Large Language Models for Identification of Medical Data in Unstructured Records

Presian Petkov[1], Latchezar Tomov[2,3](✉), and Emanuil Markov[4]

[1] B.IT., New Bulgarian University, 1618 Sofia, Bulgaria
npecko@protonmail.com
[2] Department of Informatics, New Bulgarian University, 1618 Sofia, Bulgaria
lptomov@nbu.bg
[3] Medical Faculty, Sofia University St. Kliment Ohridski, 1 Kozyak Street, 1407 Sofia, Bulgaria
[4] Technical University of Sofia, Sofia, Bulgaria

Abstract. In Bulgarian healthcare there are some unique challenges, due to the nature of the medical data, being unstructured. Some key predictors for health such as the smoking status are described in an unstructured way in medical records, which makes impossible simple retrieval with or without regular expressions, or manual check due to the large number of records available. Large Language Models are a viable alternative to automate the process of extraction of medical data from unstructured records due to their ability to process natural language, even some for which they lack enough or specific training, such as Bulgarian language. We develop a method and a procedure to test multiple LLMs for that purpose and obtain some promising results that show their capabilities. We analyze the results in the context of the price of the service and the computational costs.

Keywords: Large Language Models · Diabetes · Healthcare · Information extraction · Smoking

1 Introduction

1.1 Definition of the Problem

The Large Language Models have risen to popularity in recent years due to their ever-growing number of applications and a skyrocketing improvement cycle. Our goal is to utilize their potential to solve the problem of analyzing medical data in unstructured records.

Such data can be formatted in various ways and its interpretation relies heavily on the embedded context. For the purposes of this research, we were provided with a dataset consisting of medical records – discharge summaries or ambulatory sheets. In accordance with the standards set by the National Health Insurance Fund, these records contain at minimum the following sections: *Medical History*, *Status*, *Treatment*, and *Examinations*. Sections Treatment and Examinations are relatively simple to deconstruct, using conditions and regular expressions. They contain certain keywords – names or

T. Zlateva and G. Tuparov (Eds.): CSECS 2024, LNICST 609, pp. 119–135, 2025.
https://doi.org/10.1007/978-3-031-84312-9_8

unique identifiers of the examinations (standardized by the NHIF), names of medicinal agents – ATC classification, as well as the respective Bulgarian names featured in the National Council on Prices and Reimbursement of Medicinal Products' dedicated list. Sections Medical History and Status consist of freeform text. Devising rules for the extraction of the data proves to be a difficult task, due to the large array of variants in which it can be structured. This includes the length of time since a given patient has been diagnosed with a condition, whether they have unhealthy habits or the general condition of the patient.

In turn, the Large Language Models are effective in their analysis of data that does not adhere to strict patterns, and at the same time do not require specialized knowledge of programming. They accept instructions for the expected output. A significant leap in the development of LLMs was observed in 2022 and especially in 2023, that allowed the use of models on a local workstation, or the ability to use them via an API endpoint.

The task at hand is to research the currently available LLMs and evaluate their performance. There are certain prerequisites which must be met to achieve satisfactory results, and that involves careful consideration and the devising of test-case scenarios. In addition, we must deal with some problems and unexpected outcomes along the way, taking note of the shortcomings of the models we choose in relation to our goal. [1].

2 Selection of Models and Data

2.1 Overview of the LLM Models

At the moment of investigation, we have a wide array of opportunities at our disposal as to what LLMs to use. In the beginning, our plan was to test the capabilities of BERT via the ML. NET framework. It was quickly discovered that it will not, in fact, suffice for our needs. This is why we continued our research and investigated other models. Some of them performed poorly. Others were superb in their performance, but in their current stage could not be used in a scalable or automated way. After thorough experimentation, several models were chosen above the rest, with a mixture of free and paid licenses.

Since the process of selection might be of interest or value to other research, we will go over all the models we tested, including those that did not make it to the final lineup. The next table shows some of the most popular language models, as well as containing technical details about them. [2]. (Table 1)

BERT

B.E.R.T or Bidirectional Encoder Representations from Transformers is the name of a large language model. It was introduced back in 2018 and developed by a team of researchers at Google. A study conducted in 2020 concluded that "in less than a year, BERT become a ubiquitous baseline in Natural Language Processing (NLP) experiments counting over 150 research publications analyzing and improving the model." [3].

