Софийски университет „Климент Охридски“
Факултет по математика и информатика

ПРОЕКТ
ПО ИЗКУСТВЕН ИНТЕЛЕКТ

летен семестър 2024/2025

СИСТЕМА ЗА АВТОМАТИЗИРАНО ТЕМАТИЧНО ГРУПИРАНЕ И ОБОБЩАВАНЕ НА ТЕКСТОВИ ДОКУМЕНТИ

Стоян Стоянов Иванов
9MI0400132
Информатика, 3 курс

Ръководител на курса:
проф. д-р Мария Нишева-Павлова

май 2025 г.

СЪДЪРЖАНИЕ

1. Формулировка на задачата

Настоящият проект представлява софтуерна система за автоматизирано тематично групиране и обобщаване на текстови документи, използваща корпуса 20Newsgroups – една от най-широко използваните колекции в областта на Обработката на естествен език (*NLP*[1]). Колекцията включва приблизително 20 000 новинарски съобщения, категоризирани в 20 различни тематични групи, и е особено подходяща за задачи като класификация, *клъстериране*[2] и извличане на информация.

Целта на системата е да улесни анализа на големи обеми неструктурирана текстова информация, като автоматично групира сходни документи и генерира представителни обобщения за всяка открита тематична група. Тази функционалност е особено полезна при предварителен преглед на големи архиви от текст.

За реализирането на такава система се налага комбиниране на три основни компонента: надеждно извличане и структуриране на текстови данни, ефективно тематично клъстериране и резюмиране чрез техники от *NLP* и интуитивен потребителски интерфейс за управление на процеса и визуализиране на резултатите.

Тези изисквания предопределят избора на хибридна архитектура, която комбинира две силно утвърдени технологии – *Java*[3], за разработка на настолни приложения и мащабируема архитектура, и *Python*[4], използвайки библиотеки за обработка на естествен език и машинно обучение – в случая, *scikit-learn*[5].

2. Използвани алгоритми

Описание на използваните структури от данни и съответните класове, имплементирани в системата за обработка на текстове. Фокусът е върху два основни компонента: резюмиране на текстове и клъстериране на документи.

---

[1] ***NLP*** *(Natural Language Processing) или Обработка на естествен език е област от изкуствения интелект, която се занимава с взаимодействието между компютрите и човешкия език. Целта ѝ е да даде възможност на компютрите да разбират, интерпретират, генерират и реагират на текст и говор на начин, който е смислен за хората*

[2] ***Клъстериране*** *означава групиране на текстове според тематичната им близост, като по този начин се извличат основните категории или теми, около които се организира съдържанието.*

[3] ***Java*** *е обектно-ориентиран, програмен език, разработен от Oracle. Java се използва широко за разработка на уеб, мобилни, сървърни и настолни приложения. Отличава се със стабилна и мащабируема архитектура, силна типизация и богата стандартна библиотека.*

[4] ***Python*** *е интерпретативен програмен език, известен със своята простота, четимост и лаконичен синтаксис. Отличава се с динамична типизация, автоматично управление на паметта, голям набор от библиотеки, включително за машинно обучение, силна общност и лесна интеграция с други езици.*

[5] ***scikit-learn*** *е популярна библиотека в Python за машинно обучение.*

2.1. Резюмиране на текстови документи

Класът *TwentyNewsGroupsSummarizer* имплементира интерфейса *AbstractSummarizer* и реализира ***екстрактивно резюмиране***[6], чрез избор на най-значимите изречения от оригиналния текст.

2.1.1 Основни стъпки

1. *Токенизация*[7] на изречения и думи.
2. Филтриране на *стоп-думи*[8].
3. Изчисляване на честота на думите.
4. Извличане на ключови изречения на база на важни думи.
5. Форматиране на обобщението за изход.

2.1.2. Алгоритъм

1. Създава се речник на думите с техните честоти. Повторна поява на дума увеличава стойността с 2, което подсилва нейното значение.
2. Премахват се всички стоп-думи от речника.
3. Думите се сортират по честота в низходящ ред.
4. Извличат се изречения, които съдържат най-честите важни думи. Винаги се избира първото изречение, тъй като често съдържа заглавието.
5. Извличането спира при достигане на зададен лимит.
6. Избраните изречения се форматират с табулации и нови редове на всеки 15 думи.

