



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 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.03 Прикладная информатика

БАКАЛАВРСКАЯ ПРОГРАММА 09.03.03/02 Программно-технические средства
информатизации

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Тип практики Преддипломная практика

Название
предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Студент ИУ6-85Б

Нгуен Тхай Ха

(Подпись, дата)

(И.О. Фамилия)

Руководитель практики
от МГТУ им. Н.Э.Баумана

Смирнова Е. В.

(Подпись, дата)

(И.О. Фамилия)

Оценка _____

2024 г.

З А Д А Н И Е на производственную практику

по теме построение подели обработки текстов на языке Python

Студент группы ИУ6-85Б

Нгуен Тхай Ха

(Фамилия, имя, отчество)

Направление подготовки 09.03.03 Прикладная информатика

Бакалаврская программа 09.03.03_02 Программно-технические средства информатизации

Тип практики Преддипломная практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Техническое задание:

разработка моделей обработки текста для веб-приложений. Определение задач моделей, которые необходимо
построить, выбор подходящих инструментов для их построения.

Оформление отчета по практике:

Отчет на 15-25 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Нет

Дата выдачи задания « 07 » февраля 2024 г.

Руководитель практики
от МГТУ им. Н.Э.Баумана

Студент

(Подпись, дата)

Смирнова Е. В.

(И.О. Фамилия)

(Подпись, дата)

Нгуен Тхай Ха

(И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах

Содержание

Введение.....	5
1 Основные задачи обработки текста.....	6
2 Модели предобработки.....	7
2.1 Обработка ПДФ файлов.....	7
2.2 Предобработка текстов	8
3 Модель тематического моделирования	10
3.1 Латентное размещение Дирихле.....	10
3.2 Проектирование в языке Python.....	12
4 Модель косинусного сходства	14
4.1 Косинусное сходство	14
4.2 Встраивание слов.....	15
4.3 Проектирование модели на языке Python	17
ЗАКЛЮЧЕНИЕ	20
СПИСОК ИСПОЛЬЗУЕМОХ ИСТОЧНИКОВ.....	21

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Косинусное сходство (Cosine Similarity) – это мера сходства между двумя векторами предгильбертового пространства, которая используется для измерения косинуса угла между ними

Метод латентного размещения Дирихле (Latent Dirichlet Allocation, LDA) – применяемая в машинном обучении и информационном поиске порождающая модель, позволяющая объяснять результаты наблюдений с помощью неявных групп, благодаря чему возможно выявление причин сходства некоторых частей данных

Тематическое моделирование (Topic Modeling) – как вид статистических моделей для нахождения скрытых тем, встреченных в коллекции документов.

Распределение Дирихле (Dirichlet distribution) – это семейство непрерывных многомерных вероятностных распределений параметризованных вектором α неотрицательных вещественных чисел.

полиномиальное распределение (Multinomial Distribution) – это обобщение биномиального распределения на случай $n > 1$ независимых испытаний случайного эксперимента с $k > 2$ возможными исходами.

непрерывный набор слов (Continuous Bag Of Words – CBOW) – популярный метод обработки естественного языка, используемый для создания вложений слов

Skip Gram – популярный метод обработки естественного языка, используемый для создания вложений слов

Введение

Целью преддипломной практики является описание разработки базовых моделей обработки языка. Эти модели используются в качестве основы для добавления и завершения системы обработки текста приложения. В процессе выполнения работы определяются основные задачи обработки текста, из которых перечисляются необходимые модели. Рассмотрены и выбраны методы и инструменты проектирования моделей.

1 Основные задачи обработки текста

Веб-приложение получает текст пользователя в виде PDF-файла из браузера. Затем файл отправляется на сервер, где его необходимо обработать в соответствии со следующими требованиями:

- содержимое файла, в данном случае текст, необходимо извлечь и сохранить для использования в других задачах;
- извлеченный текст необходимо обработать, чтобы привести его в форму, с которой смогут работать библиотеки и алгоритмы;
- предварительно обработанный текст будет использоваться для расчета необходимых значений для определения: темы текста, смысловой схожести текста с другими текстами.

Для выполнения необходимого расчета значения, описанного выше, были выбраны два метода: косинусное сходство и тематическое моделирование из-за их простоты и популярности в библиотеках языка Python.

Таким образом, основной процесс обработки данных, полученных от пользователей, показан на рисунке 1.