Bert uses *WordPiece* [4], which converts each word to an integer code. The vocabulary size is 30,000. Tokens that do not appear in its vocabulary are replaced with [unk] (unknown). During the process of pre-training, the Toronto BookCorpus [5] with a size of 800 million words and the English Wikipedia (2.5 billion words) were used. The model was pre-trained on a couple of tasks simultaneously. Language modeling, that

Table 1. Popular LLMs

Name	Release date	Developer	Parameters	Corpus Size (Mixed)
BERT	2018	Google	340 million	3.3 billion words
GPT-2	2019	OpenAI	1.5 billion	10 billion tokens
GPT-3	2020	OpenAI	175 billion	300 billion tokens
GPT-4	2023	OpenAI	1.76 trillion	45 TB of text
LaMDA	2022	Google	137 billion	168 billion tokens
LLaMA	2023	Meta	65 billion	1.4 trillion tokens
LLaMA 2	2023	Meta	70 billion	2 trillion tokens

entailed 15 percent of tokens being selected with the objective of predicting the selected token given its context. And subsequently - sentence prediction. The model predicts if two spans of text appeared sequentially in the training corpus, outputting either [IsNext] or [NotNext]. [6] The weights for the model are open-source on GitHub[1].

BERT has "encoder-only" transformer architecture. It cannot perform text generative tasks or be prompted. In comparison, bidirectional models don't function as well without the right side, creating an obstacle for prompting purposes. Even short text generation requires advanced, as well as expensive in terms of computation, methods [7].

The model can be fine-tuned with small amounts of data for more specific tasks. This includes mood analysis, product recommendations, price and sales forecasts, user classifications, object recognition, fraud detection, sale spikes, and image classification. Those can all be beneficial for commercial business needs, but this functionality allows us to fine-tune the model with the medical records we have at our disposal with 80% of the data used for training and 20% for testing, to avoid overfitting.

To make use of BERT, we developed a program with the ML.NET framework, *TorchSharp* (.NET library that provides access to *PyTorch*), and *LibTorch CUDA*. As we have mentioned, the data must be pre-labeled to be used for training. Since our team did not have the capabilities to manually do so, an automated solution had to be devised, and we went with regular expressions. Before beginning the project, this approach was considered and the shortcomings of it became obvious rather quickly, which is why the need for using LLMs arose in the first place. One such example is false positives, where a pattern can be found in a sentence, but excludes any context, and having to account for that is no short of scope creep and ultimately putting in as much work as manually labeling or even more. Nevertheless, this step was a prerequisite, so a simple Python script was written that categorized our data using RegEx. There were four labels we provided – *unknown*, *smoking*, *hypertension*, and *diabetes*. The code splits the CSV file line by line, extracts the sentence and label, and trains the model according to the 80/20 distribution. After this, it outputs metrics provided by ML.NET like micro/macro

[1] https://github.com/google-research/bert.

accuracy and logarithmic loss[2] that evaluate the performance of the model on whatever data remains which it was not trained on. In the last step, it starts a chat-like sequence that waits for input from the user and tries to make a predicted classification based on the text.

Training the model was relatively fast for the hardware capabilities and volume of the dataset at our disposal (CPU - Intel i7-9850H, RAM – 16 GB, HDD; Total training time – 7 min). The CUDA integration allowed for the processes to be offloaded to the graphical processor (Mobile GPU Nvidia RTX2070). In the end, the results were not good. We did not manage to receive accurate predictions, and the requirement for manual classification of the data was not an intended part of our workflow.

LLaMA

The first version of LLaMA had four trained models. They have 7, 13, 33 and 65 billion parameters. The biggest model is on the same level as PaLM and Chinchilla. [8] Meta released the weights for the community of researchers, but only a week after that, the models were distributed to the public through the BitTorrent network, as the result of a 4Chan leak.