2.1.3. Пример за използване

```
String input = "...";
AbstractSummarizer summarizer = new TwentyNewsGroupsSummarizer();
String summary = summarizer.summarize(input, 5);
System.out.println(summary);
```

---

[6] ***Екстрактивно резюмиране*** *е техника за автоматично обобщаване на текст, при която се извличат директно най-важните изречения или фрази от оригиналния текст, без да се променя тяхната форма.*

[7] ***Токенизация*** *е процесът на разделяне на текст на токени.*

[8] ***Стоп-думи*** *са често срещани, но слабо информативни думи (напр. a, the, is, and), които се премахват от текста, за да се повиши значимостта на останалите думи.*

2.2. Клъстериране на документи

Класът *KMeansClusterer* реализира алгоритъма k-средни (*K-Means*[9]) за клъстериране на документи, базиран на евклидова дистанция между векторни представяния, в случая - *TF-IDF*[10].

2.2.1 Компоненти

CentroidInitializer отговаря за началната инициализация на *центроидите*[11]. Входът е матрица от n реда и d колони, всеки ред е *вектор*[12] на документ, брой клъстери и генератор на случайни числа. Изходът – масив от k вектора, представляващи началните центроиди.

Алгоритъм:

1. За всяко i от 0 до k-1: Избира случаен ред от входната матрица. Използва го като центроид i.

2. Връща масив от k центроиди.

Наивна, но бърза и ефективна откъм време за изпълнение и тестване инициализация.

KMeansClusterAssigner е компонент за присвояване на всяка точка към най-близкия центроид. Приема матрица, всеки ред е документ/вектор, и текущите центроиди, и връща масив от етикети, указващи клъстера на всеки документ.

Алгоритъм:

За всяка точка от данните т.е. ред на матрицата:

1. Преобразувай реда в вектор.

2. Изчисли Евклидовото разстояние до всеки центроид.

$$\text{distance}(\vec{x}, \vec{c}) = \sqrt{\sum_i (x_i - c_i)^2}$$

3. Запази индекса на центроида с най-малко разстояние т.е. най-близкия.

---

[9] ***K-Means*** *е популярен алгоритъм за клъстериране, който разделя набор от обекти в k на брой групи (клъстери) въз основа на сходството между тях.*

[10] ***TF-IDF*** *е статистическа мярка, използвана за оценка на важността на дума в даден документ спрямо целия корпус от документи.*

- *TF (Term Frequency): Колко често се среща дума в конкретен документ.*
- *IDF (Inverse Document Frequency): Колко рядко се среща думата в цялата колекция от документи.*

$$TFIDF(t,d) = TF(t,d) x log\left(\frac{N}{DF(t)}\right)$$

където: t – дума, d – документ, N – брой документи, DF(t) – брой документи, съдържащи думата t.

[11] ***Центроид*** *в контекста на клъстерирането е вектор, който представлява средната стойност на всички точки/документи в даден клъстър. Той служи като представителен център на групата и често се използва за измерване на сходство между документи и клъстъри.*

[12] ***Вектор*** *в NLP представлява числово представяне на текст. Всеки текстов документ може да бъде преобразуван във вектор от характеристики (напр. TF-IDF стойности), което позволява машинна обработка и сравнение между текстове.*

4. Присвои на текущата точка този индекс като неин клъстер.

CentroidUpdater изчислява новите центроиди след всяка итерация. Входът е матрица с вектори, масив с индекси на клъстерите, брой клъстери и генератор за избор на стойност при празен клъстер, а изходът е масив от нови центроиди.

Алгоритъм:

1. Инициализира k нулеви вектора.

2. За всеки ред: добавя го към съответния клъстер и увеличава броя му.

3. За всеки клъстер: ако съдържа точки: изчислява средна стойност, иначе – избира случаен документ за нов центроид.

Например, документ за „космос“ ще бъде в същия клъстер като документи за „астрономия“, а центроидът ще представя идеята „наука за космоса“.