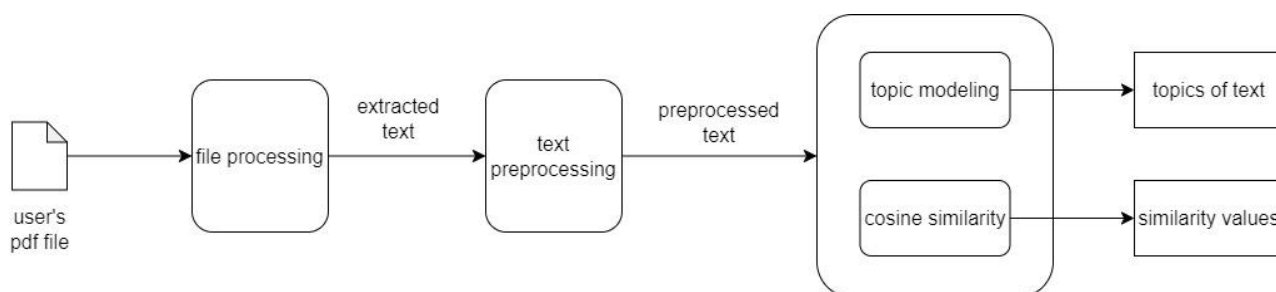


Рисунок 1 - Процесс обработки

2 Модели предобработки

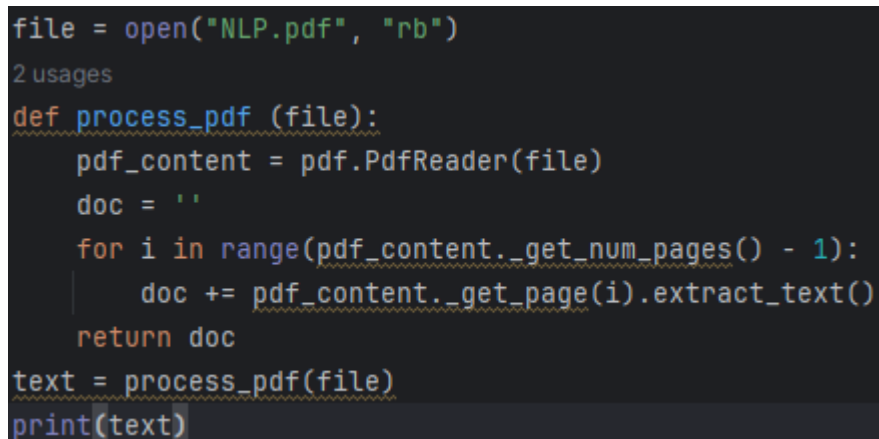
Эта модель решает две задачи: обработку файлов PDF и обработку текста, извлеченного из файла.

2.1 Обработка ПДФ файлов

В языке программирования Python существует множество методов извлечения содержимого из файлов PDF. Согласно требованиям приложения, содержимое, которое необходимо извлечь – это текст в файле. Для этого используется библиотека `pypdf`.

Используя функцию `PdfReader` и функции объекта объектной переменной в библиотеке `pypdf`, текст в файле сохраняется в переменной, которую можно использовать напрямую или при необходимости сохранить в файл.

В примере на рисунке 2 PDF-файл «NLP.pdf» открывается как файловая переменная, затем эта переменная считывается `PdfReader`, ее содержимое сохраняется в переменной `pdf_content`. Затем текстовое содержимое `pdf_content` сохраняется в переменную `doc` и возвращается.

A screenshot of a code editor showing Python code for processing a PDF file. The code opens a file named 'NLP.pdf' in binary mode, defines a function 'process_pdf' that uses 'pdf.PdfReader' to read the file and iterates through its pages to extract text, and then calls this function on the opened file to print the extracted text.

```
file = open("NLP.pdf", "rb")
2 usages
def process_pdf (file):
    pdf_content = pdf.PdfReader(file)
    doc = ''
    for i in range(pdf_content._get_num_pages() - 1):
        doc += pdf_content._get_page(i).extract_text()
    return doc
text = process_pdf(file)
print(text)
```

Рисунок 2 – Простая функция обработки файлов ПДФ

В результате выполнения программы, как показано на рисунке 3, видно, что извлеченный текст соответствует тексту в исходном файле PDF, как показано на рисунке 4.

See discussions, stats, and author profiles for this publication at : <https://www.researchgate.net/publication/319164243>
 Natural Language Processing: State of The Art, Current Trends and Challenges
 Article in Multimedia Tools and Applications · July 2022
 DOI: 10.1007/s11042-022-13428-4
 CITATIONS
 501
 READS
 73,702
 4 authors, including:
 Kiran Khatter
 BML MUNJAL UNIVERSITY
 39 PUBLICATIONS 784 CITATIONS
 SEE PROFILE
 Sukhdev Singh
 Thapar Institute of Engineering and Technology
 23 PUBLICATIONS 911 CITATIONS
 SEE PROFILE
 All content following this page was uploaded by Kiran Khatter on 01 April 2018.
 The user has requested enhancement of the downloaded file. Natural Language Processing: State of The Art, Current Trends and Challenges
 Diksha Khurana¹, Aditya Koli¹, Kiran Khatter^{1,2}
 and Sukhdev Singh^{1,2}
¹Department of Computer Science and Engineering
 Manav Rachna International University, Faridabad -121004, India
²Accendere Knowledge Management Services Pvt. Ltd., India