Partnering with Microsoft, on July 18th, 2023, Meta made an announcement for their new generation of the model - LLaMA-2. It came in three volumes in terms of parameters– the largest being 70 billion, 13 billion, and 7 billion. There were diminishable changes to the architecture, but 40% more data was used for the training. [9] There was discussion of a model with 34 billion parameters that might be released to the public if it met certain security requirements. In addition to general-purpose models, LLaMA-2 also includes models that are fine-tuned for interactive dialogue, called LLaMA-2 Chat. In an even greater act of distinction, all the weights were released to the public and are free of charge for applications with commercial intent[3]. Despite this, because of some limitations, the Open-Source Initiative, who are in charge of maintaining the "Open-Source Definition", does not agree to describe LLaMA as "open-source". [10].

Sources for the training data include web pages extracted by *CommonCrawl*, open-source GitHub repositories, Wikipedia in twenty languages, Project Gutenberg books in the public domain, LaTeX source code of scientific articles in ArXiv and questions and answers in Stack Exchange.

After the first version of LLaMA was released, Georgi Gerganov developed *llama.cpp*, an inference of the Meta model in C/C + +.[11] It allows quantization of the models, which is a process where the weights' floating-point precision is decreased, ranging from 32-bit or 16-bit floats to between 8-bit and 2-bit floats or integers. The quantization method used is GGUF (successor to GGML), that allows the models to be stored in a single file and optimized for CPU inference, as opposed to other quantization methods, namely *AWQ*, which favors GPUs [12].

The installation is simple. We clone the repository and build the main program. The next step is to quantize the models using the provided built-in methods. For LLaMA-1 we went with the 13 billion parameters model and 4-bit quantization. That decreased

[2] Log loss is an essential metric that defines the numerical value bifurcation between the presumed probability label and the true one, expressing it in values between zero and one.

[3] https://llama.meta.com/llama-downloads/.

the size from 24 GB to approximately 8 GB and could be run on a mobile CPU with a minimum of 10 GB of RAM.

Two important settings to mention here are prompt engineering and parameter tweaking. The "prompt", or input text, is a major contributor to what output text will be generated. We found that staging the input by prefacing our question and expected subsequent answer with *roles* yielded better results. Our questions would be prefaced with "User:" and the answers from LLaMA would start with "Assistant:". Once this was done, we could fine-tune how we want the answer to be structured, by adjusting the parameters settings. This entailed the number of predicted tokens during the generation process, less is better for a more precise response that does not stray from our purpose. Also, the context size, where the maximum value is that on which the model was trained. CPU thread count that can be dedicated for computational purposes, or parameters related to the GPU should we want and have the means to offload some of the work. Another parameter for tuning is the temperature, which controls the "randomness" of the output [13]. Again, we are aiming for more accurate, shorter replies. *Top-K* and *Top-P* sampling that picks tokens with higher probability to be next. And finally, interactive mode. This is not trivial since as we discussed, conditioning the model to account for roles improves the output.

A part of the answers in our controlled tests, where we aimed to find weak points, were correct. In some cases, the string would be cut short due to our token limitation, but this means that the model did not adhere to our rules for how we want the reply to be structured. Sometimes it would become too creative, a problematic example of which is imagining a patient's diagnosis and contributing to its length with various other conditions. Even in scenarios where it managed to find the sought-after information, and consider our reference of key context, the predicted text was meaningfully wrong.

Shortly after LLaMA-2 was released, an API was deployed for users of the llama.cpp project. This is what we used for the newer model. A detailed description of the requests we made to it can be found in the next section of this paper, for OpenAI's GPT models, as the process is almost identical and easily interchangeable between the two. The performance was noticeably improved.

Apart from the base models engineered by Meta, fine-tuned models began emerging in the machine learning community. We will make use of those in our final grading.

Vicuna is a version of LLaMA, which was fine-tuned with user interactions. *ShareGPT* is how these interactions were collected. According to initial evaluations that referenced GPT-4 as a judge, Vicuna-13B scored greater than 90% the quality of Bard and ChatGPT. It also performed better than other models like LLaMA and Stanford's Alpaca, again in 90% plus percent of the cases. The price for training Vicuna-13B was around three hundred dollars. The code and weights, along with the demo, are publicly available. [14].

Wizard Vicuna is an amalgamation of WizardLM's dataset, ChatGPT's conversation extension and VicunaLM's tuning method. The benefits are an improved handling of the dataset itself - given the fine-tuning and additional context from chat-type user interactions. In addition, VicunaLM overcomes the limitations of single-turn conversations. It does so by introducing multi-round conversations. A 7% performance increase over VicunaLM is reported. [15].

