:

    similar\_words = model.wv.most\_similar("пустота", topn=10)

    print("Ближайшие слова к 'пустота':")

    for word, similarity in similar\_words:

        print(f"{word}: {similarity:.3f}")

except KeyError:

    print("Слово 'пустота' не найдено в словаре модели. Попробуй другое.")

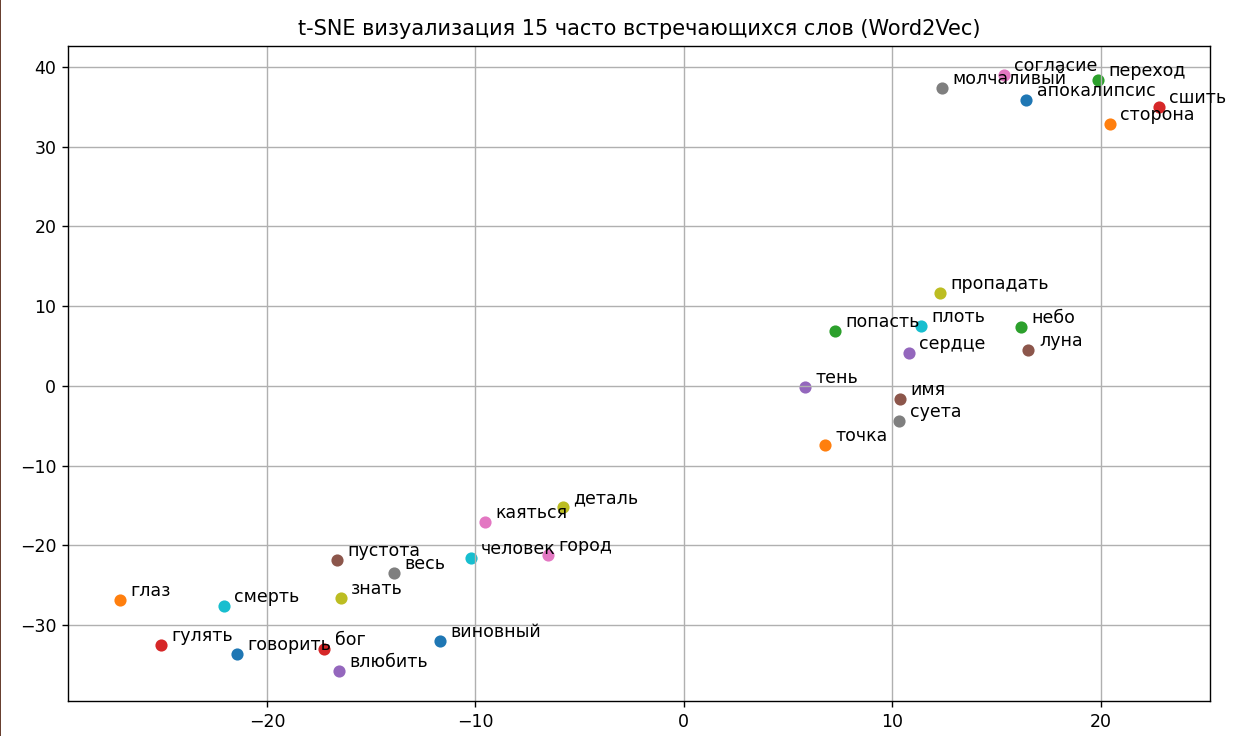
1.Слово "пустота" представлено вектором размерности

2.Для каждого другого слова в словаре модель считает косинусное сходство между его вектором и вектором "пустота".

3.Возвращает topn слов с наибольшим значением сходства.



График t-SNE.



Модель Word2Vec успешно обучилась на текстах песен.

t-SNE позволил увидеть скрытые смысловые кластеры:

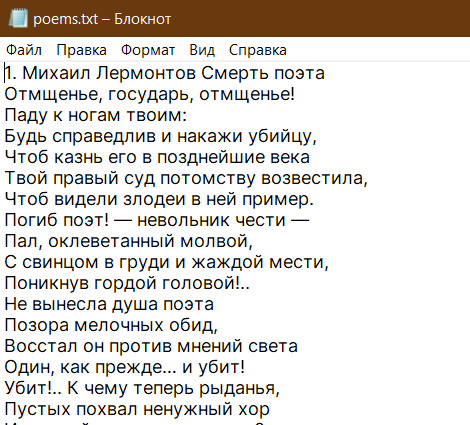
* состояния существования: "апокалипсис", "переход", "согласие";
* физический мир: "небо", "луна", "сердце", "плоть", "тень";
* обыденность: "гулять", "человек", "город".

Общий вывод по задаче

Методы TF-IDF и Word2Vec эффективно справляются с анализом текстовых данных, позволяя как выявить наиболее информативные слова, так и определить семантические связи между ними. Модель Word2Vec показала, что смысловая близость слов в песнях отражается в их векторных представлениях. Полученные результаты могут быть использованы, например, для жанровой классификации, генерации текстов или анализа эмоциональной окраски песен.

