https://doi.org/10.53656/math2025-5-4-idl

Educational Technologies
Образователни технологии

ИНФРАСТРУКТУРА ЗА СЪЗДАВАНЕ НА ЧАТБОТ С ГОЛЕМИ ЕЗИКОВИ МОДЕЛИ И КОНТЕКСТНО РАЗШИРЯВАНЕ НА ИНСТРУКЦИИТЕ

Йордан Кралев[1,2], Светла Коева[2]

1)Факултет „Автоматика“, катедра „Системи и управление“, Технически университет – София

2)Институт за български език „Проф. Любомир Андрейчин“, Българска академия на науките

Резюме. В статията се представя инфраструктура, която интегрира свободно достъпни инструменти и големи езикови модели в персонализиран чатбот. Целта е да се демонстрира как големите езикови модели могат да бъдат използвани с помощта на достъпно като цена GPU, както и възможностите за разширяване на потребителските заявки с допълнителен контекст в зависимост от целите и областта на приложение. Работата на предварително обучени големи езикови модели се демонстрира с помощта на технологията RAG (Retrieval-Augmented Generation, или извличане с разширено генериране) за български за решаване на конкретна задача в дадена тематична област. Избраната задача е резюмиране на документи, а областта е наука и образование. Представеното решение може да се приложи за създаването на други приложения за български език.

Ключови думи: големи езикови модели; изкуствен интелект; чатботове

1. Въведение

Големите езикови модели (LLM) са технология на изкуствения интелект, която „разбира“, обобщава, генерира и прогнозира ново съдържание, използвайки техники за дълбоко обучение върху масивни набори от данни. Създаването на собствен голям езиков модел все още е предизвикателство (а вероятно и ще остане поне известно време). Добре известно е, че предварителното обучение на голям езиков модел

изисква голямо количество данни. Например през 2023 г. LLaMA използва набор от данни за обучение с обем 4,6 терабайта (Goyal et al. 2023). Може да се използва API (приложно-програмен интерфейс) на големи езикови модели, които се предлагат в момента. Неудобствата са свързани с абонамента, изискването за непрекъсната интернет връзка, възможните ограничения за персонализация, контрола над модела, но най-вече се отнасят до сигурността на данните, които се използват при комуникацията с модела.

Работата на предварително обучени големи езикови модели ще бъде демонстрирана с технологията RAG (Retrieval-Augmented Generation, или извличане с разширено генериране) за български за решаване на конкретна задача в дадена тематична област. Избраната задача е резюмиране на документи, а областта е наука и образование. Предлаганата инфраструктура комбинира свободно достъпни инструменти и големи езикови модели и илюстрира как инструментите могат да бъдат комбинирани както за използване на моделите локално само с CPU или с достъпно като цена GPU, така и за разширяване на потребителските заявки с допълнителен контекст в зависимост от целите, за които се използват големите езикови модели. Потребителите могат да конфигурират заявките в зависимост от предпочитанията си за използването на големите езикови модели за конкретни задачи: опростяване на текст (Fang et al. 2025), отговаряне на въпроси (Xu et al. 2025), класификация на документи (Sun et al. 2023) и др., и в конкретни области: финанси (Murtaza et al. 2025), математика (Henkel et al. 2024) и др. По този начин резултатите от работата на големите езикови модели могат да бъдат фокусирани и ще позволяват по-адекватно и по-бързо вземане на решения.