Рисунок 3 – Результат выполнения программы

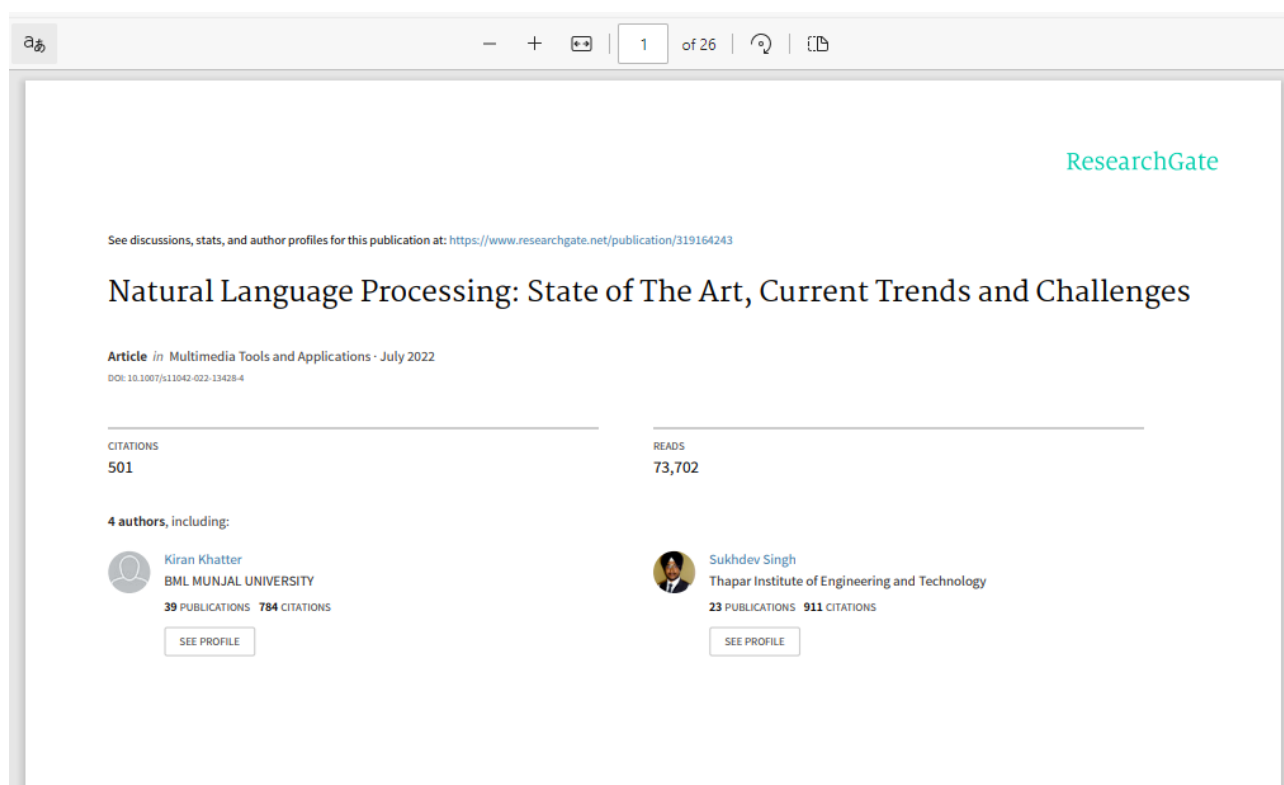


Рисунок 4 – Исходные тексты в файле ПДФ

2.2 Предобработка текстов

Видно, что после извлечения в тексте также присутствуют знаки препинания и переносы строк. Слова в тексте имеют много похожих слов, но пишутся по-разному: заглавные буквы, существительные во множественном числе, разные формы деления глаголов,...

Конечными данными, необходимыми для обработки текста, является список только букв в тексте. Для этого текст должен пройти процесс предварительной обработки, как показано на рисунке 5.