Hermes is another large language model. It was fine-tuned on over 300 thousand instructions and was trained exclusively via generated output from GPT-4. This resulted in an enhanced LLaMA model, that could rival the performance of GPT-3.5 across numerous assignments. What separates Hermes apart are lengthy responses, less "hallucination" in its responses, and a lack of censoring. [16].

GPT

Generative pre-training, also known as GP, is an established concept in machine learning [17]. In 2017, Google developers published the paper "Attention Is All You Need", that featured the modern transformer architecture. [18] This advance led to the development of XLNet, in addition to BERT. They are examples of pre-trained transformers (PT), which were not developed to be generative - they only have an encoder. OpenAI published in 2018 "Improving Language Understanding by Generative Pre-training", thus introducing the first system featuring a generative transformer that was trained in advance – GPT-1. [19] The successor GPT-2 has more parameters by a factor of 10 (1.5 billion), also pre-trained on BookCorpus. It was also trained with eight million pages taken from the web. On the topic of its outputs, Vox said "the prose is pretty rough, there's the occasional non-sequitur, and the articles get less coherent the longer they get". [20] The Verge noted that samples containing a larger amount of generated text from GPT-2 had the tendency to stray away from the initial topic and were not very coherent. [21].

We tested GPT-2 by using the transformers Python library. We hosted a *Flask* application. It loads the GPT-2 model and accepts requests. In addition to it, a Python program was used, that connected to the local host, sent the prompt as JSON data, and extracted the model's prediction from the response. We could not get any good results. Perhaps due to the nature of our data, and the limitations of this version of GPT, but it faced the same problems as the journalists outlined above, as well as lacked the ability to translate the text and find meaningful context that would serve our purposes.

On the 28th of May 2020, in a pre-published project on arXiv by OpenAI, 31 engineers describe their accomplishments on the development of GPT-3. [22] GPT-4 was launched on the 14th of March 2023. OpenAI made a statement regarding GPT-4 – that it was "more reliable, creative, and able to handle much more nuanced instructions than GPT-3.5". [23] While undisclosed by OpenAI, based on the speed, and evaluation done by one George Hotz, it was estimated that the fourth model has approximately 1.76 trillion parameters. [24] Both of these models are trained for natural language processing and interpreting programming code. They allow for input text without the need for additional parameters, mimicking a human-like chat interaction.

The most important benefit for our team is the ability to connect to a REST API. Doing so is associated with costs related to the number of input/output tokens, those can be seen on OpenAI's pricing page. As of April 2024, GPT-4 is 60 times more expensive per million input tokens and 40 times more expensive per million output tokens compared to GPT-3.5.

To connect to the endpoint, we install OpenAI's Python library, choose the model we want to use, send a *messages* body that contains our role and prompt, and receive a JSON response, from which we take the relevant information - the *content*, the generated text. Our questions are a part of an *enum* data structure. They are written in the Bulgarian

language, the question is prefaced with a "User:" role, followed by a placeholder, where we will iteratively insert the medical history entries from our input dataset, the expected format of the answer from the LLM, and a suffix of its role – "Assistant:". In conclusion, a limit is put on the maximum number of tokens contained in the output, in order to get a more precise answer and extract the prediction from the response. The resulting file will then be used for our final program, which will evaluate the respective model's performance.

BgGPT

This is a Bulgarian language model, created by the INSAIT Institute, part of Sofia University, and trained from the 7-billion parameter Mistral.

The model is fine-tuned with the intent of improving its capabilities in the Bulgarian language. This is done by using several datasets. Those include specialized datasets sourced by INSAIT Institute, web crawl data, and machine translations of English datasets that have garnered some popularity. The Bulgarian data is then adjusted with the help of English data, seeking the goal of retaining reasoning skills.

The model's tokenizer has been upscaled. This allows for a more efficient encoding process of Bulgarian words, that are written via the Cyrillic alphabet. Thus, not only increasing the throughput of Cyrillic words, but also of the performance. [25].