2. Кратко представяне на използваните технологии

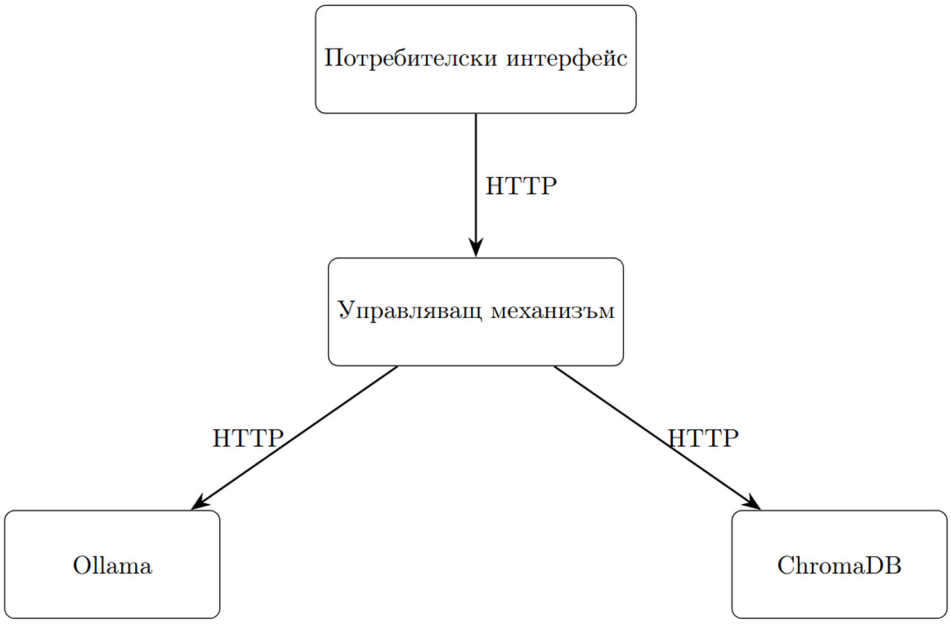
След въвеждането на архитектурата Transformer (Vaswani et al. 2017) е постигнат сериозен напредък в широк набор от задачи в областта на изкуствения интелект и компютърната обработка на езика. Напредъкът се основава на факта, че моделите, създадени с тази архитектура, са лесни за паралелизиране, мащабиране и фина настройка или адаптиране към широк кръг от задачи. Мащабирането

се прилага както по отношение на размера на моделите, така и на размера на данните за обучение (Radford et al. 2018; Kaplan et al. 2020). Днес, няколко години след въвеждането на Transformer, хардуерът и инфраструктурата за обучение на големи езикови модели са по-широко достъпни и е налице много по-добро и по-задълбочено разбиране докъде може да се мащабират моделите и колко данни са необходими за предварително обучение (Hoffmann et al. 2022). По този начин вниманието се измества от просто мащабиране на размерите на модела към фина настройка на съществуващите модели (Ouyang et al. 2022). BERT (Bidirectional Encoder Representations from Transformers) е модел за дълбоко обучение, който използва архитектура Transformer, и по-точно частта на енкодера (Devlin et al. 2019). BERT се обучава чрез произволно маскиране на думи в изреченията и предсказването им въз основа на околния контекст и прилага двупосочна обработка на контекста. По тази причина редица изследвания показват, че моделът BERT и последващите модификации се справят добре с откриването на логически връзки. Обратно, моделите само с декодер, като GPT (Generative Pre-Trained Transformer) или LLaMA, генерират изход авторегресивно – предсказвайки най-вероятната следваща дума въз основа на предходния контекст. Чрез премахване на енкодера GPT моделите (Achiam et al. 2023) могат да обработват входните данни директно и да генерират изход по-бързо. Тази архитектура позволява моделите да бъдат обучавани върху голямо количество неанотирани данни. В момента сме свидетели на непрекъснатата поява на нови езикови модели, част от които са свободно достъпни, а друга част – компресирани по начин, така че могат да бъдат използвани с широко разпространен хардуер.

Технологията *генериране с разширено извличане* (Retrieval-augmented Generation – RAG) служи за подобряване на инструкциите към големи езикови модели с подходящи данни с цел получаването на по-уместни и по-точни отговори (Lewis et al. 2020). Целта на технологията е да се съчетаят възможностите на големите езикови модели, които са предварително тренирани с общи познания за света, с информация, специфична за дадени приложения и области. RAG

използва векторни бази от данни, в които се съхранява подходящата информация, от която се извличат данни с помощта на семантично търсене и се добавят към контекстния прозорец на подканата. Предимствата при използването на RAG са: подобрена точност на предоставяната от големите езикови модели информация; намаляване или преодоляване на случаите на халюцинация на големите езикови модели посредством генерирането на отговори, които се предизвикват от разширена достоверна информация; фина настройка за специфични тематични области, което позволява решаването на конкретни задачи.

Технологиите за RAG са разнообразни и включват приложения за събиране на данни, генериране на синтетични данни (Shi et al. 2024) или използване на т.нар. процес на разсъждение по веригата на мисълта (chain of thoughts)(Wei et al. 2025). Различните вариации и приложения на RAG са описани в редица изследвания (Gan et al. 2025; Li et al. 2025).