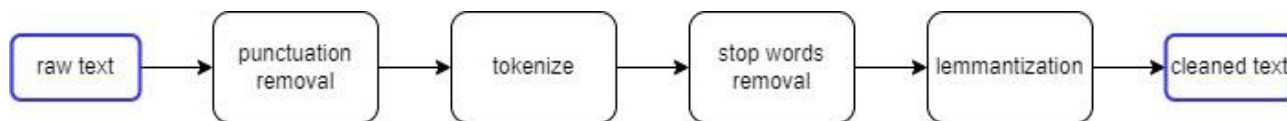


Рисунок 5 – Простой процесс обработки текстов

Для выполнения вышеперечисленных операций используются функции и константы библиотеки `gensim`, кроме того, для выполнения лемматизации используется функция `WordNetLemmatizer` библиотеки `nlTK`. Библиотеки `gensim` и `nlTK` очень популярны и широко используются в области обработки естественного языка.

Основная функция обработки текста показана на рисунке 6. В этой функции после извлечения текста из pdf-файла с помощью функции обработки файла из раздела 2.1 текст возвращается в виде списка слов (токенизация) удаляет стоп-слова (заканчивающиеся слова, не имеющие особого семантического значения), и выполняет лемматизацию. Функция возвращает список слов в тексте, приведенный к инфинитивной форме. Результат выполнения программы представлены на рисунке 7.

```
def preprocess(file):
    doc = process_pdf(file)
    token_list = gensim.utils.simple_preprocess(doc, min_len=2)
    stopwords = gensim.parsing.preprocessing.STOPWORDS
    cleaned_text = [token for token in token_list if token not in stopwords]
    processed_text = [WordNetLemmatizer().lemmatize(word) for word in cleaned_text]
    return processed_text

file = open("NLP.pdf", "rb")
preprocessed_text = preprocess(file)
print(preprocessed_text)
```

Рисунок 6 – Простая функция обработки текстов

```
"D:\ky8\prak\NLP moduls\.venv\Scripts\python.exe" "D:\ky8\prak\NLP moduls\main.py"
['discussion', 'st', 'at', 'author', 'pr', 'ofiles', 'public', 'ation', 'http', 'www',
Process finished with exit code 0
```

Рисунок 7 – Предобработанные тексты

3 Модель тематического моделирования

Тематическое моделирование (topic modeling) как вид статистических моделей для нахождения скрытых тем, встречающихся в коллекции документов. Различные тематические модели используются для анализа текстов, текстовых архивов документов, для анализа изменения тем в наборах документов. Интуитивно понимая, что документ относится к определённой теме, в документах, посвящённых одной теме, можно встретить некоторые слова чаще других.

Тематическое моделирование работает в основном на вероятностных моделях. Среди вероятностных моделей наиболее часто используются две: вероятностный латентно-семантический анализ и латентное размещение Дирихле.

Для реализации тематического моделирования выбрана модель: латентное размещение Дирихле (LDA)

3.1 Латентное размещение Дирихле

Модель LDA можно визуально представить в виде таблички, как показано на рисунке 8. в этой модели зависимости между многими переменными могут быть кратко зафиксированы. Ящики представляют собой «пластины», представляющие реплики, которые представляют собой повторяющиеся объекты. Внешняя пластина представляет документы, а внутренняя пластина представляет повторяющиеся позиции слов в данном документе; каждая позиция связана с выбором темы и слова. Имена переменных определяются следующим образом:

- M – это количество документов;
- N – это количество слов в данном документе (документ i содержит n слов);
- α – это параметр априора Дирихле для распределения тем по документам;
- β – это параметр априора Дирихле для распределения слов по темам;

- θ_i – это распределение тем для документа i ;
- φ_k – это распределение слов по теме k ;
- z_{ij} – это тема для j -го слова в документе i ;
- ω_{ij} – это конкретное слово.

в этой модели слова ω_{ij} являются единственными наблюдаемыми переменными, а остальные переменные являются скрытыми переменными.

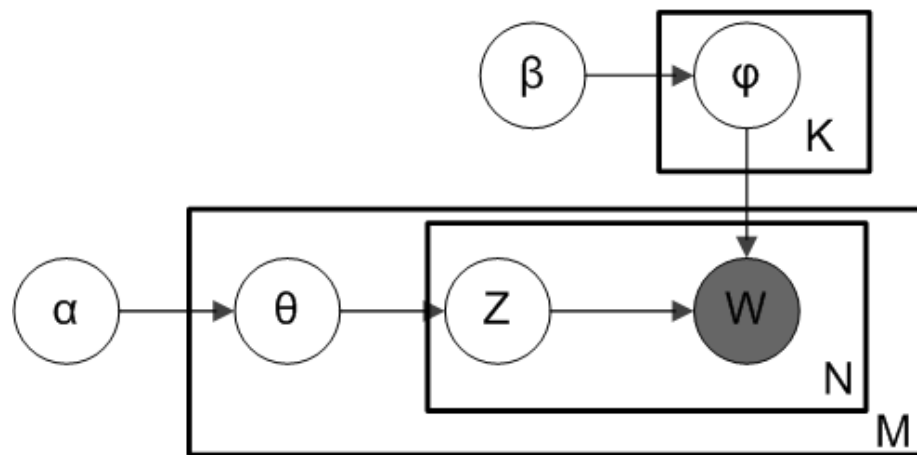


Рисунок 8 – Модель LDA для тематического моделирования

Полученная модель является наиболее широко применяемым на сегодняшний день вариантом LDA. Здесь, K обозначает количество тем, а $\varphi_1, \dots, \varphi_K$ представляют собой V -мерные векторы, хранящие параметры распределенных по Дирихле распределений тем-слов (V – количество слов в словаре).