For our testing, we used the quantized 8-bit GGUF version of the model, as the original is not large to begin with, and would not require a lot of hardware capabilities. Some observations that can be made after generating the output are that the model has the tendency to answer in English or produce lengthier amounts of text, that end up being cut off, despite a lack of restriction regarding the number of allowed tokens, even though we still expect concise answers. Another factor worth mentioning is a significant percentage of false positives, as can be observed later in the confusion matrix. The reason for this, if reviewed manually, is a pattern of providing information about the duration of the disease in a patient's diagnosis, where said duration is either wrong or there is no data in the original text to begin with. Despite this, the model exhibited satisfactory results and a relatively high accuracy level. The accuracy further improved with the second version, producing fewer false positives.

Claude

Claude represents a group of LLMs. The developer is the company Anthropic. These models are GPTs; fine-tuned with "Constitutional AI" and Reinforcement Learning from Human Feedback.

Anthropic has developed an approach to aid the training of AI systems, LLMs like Claude in particular, aiming to improve their helpfulness and harm-reduction, but also eliminating the reliance on large amounts of human interference. It is called Constitutional AI, and it is detailed in the paper "Constitutional AI: Harmlessness from AI Feedback". The process involves two phases. The first being supervised learning. And the second being reinforcement learning. The model is expected to generate responses to prompts during the supervised learning phase. It then critiques these responses itself, following a ruleset set of principles, guidance – a "constitution", and then makes a revision of the responses. Regarding the reinforcement learning phase, that involves a

training process for the model that uses AI-generated feedback. AI makes an evaluation of the responses in accordance with the principles detailed in the aforementioned constitution. All of this allows assistants to be trained in a way that they can express concerns or straight-out object requests deemed harmful. Transparency is enhanced, and less human supervision is required. A total of 75 sections can be found inside the "constitution" for Claude. Some of them include paragraphs from the United Nations' "Universal Declaration of Human Rights". [26].

The model proved to be proficient in a variety of tasks. However, it also exhibited some limitations in the fields of math and programming, in addition to reasoning. Two versions of Claude were released – a basic one and "Instant". The latter being less expensive, lighter, and faster. The token input context length was 100 thousand. Claude 2 followed soon after. It was available to the public, unlike the first ones, which were only available to a select few, which had to be approved by Anthropic, prior to exploitation. Claude 2 expanded the context window, and offered the ability to upload documents that Claude can process. There was some backlash caused by reduced usability, due to strict ethical guidelines. In some cases, assistance with requests was refused or considered benign, i.e., programming inquiries. With Claude 2.1, the number of tokens was increased to 200 thousand. At the time of writing, Claude 3 is the latest addition. It comes in three models – Haiku, Sonnet, and Opus. The performance and price vary between them, although in our testing there was no significant difference exhibited between the lowest-range model – Haiku, and the flagman – Opus. Claude 3 has been shown to perform meta-cognitive reasoning. This includes self-realization in the process of testing during some benchmark evaluations. [27] While subject to change, applicable to both parties, the high-end Opus model is 50% more expensive per million input tokens than GPT-4 Turbo, and 250% more expensive for a million output tokens.

Gemma

Gemma are a group of light, cutting-edge and open-source models. They are developed by the DeepMind team behind Gemini, featuring the same level of technological advancements and utilizing the research put into them. The name is inspired by Gemini, and comes from "gemma", translating to "precious stone."

The weights of the models are released in two sizes – 2B. And 7B. Parameters. The smaller model is intended for mobile devices, and laptops, while the larger model is developed for desktop computers and small servers. Each model is released with pretrained and instruction-tuned versions. Pretrained versions are not trained on any specific tasks or instructions beyond the Gemma core data training set. Instruction-tuned versions are trained with human language interactions and can respond to conversational input, like a chat bot. It is reported that Gemma performs much better compared to models larger than it on a variety of industry-leading benchmarks. [28].

Mistral

Mistral is a decoder-based LLM with seven billion parameters. It employs several innovative architectural choices [29]. These include:

- Sliding Window Attention – This feature allows the training of the model to be done with a context length of 8k and a cache of fixed size. As a result, we have 128k tokens for the attention span.

- Grouped Query Attention – Enables inference that is faster and a cache of a smaller size.
- Byte-fallback BPE tokenizer – Characters are not mapped to tokens that fall outside of the vocabulary.