Фигура 1. Архитектура на разработената система

3. Архитектура на инфраструктурата

Проектираната система за автоматично резюмиране на текстове е съставена от четири компонента, показани на фиг. 1 – база от знания (ChromaDB), езиков модел (Ollama LLM Service), потребителски интерфейс (Frontend Service) и управляващ механизъм (Backend Service). Системата включва RAG, което осигурява генериране на изрази на естествен език за разширяване на контекста на инструкциите към езиковия модел с части от предварително зададено съдържание.

Основният компонент на системата е изпълнението на езиковия модел, което се осигурява от софтуера Ollama[1]. Софтуерът Ollama позволява локално използване на големи езикови (и многомодални) модели по няколко начина: директно на предоставените модели в хранилището на Ollama или импортиране на нови модели в GGUF или Safetensor формат. Ollama позволява и персонализиране на моделите чрез настройка на глобални параметри и системни инструкции.

Базата от данни ChromaDB[2] служи за съхранение на векторизираните документи, търсене и извличане на съдържание от тях. Предимството на този софтуер за управление на данните е оптимизацията за работа с матрични структури. Поддържат се функции за бързо семантично търсене (similarity search) на избрана матрична норма. Обикновено входното съдържание се сегментира на малки по обем части, които формират записите в базата от данни.

Потребителският интерфейс на системата позволява да се въвеждат заявки към модела и да се визуализират получените отговори. Той е проектиран да осигури интуитивно взаимодействие с потребителите върху различни по размер екрани. Основният екран съдържа голямо текстово поле за въвеждане на заявки: въпроси, ключови думи или цели текстове. След изпращането на заявката интерфейсът показва индикатор за зареждане, информирайки потребителя, че системата обработва информацията. Когато отговорът е готов, той се визуализира в отделна секция като форматиран текст.

Управляващата услуга координира работата на системата, като, от една страна, предоставя интерфейс на потребителската страница, а от

друга – осигурява взаимодействието с езиковия модел и базата от данни, като обезпечава необходимата междинна обработка на данните.

3.1.Изграждане на базата от знания

Базата от знания е важен компонент в разработваната система, проектирана за резюмиране на текстове на български език (инфраструктурата може да се конфигурира за много различни приложения). Тя служи като структурирано хранилище за семантично представяне на документи чрез техники за векторно индексиране. Основната ѝ роля е да осигурява контекстна информация за големите езикови модели, като ги разширява със специализирани знания от конкретна област (напр. наука и образование). Подготовката на текстовите документи за ChromaDB започва със зареждането им чрез специализирани компоненти като TextLoader за текстови файлове. Като алтернатива могат да се използват PDFLoader за PDF документи с поддръжка на OCR, HTMLReader за уебсъдържание или JSONParser за структурирани метаданни. Тези компоненти са налични в библиотеката LongChain[3] за интегриране на езиковите модели в различни приложения.

След зареждането текстът се разделя на по-малки сегменти. Сегментацията на текста включва разделянето на големи текстови документи на по-малки части (chunks) с цел оптимизиране на изчислителните ресурси и подобряване на точността при семантично търсене. Този подход позволява по-ефикасно индексиране и извличане на информация от базата от данни. Сегментацията на текста се извършва в зависимост от структурата на документа. Например в научни текстове се използва йерархична сегментация, която разделя документа на глави, секции и параграфи. При обикновени текстови файлове се използва рекурсивна сегментация по символи, която разделя текста на части с фиксиран размер и припокриване. Тази техника осигурява използване на контекста от отделните части на текста, което е важно за точността на семантичното търсене.

След сегментацията всеки текстов сегмент се преобразува във векторно представяне. Това се извършва с помощта на модели, които генерират вектори с висока размерност (например 1024). Размерността

на векторите е важна, тъй като тя влияе на точността на семантичното търсене и извличането на релевантна информация. Обучението на моделите за векторно представяне изисква големи количества данни и изчислителни ресурси, но позволява персонализиране на моделите за конкретни приложения чрез фина настройка. Генерирането на векторно-матрично представяне за всеки текстов сегмент се постига чрез използване на подходящ компонент – OllamaEmbeddings от библиотеката LongChain.