θ и φ – матрица слов, созданная путем разложения исходного документа, представляющая корпус моделируемых документов. В этом представлении θ состоит из строк, определенных документами, и столбцов, определенных тем, а φ состоит из строк, определенных тем, и столбцов, определенных словами.

Чтобы фактически определить темы в корпусе, создается генеративный процесс для документов следующим образом: документы представляются как случайные смеси по скрытым темам, где каждая тема характеризуется распределением по всем словам. LDA предполагает следующий генеративный процесс для корпуса D , состоящего из M документов, каждый из которых имеет длину N_i , тогда

1. Выбор $\theta_i \sim \text{Dir}(\alpha)$, где $i \in \{1, \dots, M\}$ и $\text{Dir}(\alpha)$ – это Распределение Дирихле с симметричным параметром α , который обычно разрежен ($\alpha < 1$)
2. Выбор $\varphi_k \sim \text{Dir}(\beta)$, где $k \in \{1, \dots, K\}$ и β обычно разрежен
3. для каждого слова в позиции i и j , где $i \in \{1, \dots, M\}, j \in \{1, \dots, N_i\}$:
 - выбор тему $z_{ij} \sim \text{Multinomial}(\theta_i)$
 - выбор слово $\omega_{ij} \sim \text{Multinomial}(\theta_i)$

полиномиальное распределение в этом случае известно как категориальное распределение.

3.2 Проектирование в языке Python

Простая модель LDA построена на языке Python, как показано на рисунке 9. Модель построена на основе библиотеки gensim.

```
def LDA_model (file):
    preprocessed_text = [preprocess(file)]
    dictionary = corpora.Dictionary(preprocessed_text)
    corpus = [dictionary.doc2bow(text) for text in preprocessed_text]
    #training LDA model
    lda = models.LdaModel(corpus=corpus, id2word=dictionary, num_topics=5)
    topics = []
    for idx, topic in lda.print_topics(-1, num_words=10):
        # Extract the top words for each topic and store in a list
        topic_words = [word.split('*')[1].replace(' ', '').strip() for word in topic.split('+')]
        topics.append((f"Topic {idx}", topic_words))
    return topics
file = open("NLP.pdf", "rb")
print("topic modeling testing")
topics = LDA_model(file)
print(topics)
```

Рисунок 9 – Модель LDA для тематического моделирования

В этой модели после обработки текста с помощью функции предварительной обработки из раздела 2.2 на его основе строится словарь. Словарь содержит все слова, встречающиеся в тексте, каждой букве присвоен идентификатор. Затем из словаря формируется корпус (пакет слов), который представляет собой список слов и их количество вхождений в тексте, представленный как (id, n) , где id — идентификатор слова в словаре, а n — сколько раз оно встречается в тексте.

Вероятностная модель LDA обучается на основе словаря и корпуса, созданных ранее, на этом этапе модель генерирует случайные значения для

распределения тем в тексте, затем вычисляет вероятностное распределение слов в тексте, обновляя значение для распределение тем, это повторяется непрерывно, пока не будут учтены все возможности. Более сложную вероятностную модель можно обучать отдельно с большим количеством текстов для получения более точных выводов, такую модель можно сохранить для быстрой и точной работы с большим количеством текстов.

После завершения вероятностной модели она используется для идентификации слов в тексте, которые с вероятностью могут стать темой текста. Результаты запуска модели показаны на рисунке 10. В pdf-файле есть тема обработки естественного языка, видно, что модель задала темы текста, достаточно близкие к содержанию текста.

```
"D:\ky8\prak\NLP moduls\.venv\Scripts\python.exe" "D:\ky8\prak\NLP moduls\main.py"  
topic modeling testing  
[('Topic 0', ['language', 'word', 'natural', 'information', 'processing', 'text', 'model', 'et', 'data', 'machine']),
```

Рисунок 10 – Результат выполнения программы

4 Модель косинусного сходства

4.1 Косинусное сходство

Косинусное сходство – это мера сходства между двумя векторами предгильбертового пространства, которая используется для измерения косинуса угла между ними.