The model demonstrated superior performance in comparison to others in its category. Mistral claims to outperform Meta's larger LLaMA-2 models and match or exceed GPT-3.5 on specific benchmarks. In our personal testing, the former is true, at least for the 13B LLaMA-2 we chose, however this is subjective, and depends on the tasks at hand. It did not outperform GPT-3.5 Turbo. Mistral AI is significantly cheaper than GPT models. It is around 187 times cheaper than GPT-4 and about 9 times cheaper than GPT-3.5. While GPT-4 is not strictly a language-only model and can take inputs such as images and text, Mistral offers an alternative that has a balance of accessibility, cost, and other capabilities. GPT-4 is not open source and requires API access. Mistral models, on the other hand, can be found on Hugging Face (company that develops computation tools for building applications using machine learning).

In addition to the base model, there is an "Instruct" variant, which is fine-tuned to follow instructions. There is also a larger model called "Mixtral 8x7B". Some of the key differences are that Mixtral has a similar architecture to Mistral, but each layer in Mixtral comprises eight feedforward blocks. It is reported to outperform LLaMA-2 70B and GPT-3.5 on most standard benchmarks, also surpassing Mistral 7B. Mixtral uses only 13B active parameters for each token, which is five times less than LLaMA-2 70B, making it much more efficient. All of this comes at a cost. Mixtral requires more resources than Mistral. While Mistral works well on a 24GB RAM 1 GPU instance, Mixtral requires 64GB of RAM and 2 GPUs [30].

Mistral is a versatile and powerful generative text model that can be used for various applications, some of which are content creation, education and natural language processing.

Final Choice of Models

Listed below are the models we ended up evaluating. The majority of them are free to use and were tested using llama.cpp's API, after downloading the models locally from *Hugging Face*. Anthropic's Claude and OpenAI's GPT models have their own API endpoints. They have costs associated with the number of tokens - input and generated output, on a pay-per-usage basis. The testing solutions for the free and paid models were almost identical in structure[4].

1. *llama-2-13b.q8_0.gguf*
2. *hermes-2-pro-mistral-7b.q8_0.gguf*
3. *vicuna-13b-v1.5.q8_0.gguf*
4. *wizard-vicuna-13b.q8_0.gguf*
5. *bggpt-7b-instruct-v0.2.q8_0.gguf*
6. *mistral-7b-instruct-v0.2.q8_0.gguf*
7. *gemma-7b.q8_0.gguf*
8. *claude-3-opus-20240229*

[4] https://github.com/npecko/LLM-Medical-Analysis/tree/main/Scripts.

9. *gpt-3.5-turbo*
10. *gpt-4*
11. *gpt-4–0125-preview*

2.2 Data Selection

As previously mentioned, we had a dataset consisting of medical records at our disposal. That contained 10,000 entries in total. Let us observe how those were structured (Table 2).

Table 2. Input Data

Patient ID	Medical History	Hospital State
3	Страда от диабет. Провежда лечение с перорални препарати. С добър контрол на кръвна захар и кръвно налягане	КОЖА-суховата с нормален тургор и оцветка
9	Температура, кашлица, отпадналост	ГЛАВА и ШИЯ-Без особености; ПЕРИФЕРНИ ЛИМФНИ ВЪЗЛИ-Неувеличени
65	Редовен контролен преглед за терапия, добро общо състояние, не съобщава оплаквания	ЕЗИК-суховат; ГЪРЛО-Спокойно

Translation Row-Per-Row

1. Suffers from diabetes. Treatment includes oral medication. Good control of blood sugar and blood pressure; SKIN – dry, normal turgor and color.
2. High temperature, cough, fatigue; HEAD and NECK – no abnormalities. PERIPHERAL LYMPH NODES – not enlarged.
3. Regular control exam for therapy, good general condition, does not report complaints; TONGUE – dry, THROAT – calm.

Now that we elaborated on the contents of the dataset, let us review what factors could potentially be problematic when trying to extract and analyze the data. The formatting of the hospital records is inconsistent. One entry could contain the string “Regarding patient X…”, while another could be written as “The woman is in good physical condition.”. Should we refer to the person as a *patient*, it is not entirely evident that the model will understand our intent. Another example is the formatting of the length of time. “Patient has been diagnosed since 2015.” can have the same meaning as “Patient has been diagnosed for 9 years.”, when read in 2024. Context is crucial for the questions we will pose, and it is dependent on a “weak” spot of the models, namely the end point in time of their learning. To understand this better, if the given implementation that utilizes the models does not have a method for retrieving the current year, the model will fall back on its training data or use context provided by the user, if there is any. The training data contains text, the contents of which are written up to a certain year before the training was cut off. If the current year is explicitly mentioned, then the LLM

has a better chance of delivering an accurate output. Arithmetic operations are also not actually executed, instead the most likely subsequent token is generated, based on the corpus which was used for the learning. Finding the difference between two years is not a particularly trivial task.