Последният етап е съхранението на обработени документи заедно с тяхното матрично представяне в ChromaDB, което се осъществява от обекта Chroma от библиотеката LongChain. Традиционните бази от данни като Postgres или ClickHouse са предназначени за работа със структурирани данни и не предлагат оптимизирано семантично търсене. Те могат да бъдат използвани за съхраняване на метаданни, но не са подходящи за векторни операции. ChromaDB използва алгоритми като HNSW (Hierarchical Navigable Small World) и IVF (Inverted File Index), които позволяват бързо приблизително търсене на най-близки съседи. Тези алгоритми намаляват сложността на търсенето от O(N) до O(log N) и поддържат баланс между точност и скорост чрез параметри като efConstruction и efSearch. Други архитектурни предимства при голям обем на данните са автоматичното разпределяне на колекциите във всички инстанции на базата, LRU (Least Recently Used) кеширане за често използвани вектори и автоматично преиндексиране при добавяне или премахване на записи.

3.2. Взаимодействие с потребителите

Взаимодействието с потребителите се основава на обработката на заявки и генерирането на отговори. Този процес комбинира извличането на информация от базата данни с възможностите за генериране на съдържание от големи езикови модели, за да се предоставят контекстно обосновани и точни отговори.

Когато потребителят изпрати заявка чрез потребителския интерфейс, тя се предава като JSON обект към управляващата услуга. Заявката обикновено съдържа въпрос или ключова дума, която трябва да бъде анализирана. Системата първо преобразува заявката във

векторно-матрично представяне. Векторното представяне позволява заявката да бъде сравнена за сходство с документите в базата от данни ChromaDB. Семантичното търсене идентифицира най-близките документи или части от документи, които съответстват на заявката. Обикновено се извличат няколко (3 – 5) най-близки съвпадения, които ще бъдат използвани като контекст за генериране на отговор.

След извличането на релевантните документи (или части от тях) те се комбинират в единен контекст, който ще бъде използван като вход за езиковия модел. Този контекст може да бъде структуриран така, че да съдържа заглавия, метаданни или други полезни елементи. Целта е да се осигури достатъчно информация за модела, за да може да генерира точен и добре обоснован отговор. След подготовката контекстът се подава като част от потребителската заявка към езиковия модел.

Езиковият модел анализира предоставения контекст и въпроса, след което генерира отговор, който е формулиран така, че да е ясен и разбираем за потребителя. Генерираният отговор се връща към потребителския интерфейс като JSON обект и се визуализира в удобен за четене формат. Ако възникне грешка по време на обработката, системата информира потребителя с подходящо съобщение.

4. Инфраструктура за създаване на чатботове за работа с големи езикови модели

Представената инфраструктура е изградена чрез комбиниране на съвременни уеб технологии и инструменти за контейнеризация, които работят в синхрон. Node.js[4] и Express формират основата на услугата за управление. Node.js представлява JavaScript платформа за сървърни приложения, която позволява обработка на асинхронни заявки и комуникация с други системни компоненти. Express.js[5] улеснява създаването на REST API, като предоставя механизми за диспечериране и обработка на HTTP заявки.

HTML5, CSS3 и JavaScript се използват за формулиране на потребителския интерфейс, като HTML5 осигурява семантична структура на формата за взаимодействие с потребителя, докато CSS3 осигурява адаптивен дизайн и настройка на стилистичните свойства на

семантичните елементи. JavaScript функции управляват потребителското взаимодействие с помощта на интерфейса Fetch API за изпращане на заявки към управляващата услуга и обработване на JSON отговорите.

Услугата Docker[6] заедно с инструмента Docker Compose осигуряват изолирана среда за всеки компонент от разработената система. Контейнеризацията чрез Dockerfile позволява Ollama да работи в собствен контейнер с предварително заредени модели, ChromaDB да се изпълнява като отделен сървър на данни, а управляващата услуга, както и потребителският интерфейс да са с независимо изпълнение. Docker позволява лесно мащабиране и управление на зависимостите между компонентите на системата.