Если даны два вектора признаков, A и B , то косинусное сходство, $\cos(\theta)$, может быть представлено как формула (1):

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^n A_i \cdot B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \cdot \sqrt{\sum_{i=1}^n (B_i)^2}}. \quad (1)$$

В случае информационного поиска, косинусное сходство двух документов изменяется в диапазоне от 0 до 1, поскольку частота термина (веса $tf-idf$) не может быть отрицательной. Угол между двумя векторами частоты термина не может быть больше, чем 90° .

Одна из причин популярности косинусного сходства состоит в том, что оно эффективно в качестве оценочной меры, особенно для разреженных векторов, так как необходимо учитывать только ненулевые измерения.

Помимо косинусного подобия, в NLP также можно применять мягкую косинусную меру.

«Мягкая» косинусная мера - это «мягкая» мера сходства между двумя векторами, то есть мера, которая учитывает сходства между парами признаков. Традиционное косинусное сходство рассматривает признаки векторной модели как независимые или полностью обособленные, тогда как «мягкая» косинусная мера учитывает сходства признаков в векторной модели. Это позволяет обобщить идею косинусной меры, а также идею сходства объектов в векторном пространстве («мягкое» сходство).

Например, в области обработки естественного языка сходство между объектами весьма интуитивно. Например, слова «играть» и «игра» различны и,

таким образом, отображаются в различных измерениях в векторной модели, хотя, очевидно, что они связаны семантически.

Для расчета «мягкой» косинусной меры вводится матрица s сходства между признаками. Она может рассчитываться, используя расстояние Левенштейна. Затем производится умножение с применением данной матрицы. Если даны два N -мерных вектора a и b , то мягкая косинусная мера рассчитывается следующим формулой (2):

$$soft_cosine(a, b) = \frac{\sum_{ij}^N s_{ij} a_i b_j}{\sqrt{\sum_{ij}^N s_{ij} a_i a_j} \sqrt{\sum_{ij}^N s_{ij} b_i b_j}}, \quad (2)$$

где s_{ij} = сходство (признак $_i$, признак $_j$).

При отсутствии сходства между признаками ($s_{ii} = 1, s_{ij} = 0$ для $i \neq j$), данное уравнение эквивалентно общепринятой формуле косинусного сходства. Степень сложности этой меры является квадратичной, что делает её вполне применимой к задачам реального мира. Степень сложности может быть также трансформирована в линейную.

4.2 Встраивание слов

Чтобы вычислить значение косинуса между двумя документами, необходимо преобразовать их в векторную форму. Методами преобразования текста в вектор являются встраивание слов. В данном случае применяется метод word2vec.

Преобразование текста в вектор здесь означает использование семантического значения слова для представления этого слова. Чем более похожи слова семантически, тем ближе они друг к другу по значению. Таким образом, текст можно записать как последовательность чисел. Например: возьмем 4 слова: King (король), Queen (королева), Man (мужчина), Woman (женщина). Видно, что слово King схоже по значению с Man и противоположно Queen, а слово Queen схоже с Woman и противоположно King. Используя семантику приведенных выше слов для представления их в векторном

пространстве, слово «король» будет близко к слову «мужчина» и далеко от «королева и женщина», это также верно для королевы и женщины. Таким образом, приведенные выше 4 слова в векторном пространстве имеют форму, показанную на рисунке 11.

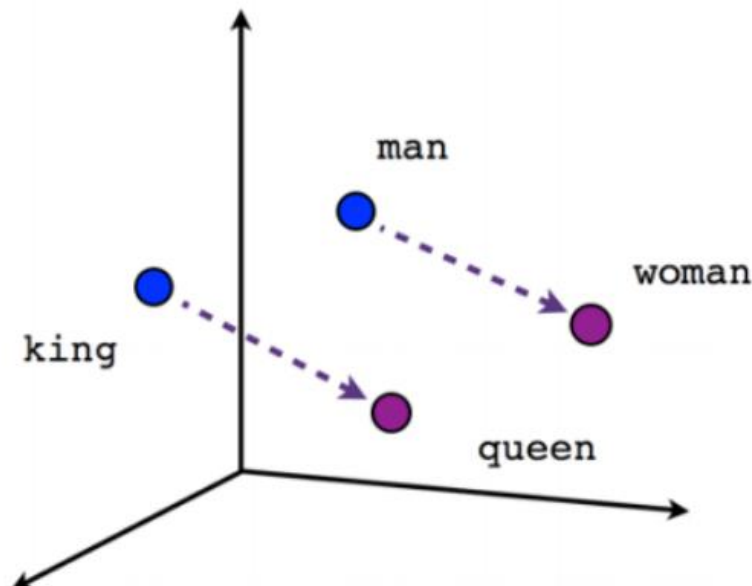


Рисунок 11 – Представление слова в векторном пространстве

Существует два основных метода построения модели word2vec – это непрерывный набор слов (CBOW) и Skip Gram:

- модель CBOW прогнозирует текущее слово с учетом слов контекста в определенном окне. Входной слой содержит слова контекста, а выходной слой содержит текущее слово. Скрытый слой содержит измерения, которые мы хотим представить текущему слову, присутствующему в выходном слое;

- Skip Gram прогнозирует окружающие слова контекста в конкретном окне с учетом текущего слова. Входной слой содержит текущее слово, а выходной слой содержит контекстные слова. Скрытый слой содержит количество измерений, в которых мы хотим представить текущее слово, присутствующее во входном слое.

Два вышеуказанных метода показаны на рисунках 12 и 13.

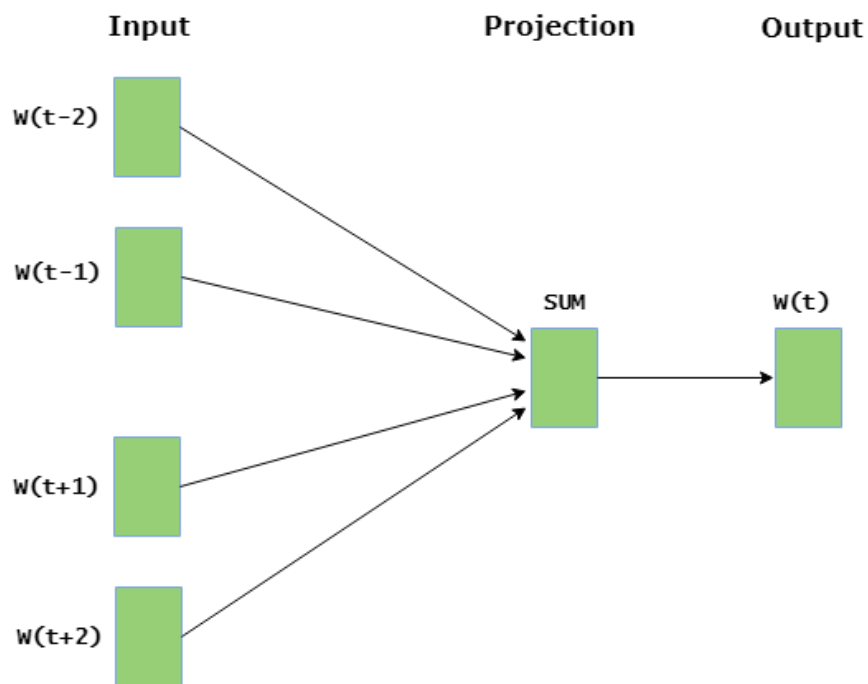


Рисунок 12 – Модель CBOW

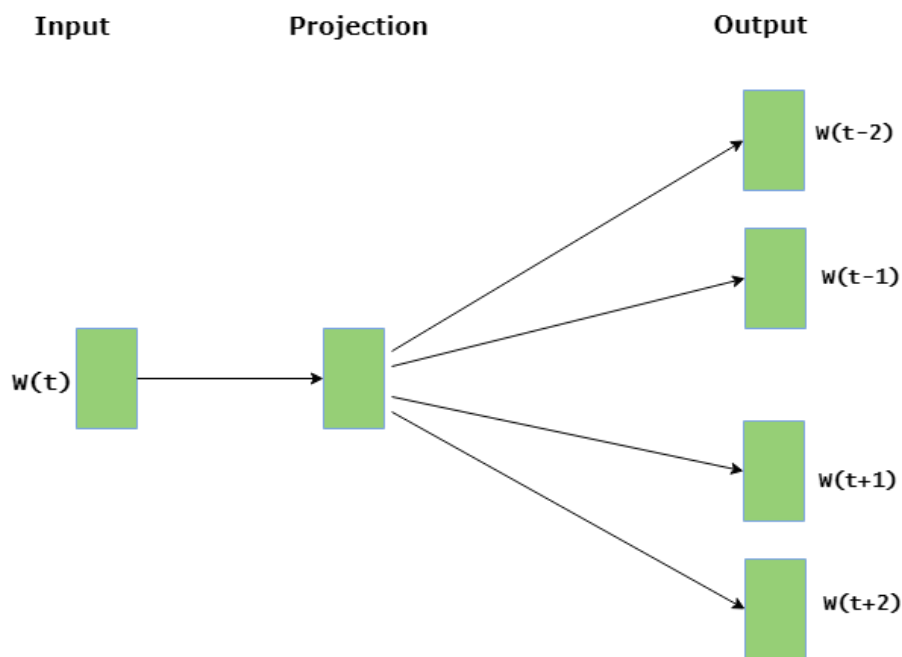


Рисунок 13 – Модель Skip Gram

4.3 Проектирование модели на языке Python

Базовая модель строится на языке Python в основном с использованием функций библиотеки `gensim`, как показано на рисунке 14. Модель получает два файла и возвращает результат их семантического сходства (0~1).

```

def cosine_sim_model (file1, file2):
    preprocessed_text = []
    preprocessed_text.append(preprocess(file1))
    preprocessed_text.append(preprocess(file2))
    dictionary = corpora.Dictionary(preprocessed_text)
    corpus1 = dictionary.doc2bow(preprocessed_text[0])
    corpus2 = dictionary.doc2bow(preprocessed_text[1])
    word2vect = api.load("glove-wiki-gigaword-50")
    #building similarity matrix
    similarity_index = WordEmbeddingSimilarityIndex(word2vect)
    similarity_matrix = SparseTermSimilarityMatrix(similarity_index, dictionary)
    #calculating similarity
    similarity = similarity_matrix.inner_product(corpus1, corpus2, normalized=(True, True))
    return similarity

file = open("NLP.pdf", "rb")
file2 = open("NLP_review.pdf", "rb")
file3 = open("medicine.pdf", "rb")
print("cosine similarity testing")
similarity = cosine_sim_model(file, file2)
print("similarity between file1 and file2:")
print(similarity)
similarity = cosine_sim_model(file3, file2)
print("similarity between file2 and file3:")
print(similarity)

```

Рисунок 14 – Модель для вычисления косинусного сходства

Текст в двух файлах после обработки используется для создания словаря, включающего все буквы обоих файлов и два соответствующих корпуса для каждого файла.

В переменную word2vec загружается модель, преобразующая слова в векторы. Модель здесь предварительно обучена на большом объеме текста и имеет высокую точность определения семантического значения слов. Переменная word2vec отвечает за расчет семантических значений слов в словаре, составленном из текста двух файлов.