Semantic interpretation is also very important. Confusion can arise from substrings containing part of the needed information, but not in their respective context. The algorithm needs to extrapolate key data points and connect them in a meaningful way, in relation to the questions we have forwarded. For our input dataset, support for the Bulgarian language was a must. To add on to the complexity, analyzing medical text is not a general use case, especially for models that are fine-tuned to be chat assistants, and can also contain abbreviations that need to be interpreted properly. Lackluster results on this point do not bode well for the model's overall performance, as it became clear during testing.

Choice of Questions

Having considered all the aforementioned factors, our team came up with a set of questions to evaluate the final group of models. Those questions and the expected output of the answers are:

1. Does the text contain information about the patient's bad habits? – Yes or No.
2. Is the patient a smoker? – Yes or No.
3. Does the text contain information about the duration of diabetes in the patient? – Yes or No.
4. What is the duration of diabetes in the patient? – Number of years.

From the original input volume, we chose 128 entries in total, which were characterized by containing various formatting, information, and level of complexity, to best determine and test the models' capabilities in a wide array of scenarios. They would be used for the generation of output data in a controlled way.

Given that we had four questions per entry, this resulted in 512 manually classified examples[5]. The answers are formulated in the way we expect from our models – the first three were *yes* or *no*. The last one is an integer, given there is information in the text. If there is none, then we will default to zero.

3 Results

After the careful manual classification of the input data, we must now evaluate the performance of each of the models. To do so, we wrote a script in Python that serves two functions – outputs a table with some of the most widely recognized metrics, used in statistics and machine learning, as well as a heatmap of the confusion matrix.

First, we extract the answers returned by the APIs and perform an iterative logical comparison for each of the values – manual vs. predicted. The measures we employed below rely on the metrics (True/False) (Positives/Negatives). They are self-explanatory, but to give one example – if we have classified something as True and the model has also classified it as True, then we consider the output a True Positive. False Negatives and False Positives can be statistically considered as Type II and Type I errors, respectively.

[5] https://github.com/npecko/LLM-Medical-Analysis/tree/main/Evaluation.

Going over the first three segments of answers is straightforward, but we need to consider how we handle the duration of the disease in number of years. During our testing, if you were to look at the various answers by different models for the last question, a pattern could be observed. In many of the cases, the model would first translate either a part of the text, or the text in its entirety, which, of course, would include one reference to the year(s). Subsequently, an arithmetic operation would be displayed, i.e., *current year minus year in text*. At the end you would get the duration as an integer value. Unfortunately, despite our best efforts with prompt engineering, tweaking the parameters for receiving a more accurate, less creative, restricted result, this pattern would emerge nonetheless, all too frequently. If a reviewer looks at the output, the information could suffice in providing an accurate answer, but only when interpreted in a natural language semantic. Expecting to only receive the final number is unrealistic with the current state of LLMs and is setting them to fail and underperform. However, there is no compromise to be made if the model gives a number that is incorrectly attributed to the disease or does not appear in the source altogether. For these reasons, a decision was made, that we would judge the output based on whether any of the numbers in the output matched the manually classified data. If a match was found, we will consider the result to be True, otherwise False. While this could potentially introduce some False Positives, it also does not punish the model for our chosen implementation of evaluation or the restricted way of displaying the results, which is in no way set in stone for real-world use cases.

- **Precision** – The fraction of relevant retrieved instances among all retrieved instances.

$$\frac{\text{True Positives}}{\text{True Positives} \ + \ \text{False Positives}} \tag{1}$$

- **Recall** – The fraction of relevant retrieved instances among all relevant instances.

$$\frac{\text{True Positives}}{\text{True Positives} \ + \ \text{False Negatives}} \tag{2}$$

- **F1-Score:** The harmonic mean of Precision and Recall.

$$2 \cdot \frac{\text{Precision} \ \cdot \ \text{Recall}}{\text{Precision} \ + \ \text{Recall}} \tag{3}$$

- **Macro Average:** Arithmetic mean of all the values for the different classes.

$$\frac{\sum_{i=1}^{n} x_i}{n} \tag{4}$$

  - x_i: Each individual value.