В текстово поле на интерфейса потребителят може да въведе текст за обработка. Текстовото поле е с адаптивен дизайн, позволяващ въвеждането на голям обем текст. В долната част на интерфейса е секцията за отговор, където се визуализира отговорът от езиковия модел. Дизайнът е изчистен, с централизирана структура и адаптивност към различни размери на екрана, което го прави подходящ за използване както на настолни компютри, така и на мобилни устройства.

Изследвано е и динамичното взаимодействие между услугите в системата, когато потребителят изпрати заявка чрез потребителския интерфейс. Управляващата услуга първо се обръща към Ollama за преобразуване на въведения текст във векторно-матрично представяне. Следва заявка към векторната база от данни ChromaDB за семантично търсене на релевантни текстове. След това генерираният контекст се комбинира със заявката на потребителя и се изпраща към Ollama за обработка чрез интерфейс за генериране на отговори. Ollama използва предварително зареден модел, като настройва параметри като размер на контекста, брой GPU слоеве и нишки. Накрая управляващата услуга изпраща резултата обратно към потребителския интерфейс, който го визуализира за потребителя. Показаните логове демонстрират синхронизирана работа между компонентите, откъдето може да се оценят закъсненията в системата и местата за прилагане на

оптимизация, например чрез добавяне на ресурси за паралелна обработка.

```
ollama_1| [GIN] 2025/03/07 - 16:00:24 | 200 | 2.89695479s | 192.168.208.4 | POST
"/api/embeddings"
chromadb_1| INFO:   [07-03-2025 16:00:24] 192.168.208.4:42924 - "GET
/api/v2/tenants/default_tenant/databases/default_database HTTP/1.1" 200
...
backend_1   | <|begin_of_text|><|start_header_id|>system<|end_header_id|>
backend_1   | You are an advanced language model capable of summarizing text and
generating
...
backend_1   | RAG Context: "на име! Във връзката на една покана за посещение в
България, един чужд журналист /немски/ с Име, се изрази така: Благодаря, засега
ми е
...
ollama_1    | time=2025-03-07T16:00:26.183Z level=INFO source=server.go:380
msg="starting llama server" cmd="/usr/bin/ollama runner --model
/root/.ollama/models/blobs/sha256-6a0746a1ec1aef3e7ec53868f220ff6e389f\
6f8ef87a01d77c96807de94ca2aa --ctx-size 8192 --batch-size 512 --n-gpu-layers 33 --threads
28 --parallel 4 --port 38731"
...
ollama_1    | [GIN] 2025/03/07 - 16:00:42 | 200 | 17.979607604s |   192.168.208.4 | POST
"/api/generate"
```

Показана е структурата на използвания вход, който се подава към езиковия модел. В примера за демонстрация е използван вариант на модела Llama3 с 8 милиарда параметъра и 4-битово квантуване на теглата[7]. За получаване на векторно-матрични представяния е използван BERT моделът[8] mxbai-embed-large. В проектирания входен израз първо се използва семантичен ollama маркер “<|start_header_id|>system<|end_header_id|>” за подаване на системни команди към модела, като се уточняват изискванията, ролята и задачата на модела, изискванията за изходен формат и език. Следва семантичният маркер за потребителски вход <|start_header_id|> user <|end_header_id|>, като в тази секция се задават входният текст и извлеченият контекст от базата данни. Входният израз завършва със

семантичния маркер <|start_header_id|> assistant <|end_header_id|>, с което се очаква генериране на отговор от модела.

```
const prompt = `<|begin_of_text|><|start_header_id|>system<|end_header_id|>
        You are an advanced language model capable of summarizing text and generating
structured outputs. Your task is to:
        1. Summarize the provided text using context retrieved from external sources
(RAG context).
        2. Output the summary strictly in JSON format.
        3. Answer in Bulgarian language.
        4. The JSON should include:
        - "keywords": A list of 5 key terms that capture the essence of the text.
        - "questions": A list of 3 questions that should be addressed based on the text.
        - "categories": A classification of the text into 3 categories.
        - "summary": A short summary of the provided text

        Ensure your response is valid JSON with no additional text outside the JSON
object.

        <|eot_id|><|start_header_id|>user<|end_header_id|>
        Here is the input text: "${query}"

    RAG Context: "${results.map(r => r.pageContent).join("\n")}"

    Expected Output Format:
    {
        "keywords": ["ключова дума 1", "ключова дума 2", "ключова дума 3",
"ключова дума 4", "ключова дума 5"],
        "questions": ["Въпрос 1", "Въпрос 2", "Въпрос 3"],
"categories": ["Категория 1", "Категория 2", "Категория 3"],
        "summary": "резюме на текста"
    }
        <|eot_id|><|start_header_id|>assistant<|end_header_id|>`;
```