Функционалност

Методът *cluster()* приема като аргументи матрица с n реда и d колони, където всеки ред представлява d-мерен вектор и брой желани *клъстери*[13]. Връщана стойност е масив от етикети с дължина n, където i-тият индекс е индексът на клъстера, към който принадлежи векторът на ред i.

Алгоритъм:

1. Инициализиране на k случайни центроида.

2. Повтаряй до сближаване или достигане на *максимален брой итерации за сближаване*[14]:

2.1. Присвояване: всяка точка се присвоява към най-близкия центроид.

2.2. Обновяване: новите центроиди се преизчисляват като средно аритметично на точките в клъстера.

3. Описание на програмната реализация

3.1. Контролер

Този модул предоставя графично настолно приложение, което позволява на крайния потребител да използва системата за автоматично обобщаване на текстове чрез групиране на документи. Приложението визуализира прогреса, приема входни параметри от потребителя, и комуникира със слоя-услуга, който изпълнява основните

[13] ***Клъстер*** *е група от обекти (напр. документи, изречения или думи), които са сходни помежду си според определен критерий за подобие, и едновременно с това се различават от обектите в други групи.*

[14] ***Сближаване*** *е състояние, при което клъстерите вече не се променят.*

NLP алгоритми. Този модул е изграден върху *Java Swing*[15] и следва архитектурата *MVC*[16].

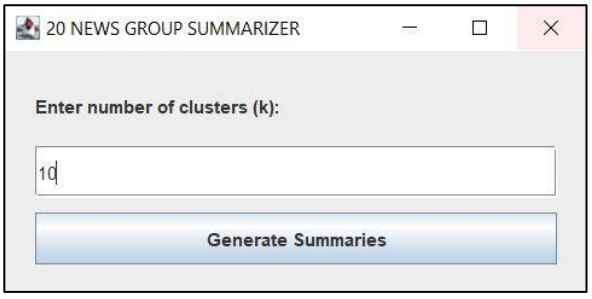
3.1.1. Компоненти

DesktopApplication е основната входна точка на приложението. Стартира *Python* скрипт, който подготвя набора от документи, инициализира обект за достъп до документи, създава имплементация на сумиращия алгоритъм и стартира графичния потребителски интерфейс.

DesktopUI отговаря за изграждането и управлението на графичния интерфейс на приложението. Валидира и обработва входа – броят клъстери. Използва *callback* интерфейс, за да визуализира напредъка от *background* процесите[17], без да блокира потребителският интерфейс.

PythonScriptRunner изпълнява външен *Python* скрипт от *Java* среда. Скриптът *export-twenty-news-groups* е отговорен за свалянето и *сериализацията*[18] на документите от *20Newsgroups* като итерира през всички документи, извлича тематичните категории, създаване поддиректория за всяка една категория и записване документа като *.txt* файл.

Ролята на *ProgressCallback* е да декларира обратни методи, чрез които сървисният слой да информира UI-а за напредък, успешно завършване и/или възникване на грешка.



[15] ***Java Swing*** *е графична библиотека за изграждане на настолни потребителски интерфейси в Java.*

[16] ***MVC (Model-View-Controller)*** *е архитектурен шаблон за структуриране на софтуерни приложения чрез разделяне на отговорностите между три основни компонента: Model – съдържа логиката за работа с данните и бизнес логиката на приложението. Той отговаря за съхранение, модификация и достъп до информация. View – представя потребителския интерфейс и визуализира данните от модела. Той е отговорен за взаимодействие с потребителя. Controller – посредничи между View и Model. Обработва входа от потребителя, взима решения и извиква съответните методи от модела.*

[17] ***Background process*** *е задача, която се изпълнява във фонов режим – без да блокира основния потребителски интерфейс или изпълнението на главната програма.*

[18] ***Сериализация*** *е процесът на преобразуване на обект в поток от байтове, който може да бъде съхранен (напр. във файл) или предаден (по мрежа), и впоследствие възстановен чрез обратния процес десериализация.*

3.2. Услуга

Този модул отговаря за реализацията на слой услуги, който изпълнява задачата по автоматично обобщение на текстови документи чрез прилагане на техники от обработка на естествен език и машинно обучение.