Затем на основе значений, полученных из word2vec и словаря, строится матрица сходства. Это матрица пар значений между словами в словаре, другими словами, это векторное пространство всех слов, содержащихся в двух файлах.

Текст представляется в векторной форме собственным корпусом в векторном пространстве, образованном матрицей сходства. Функция

Internal_product – это скалярное произведение двух векторов, это уровень семантического сходства между двумя текстами с величиной от 0 до 1, где 0 – совершенно разные, а 1 – полностью идентичные.

На рисунке 14 модель запускается дважды с тремя разными файлами, два из них, file и file2 – это два файла с содержимым, касающимся обработки естественного языка, а оставшийся файл, file3, содержит содержимое, посвященное медицине. Результаты работы программы показаны на рисунке 15, видно, что первый набор из 2 файлов имеет очень высокий уровень сходства, тогда как сходство между файлом 1 и файлом 3 достаточно низкое, это совпадает с прогнозом. о содержимом трех файлов, описанных выше.

```
cosine similarity testing
100%|██████████| 2699/2699 [00:21<00:00, 127.64it/s]
similarity between file1 and file2:
0.9096424
100%|██████████| 1181/1181 [00:09<00:00, 126.14it/s]
similarity between file2 and file3:
0.43450445
```

Рисунок 15 – Результат выполнения программы

ЗАКЛЮЧЕНИЕ

В процессе прохождения преддипломной практики были проектированы модели для выполнения основных функций при обработке естественного языка в тексте, к этим функциям относятся: извлечение данных из pdf-файлов, очистка текста, поиск тематик текстов, вычисление семантической эквивалентности между текстами.

На основе разработанных моделей можно добавлять новые функции и улучшать существующие для лучшего соответствия конечному продукту.

СПИСОК ИСПОЛЬЗУЕМОХ ИСТОЧНИКОВ

1. Тематическое моделирование [Электронный ресурс] - 2024 URL: https://ru.wikipedia.org/wiki/Тематическое_моделирование (дата обращения 13.02.2024).
2. Латентное размещение Дирихле [Электронный ресурс] - 2024 URL: https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation (дата обращения 13.02.2024).
3. руководство по библиотеке Gensim [Электронный ресурс] - 2024 URL: <https://www.geeksforgeeks.org/nlp-gensim-tutorial-complete-guide-for-beginners/> (дата обращения 13.02.2024).
4. руководство по вычислению косинусного сходства с использованием Gensim [Электронный ресурс] - 2024 URL: https://github.com/piskvorky/gensim/blob/develop/docs/notebooks/soft_cosine_tutorial.ipynb (дата обращения 13.02.2024).