5. Заключение

Представената система демонстрира успешно интегриране на Retrieval-Augmented Generation (RAG) архитектура за обработка на документи на български език. Чрез комбинация от Ollama за управление на големи езикови модели, ChromaDB за векторно търсене

и адаптивен уебинтерфейс системата позволява контекстно разширяване на заявки с минимални изчислителни ресурси. Проведени са експерименти с използване на локално изпълнявани модели като Llama3 и BERT базирана векторизация. Изследвана е точността на семантичното търсене и генерираните резюмета. Авторите предоставят пълния програмен код на системата[9].

Системата предлага две ключови предимства: 1. сигурност на данните чрез локално изпълнение; и 2. гъвкавост по отношение на персонализацията на моделите за специфични области. Производителността остава зависима от хардуерните ресурси, а качеството на изходите се влияе от липсата на специализирани предварително обучени модели за български език. Допълнително ограничение е трудността при обработка на документи със сложна визуална структура.

Перспективите за развитие включват внедряване на многомодални модели за обработка на текст, изображения и таблици, разширяване на функционалностите за различни задачи, например автоматизирана класификация на документи.

Благодарности

Изследването е разработено в рамките на проекта „Инфраструктура за фина настройка на предварително обучени големи езикови модели“, договор № ПВУ – 55 от 12.12.2024 г. (BG-RRP-2.017-0030-C01).

БЕЛЕЖКИ

1. Ollama: https://ollama.com/ [прегледан на 10.3.2025]
2. ChromaDB: https://www.trychroma.com/ [прегледан на 10.3.2025]
3. LongChain: https://www.langchain.com/langchain [прегледан на 10.3.2025]
4. Node.js, безплатна, с отворен код, междуплатформена среда за изпълнение на JavaScript: https://nodejs.org/en [прегледан на 10.3.2025]

5. Express.js, Node.js уебплатформа за приложения, която предоставя набор от функции за уеб- и мобилни приложени: https://expressjs.com/ [прегледан на 10.3.2025]
6. Docker: https://www.docker.com/ [прегледан на 10.3.2025]
7. Вариант на модела llama3 с 8 милиарда параметъра и 4-битово квантуване на теглата:
https://ollama.com/library/llama3 [прегледан на 10.3.2025]
8. Моделът BERT: https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1 [прегледан на 10.3.2025]
9. Изходен код на системата: https://github.com/DCL-IBL/dcl-summarizer [прегледан на 06.06.2025]

REFERENCES

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., & Mcgrew, B. (2023). GPT-4 Technical Report. *arXiv preprint* arXiv:2303.08774.

Devlin, J., M.-W. Chang, K. Lee, & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1 (Long and Short Papers), pp. 4171 – 4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Fang, D., Qiang, J., Zhu, Y., Yuan, Y., Li, W., & Liu, Y. (2025). Progressive Document-level Text Simplification via Large Language Models. *arXiv preprint* arXiv:2501.03857.

Gan, A., Yu, H., Zhang, K., Liu, Q., Yan, W., Huang, Z., ... & Hu, G. (2025). Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey. *arXiv preprint* arXiv:2504.14891.

Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). Llama: Open and efficient foundation language models. https://arxiv.org/abs/2302.13971

Henkel, O., Levonian, Z., Li, C., & Postle, M. (2024). Retrieval-augmented generation to improve math question-answering: Trade-offs between groundedness and human preference. In: B. Paaßen and C. D. Epp, editors, *Proceedings of the 17th International Conference on Edu- cational Data Mining*, pp. 315 – 320, Atlanta, Georgia, USA, July 2024. International Educational Data Mining Society.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., De Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan T., Noland, E., Millican, K., Van Den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W. , Vinyals, O., & Sifre L. (2022). Training compute-optimal large language models. *arXiv preprint* arXiv:2203.15556