Основната цел на този модул е да прочете и почисти документи, след това да ги преобразува в числов вид чрез *TF-IDF* векторизация, да ги групира по сходство чрез *K-Means* клъстеризация и накрая да извлече и съхрани обобщенията на всеки клъстер с помощта на специален *сумаризатор*[19].

3.2.1. Основни компоненти

DocumentSummarizerService е *абстракция*[20], която дефинира един основен метод, а именно генерирането на обобщение. Методът приема брой клъстери т.е. групи от подобни документи, и *callback*[21] – механизъм за проследяване на напредъка, отчитане кога е приключил алгоритъмът и прихващане на грешки.

DocumentSummarizerServiceImpl имплементира процеса на обобщаване в няколко стъпки:

1. Зарежда документите от хранилището.
2. Изчиства документи от шум, специални символи и безсмислено съдържание.
3. Разделя текстовете на списъци от думи, използвайки *Apache Lucene*[22].
4. Превръща *токените*[23] в числова матрица с претегляне на думите по *TF-IDF.*
5. Групира документите в k клъстера с помощта на *K-Means* алгоритъма.
6. Взема представителен документ от всеки клъстер, създава резюме и го записва.

3.2.2. Спомагателни компоненти

Целта на *DocumentPreprocessor* е премахване на шум, невалидни символи и спам, както и филтриране на неинформативни документи, които са прекалено кратки

---

[19] ***Сумаризатор*** *е компонент, който извлича съкратено представяне на съдържанието на текстов документ, запазвайки неговите ключови идеи. Може да бъде екстрактивен (избира важни изречения) или абстрактивен (генерира нов текст).*

[20] ***Абстракцията*** *е принцип от обектно-ориентираното програмиране, чрез който се скрива сложната вътрешна логика на даден компонент и се предоставя опростен интерфейс за използване. Това позволява изграждането на модулни, четими и поддържани системи.*

[21] ***Callback интерфейс*** *е програмна конструкция, при която даден обект (или метод) предоставя интерфейс, чрез който друга част от програмата може да го "уведомява" за събития, състояния или резултати.*

[22] ***Apache Lucene*** *е високо производителна и мащабируема библиотека за пълнотекстово търсене, написана на Java. Тя предлага механизми за индексиране, анализ на текст, търсене по релевантност, ранжиране и други ключови функции, необходими за изграждането на търсещи системи.*

[23] ***Токен*** *е минимална смислова единица в текст, основа за по-нататъшна обработка като анализ на честота, граматика или смисъл.*

или „мусорни“ – имат твърде малко букви или твърде много главни букви – спам, код, и бинарни файлове често съдържат такива модели. Използва регулярни изрази за премахване на *ASCII* шум, бинарни низове, *хедъри*[24] от части на файлове, линкове, съмнителни формати, множество специални символи, повтарящи се знаци, *URL*-та, и др.

DocumentTokenizer преобразува всеки документ в списък от думи, използвайки *Apache Lucene*. Основният метод премахва пунктуация, нормализира думите и премахва стоп-думите.

TFIDFVectorizer преобразува токенизираните документи в *TF-IDF* матрица, която улавя значимостта на думите. Събират се всички думи и техните честоти, след това се избира определен брой думи според честота, създава се речник, брои се всяка дума във всеки документ, и накрая се изчисляват *TF-IDF* стойности.

ClusterSummarizer създава резюме за всеки клъстер от подобни документи. Извлича всички документи, принадлежащи на даден клъстер. Избира представителен(и) документ(и). Подлага документа на сумирация и съхранява обобщението.

3.3. Хранилище

Следният модул предоставя абстракция и реализация за четене и запис на текстови документи и техните обобщения. Това е важна част от всеки *NLP pipeline*, който включва зареждане на сурови текстове, клъстеризация по теми и генериране на автоматични резюмета.

Тази архитектура позволява ясно разделение между обработка на данни и физическия достъп до файловата система, като по този начин повишава гъвкавостта, тестването и повторната употреба на кода.

3.3.1. Основни компоненти

DocumentRepository е абстракция, която дефинира два основни метода: зареждане всички текстови документи от източника и записване на обобщен текст във файл, организиран по клъстер и индекс.