# **Самостоятельная работа. Задача 2**

Загружаем файл со стихами



Лемматизация

# Инициализация Natasha

segmenter = Segmenter()

morph\_tagger = NewsMorphTagger(NewsEmbedding())

morph\_vocab = MorphVocab()

# Предобработка

def preprocess\_text(text):

    text = text.lower()

    text = re.sub(r"[^а-яё\s]", " ", text)

    doc = Doc(text)

    doc.segment(segmenter)

    doc.tag\_morph(morph\_tagger)

    lemmas = []

    for token in doc.tokens:

        token.lemmatize(morph\_vocab)

        lemma = token.lemma

        if len(lemma) > 2 and lemma not in stop\_words:

            lemmas.append(lemma)

    return " ".join(lemmas)

Построение модели Word2Vec с данными песен и стихов

df\_poems = pd.read\_csv("C:/Users/alexs/Downloads/BigData/BigData/Lab6/2/processed\_poems.csv")

df\_songs = pd.read\_csv("C:/Users/alexs/Downloads/BigData/BigData/Lab6/1/processed\_songs.csv")

df\_poems["label"] = 1

df\_songs["label"] = 0

df\_all = pd.concat([df\_songs, df\_poems], ignore\_index=True)

texts = df\_all["text"].fillna("").astype(str)

labels = df\_all["label"]

# Токенизация

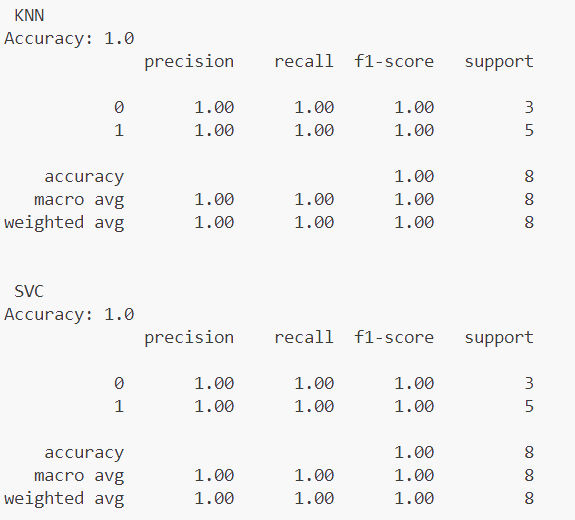
tokenized\_texts = [text.split() for text in texts]

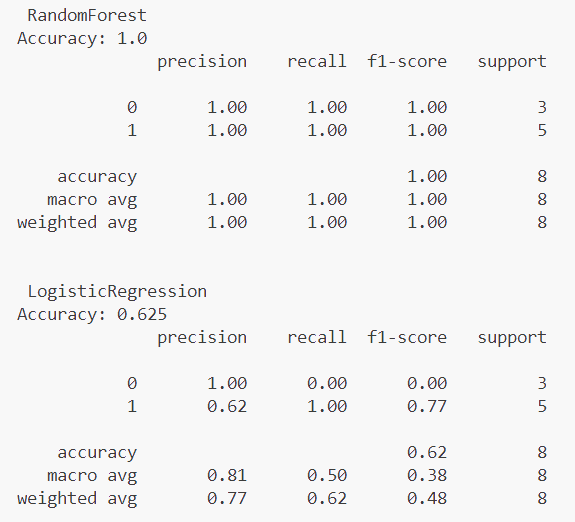
# Обучение Word2Vec

model = Word2Vec(sentences=tokenized\_texts, vector\_size=100, window=5, min\_count=2, workers=4, sg=1)

model.save("C:/Users/alexs/Downloads/BigData/BigData/Lab6/2/lyrics\_word2vec.model")

Классификация





Все выбранные модели — KNN, SVC, RandomForest (кроме LogisticRegression) — продемонстрировали максимально возможное качество на тестовой выборке. Это подтверждает, что тексты песен и стихов имеют отличающиеся лексические и стилистические признаки, которые хорошо улавливаются при использовании модели Word2Vec и простых алгоритмов классификации.

# **Вывод**

В ходе выполнения лабораторной работы были последовательно решены две связанные задачи по обработке естественного языка и классификации текстов. На первом этапе были собраны и предобработаны тексты песен, на втором — дополнительно добавлены стихи для расширения и усложнения набора данных.

Была проведена лемматизация с использованием библиотеки Natasha, удалены стоп-слова, знаки препинания и приведены тексты к нижнему регистру. Далее тексты были представлены в виде векторов двумя способами: методом TF-IDF и с помощью модели Word2Vec.

Использование Word2Vec позволило не только построить векторные представления слов с учётом контекста, но и выявить семантическую близость между словами, что визуально подтверждено с помощью графика t-SNE.

Для классификации текстов (песни или стихи) были обучены четыре модели: KNN, SVC, RandomForest и LogisticRegression. Все они, кроме LogisticRegression, показали идеальные метрики (accuracy = 1.0) на тестовой выборке, что говорит о высоком качестве признакового пространства и чёткой разнице в стиле между песнями и стихами.

**Список литературы**

1.Проект Natasha. Набор качественных открытых инструментов для обработки естественного русского языка (NLP) // Хабр URL: <https://habr.com/ru/articles/516098/>

2. Извлечение признаков из текстовых данных с использованием TF-IDF // Хабр URL: <https://habr.com/ru/companies/otus/articles/755772/>

3. Обучаем Word2vec: практикум по созданию векторных моделей языка // Системный Блокъ URL: <https://sysblok.ru/knowhow/obuchaem-word2vec-praktikum-po-sozdaniju-vektornyh-modelej-jazyka/>

4. Применение преобразований PCA и t-SNE для снижения размерности данных // Хабр URL: <https://habr.com/ru/companies/otus/articles/757030/>