Kaplan, J., Mccandlish, S., Henighan, T, Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling laws for neural language models.
https://arxiv.org/pdf/2001.08361/1000

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, no. 33, pp. 9459 – 9474. https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

Li, S., Stenzel, L., Eickhoff, C., & Bahrainian, S. A. (2025). Enhancing Retrieval-Augmented Generation: A Study of Best Practices. In: *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 6705 – 6717, Abu Dhabi, UAE. Association for Computational Linguistics.
https://aclanthology.org/2025.coling-main.449/

Murtaza, S. S. Nie, Y., Avan, E., Soni, U., Liao, W., Carnegie, A., Mathias C. J., Jiang, J., & Wen E. (2025). Implementing Retrieval Augmented Generation Technique on Unstructured and Structured Data Sources in a Call Center of a Large Financial Institution. In: *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human*

Language Technologies (vol. 3: Industry Track), pp. 598 – 606, Albuquerque, New Mexico. Association for Computational Linguistics. https://aclanthology.org/2025.naacl-industry.48/

Ouyang, L., Wu, J., Jiang, W., Almeida, D., Wainwright, K. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, L., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, K., & Lowe R. (2022). Training language models to follow instructions with human feedback. In: Oh, A. H., Agarwal, A., Danielle Belgrave, D., Kyunghyun Ch., editors, *Advances in Neural Information Processing Systems*. https://openreview.net/forum?id=TG8KACxEON.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever. I. (2018). Improving language understanding by generative pre-training. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Shi, Z., Zhang, S., Sun, W., Gao, S., Ren, P., Chen, Z., & Ren. Z. (2024). Generate-then-ground in retrieval-augmented generation for multihop question answering. *In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics* (vol. 1: Long Papers), pp. 7339 – 7353.

Sun, X., Li, X., Li, J., Wu, F., Guo, S., Zhang, T., & Wang, G. (2023). Text Classification via Large Language Models. In: *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 8990 – 9005, Singapore. Association for Computational Linguistics. https://aclanthology.org/2023.findings-emnlp.603/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wei, Z., Chen, W.-L., & Meng, Y. (2025). InstructRAG: Instructing retrieval-augmented generation via self-synthesized rationales. *arXiv preprint* arXiv:2406.13629.

Xu, R., Shi, W., Zhuang, Y., Yu, Y. And Ho, J. C., Wang, H., & Yang, C. (2025). Collab-RAG: Boosting Retrieval-Augmented Generation for Complex Question Answering via White-Box and Black-Box LLM Collaboration. *arXiv preprint* arXiv:2504.04915.

INFRASTRUCTURE FOR DEPLOYING LARGE LANGUAGE MODELS IN CHATBOTS WITH RETRIEVAL-AUGMENTED GENERATION

Abstract. The article presents an infrastructure that integrates freely available tools and large language models into a personalized chatbot. The aim is to show how large language models can be used with affordable graphics processors and what possibilities exist to enrich user queries with additional context depending on the objective and area of application. Working with pre-trained large language models will be demonstrated using RAG (Retrieval-Augmented Generation) technology for Bulgarian to solve a specific task in a particular thematic domain. The chosen task is document summarization, and the domain is science and education. The presented solution can also be used to create other applications for the Bulgarian language.

Keywords: large language models; artificial intelligence; chatbots

✉ **Dr. Jordan Kralev, Assist. Prof.**
ORCID iD: 0000-0002-5416-2115
Department of Systems and Control
Faculty of Automatics
Technical University of Sofia
8, Kliment Ohridski Blvd.
1700 Sofia, Bulgaria
&
Department of Computational Linguistics
Institute for Bulgarian Language Prof. Lyubomir Andreychin
Bulgarian Academy of Sciences
52, Shipchenski prohod Blvd., Bldg. 17
Sofia, Bulgaria
E-mail: jkralev@tu-sofia.bg

✉ **Prof. Dr. Svetla Koeva**
ORCID iD: 0000-0001-5947-8736
Department of Computational Linguistics
Institute for Bulgarian Language Prof. Lyubomir Andreychin
Bulgarian Academy of Sciences
52, Shipchenski prohod Blvd., Bldg. 17
Sofia, Bulgaria
E-mail: svetla@dcl.bas.bg