По този начин се позволява да се използват различни хранилища – например файлова система, база данни, облак, без промяна в основната логика за *NLP* обработка.

---

[24] ***Хедър**, в контекста на текстови документи и електронна кореспонденция, е метаинформация, която предхожда основното съдържание на документа – напр. дата, подател, тема, ID и т.н.*

Реализацията на горния интерфейс – *DocumentRepositoryImpl* използва *DocumentReaderWriter* за четене/запис на файлове.

Този клас на свой ред изолира логиката за четенето на файлове от директория с филтриране на дефектни документи и писането на нови файлове, включително създаване на директории при нужда. Важно е да се отбележи, че *ReaderWriter* игнорира файлове с прекалено много контролни символи - дефектни или бинарни.

3.4. Структури от данни

В този модул се намират класовете *AbstractSummarizer, TwentyNewsGroupsSummarizer, KMeansClusterer, KMeansClustererAssigner, CentroidUpdater* и *CentroidInitializer*. Описанието на функционалностите за всеки от изброените горе класове се намира в

3.5. Бъдещи подобрения

Въпреки своята изчислителна ефективност и интуитивност, *K-Means* има съществени ограничения при приложението си в *NLP*. За по-високо качество на клъстериране и обобщаване на текстове се препоръчва:

1. Използване на косинусова метрика вместо Евклидова.
2. Използване на *k-means++*, който избира началните центроиди по-информативно и намалява риска от лоша инициализация и попадане в локален минимум.
3. Алгоритъмът изчислява средно аритметично на точките в клъстера, което е неподходящо при разредени вектори, каквито са често срещани при *TF-IDF*.
4. Изисква предварително задаване на броя на клъстерите. Може да се използват евристични методи като: *elbow* метод – за определяне на оптималното k чрез визуализация на инерцията, или *silhouette* анализ – за оценка на качеството на групиране.
5. Въпреки че е полезен за тематично групиране, алгоритъмът не отчита граматическа структура, контекст, или семантична значимост на думите.

4. Примери

Потребителят стартира приложението от десктопа – с двойно кликване на *.jar* файла, или за целите на демонстрацията през интерфейса на *IntelliJ IDEA 2024.3*. Програмата автоматично стартира *Python* скрипта *export-twenty-news-groups.py*, който изтегля и обработва корпуса *20Newsgroups*, съхранява документите в локална

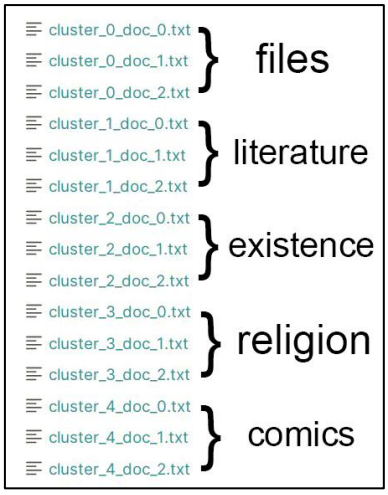
директория *20newsgroups/**, след това *Java* приложението зарежда документите, преобразува документите, извършва векторизация, клъстерира документите с *K-Means* и накрая обобщава всеки клъстър т.е. тема. Резултатът се запазва в локалната директория *summaries/*.*

4.1. Грубо обобщение (k = 5)

Алгоритъмът създава само 5 теми. Всеки клъстър съдържа голям брой документи от потенциално няколко подобни подтеми. Обобщенията са широки и обобщаващи, често комбинират различни аспекти на една тема.

Един клъстър може да обединява теми като: *sci.space*, *sci.electronics*, *sci.med*, защото всички са научни. Обобщението ще съдържа общи изречения за „наука“, без да отличава конкретна дисциплина.

Подходящо за е бърз първоначален анализ и намиране на основни тематични категории.



4.2. Баланс между специфика и обобщеност (k = 10)

По-фино тематично разделение. Клъстерите вече разделят отделни тематични групи, но някои все още може да съдържат смесица.

Примерно първият клъстер ще е *comp.graphics* и *comp.os.ms-windows.misc*, вторият клъстер ще е *rec.sport.baseball* и *rec.sport.hockey*, а третият би бил *talk.politics.mideast* и *talk.politics.misc.*

Подходящо за е умерено детайлен анализ и случаите, където точността е важна, но не прекалено разпиляна.

Accounts of Anti-Armenian Human Rights Violations in Azerbaijan #007 Prelude to Current Events in Nagorno-Karabakh They grab Papa, carry him into one room, and Mamma and me into another.

They start tearing my clothes, right there, in front of Mamma.

I don't remember where they went, what they did, or how much time passed.

I had the feeling that they beat me on the head, on my body, and tore my clothes, all at the same time, I don't even know what I said.

The atrocities started.

They were showing an Argentinian film, "The Abyss." Before the film we noticed about 60 to 70 people standing near the podium at the City Party Committee, but they were silent, there's no conversation whatsoever, and we couldn't figure out what was going on.

We bought our tickets.

There were 30 or 40 people in the theater.

They came up onto the stage.

So that if someone recognized me or if something happened, they would take me, and not Marina.

It seemed to me that those people were not themselves.

Then it was all over, as though nothing had happened at all.

Incidentally, when we came out of the theater we saw police, policemen standing there.

He stood there grief- stricken, but looking as though nothing really big had happened, like some naughty boys had just broken them quite by accident, with a slingshot.

I said, well, OK .

On the 28th everything was like it was supposed to be, we lived like we always had.

Then a girlfriend of mine, Lyuda Zimogliad, came by at around three o'clock I think.

She said that something awful was happening in town.

I go out on the balcony, but I can't see what's going on, because the noise is coming from the direction of the bus station, and there is a story building in the way.

They took his club away from him and started to beat him with it.

They start tearing my clothes, right there, in front of Mamma.

I don't remember where they went, what they did, or how much time passed.

I had the feeling that they beat me on the head, on my body, and tore my clothes, all at the same time, I don't even know what I said.

The atrocities started.

4.3. Висока детайлност (k > 100)

Клъстерите вече разделят отделни тематични групи, но някои все още може да съдържат смесица. Един клъстър може да съдържа документи с много близка терминология, дори и да са от една и съща група в корпуса. Много теми ще се припокриват. Обобщенията ще бъдат близки до съдържанието на отделни документи или идентични с едно изречение, и нямат да имат стойност като „тематично обобщение“.

Алгоритъмът губи смисъла на клъстериране – става класификация по уникалност. Подходящо е за диагностика на шума и тестване на стабилността на алгоритъма.

Бележка

Важно е да се отбележи, че в класът *ClusterSummarizer* може да се променят променливите *maxSummarySentences* (максималко колко изречения да има в документ от тема) и *minClustersByDocs* (минимално колко документа да се създадат за клъстер).

5. Заключение

Настоящият проект представя система за клъстериране и автоматично обобщаване на текстове от корпуса *20Newsgroups*, съчетавайки силните страни на *Java* и *Python*. *Java* осигурява стабилна архитектура и графичен потребителски интерфейс, докато *Python* предоставя мощни инструменти за обработка на естествен език и машинно обучение.

Използван е алгоритъмът *K-Means*, комбиниран с *TF-IDF* векторизация, който позволява тематично групиране на документи. Подходът е гъвкав, мащабируем и подходящ за различни видове текстови корпуси и позволява лесна интеграция с модулите за потребителски интерфейс и асинхронна обработка, което подобрява потребителското изживяване.

Системата е приложима както за научни изследвания, така и за реални потребители, нуждаещи се от бърз преглед на големи текстови масиви. Със своята модулна архитектура и асинхронна обработка, проектът осигурява добра основа за разширение и интегриране на по-усъвършенствани *NLP* технологии.

6. Литература

- https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f [2025-05-24]
- https://machinelearningmastery.com/gentle-introduction-text-summarization/ [2025-05-24]
- https://github.com/karimo94/Text-Summarizer/blob/master/Summarizer.java [2025-05-25]
- https://www.youtube.com/watch?v=1hGk9PUcOv4 [2025-05-30]
- https://medium.com/@ryver.dev/building-a-simple-ai-powered-text-summarizer-with-transformers-in-python-0a31c848e1d2 [2025-05-31]