- **Weighted Average:** Arithmetic mean that considers the relative importance or significance of different values in a dataset by assigning specific weights to each value.

$$\frac{\sum_{i=1}^{n} \omega_i \cdot x_i}{\sum_{i=1}^{n} \omega_i} \tag{5}$$

– n: Number of values.
– ω_i: Weight corresponding to each value.
– x_i: Each individual value.

- **Support:** Number of values in each class.

In addition to these metrics, we also have a visualized heatmap of the confusion matrix. Its purpose is providing an improved understanding of the tendency of models to make mistakes or the opposite – showcase the volume of correct classifications. The next section contains a table with the performance metrics and a visualization of the confusion matrix. Since those include data for all the models we tested, the values are an average (Fig. 1 and Table 3).

Table 3. Performance Metrics (Average for the eleven chosen models)

	Precision	Recall	F1-Score	Support
0	0.76	0.65	0.67	323
1	0.65	0.69	0.59	189
Accuracy			0.67	512
Macro Avg.	0.71	0.67	0.63	512
Weighted Avg.	0.72	0.68	0.64	512

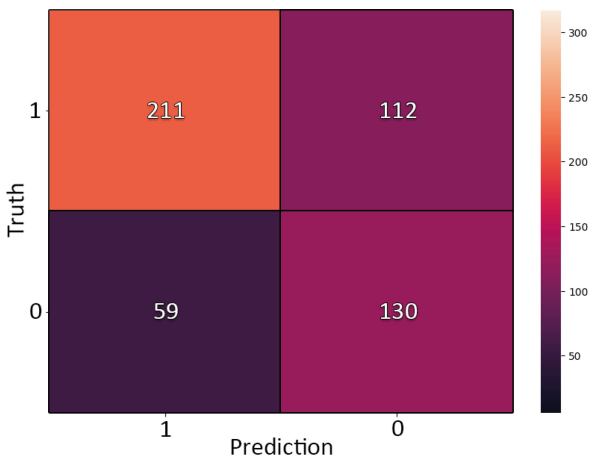

Fig. 1. Confusion Matrix (Average for the eleven chosen models)

Final Comparison

We can observe below a comparison of the F1-Scores of the different models (Figs. 2, 3 and 4).

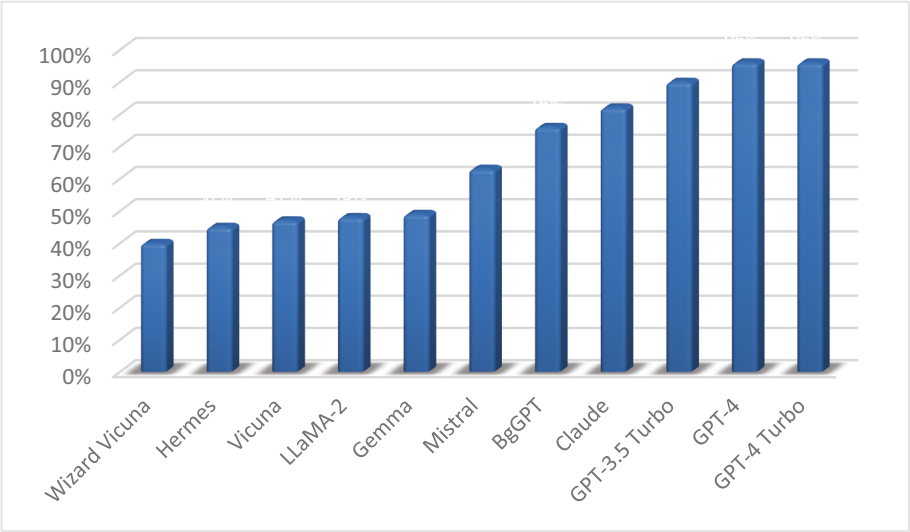

Fig. 2. F1-Scores Accuracy (Percentage)

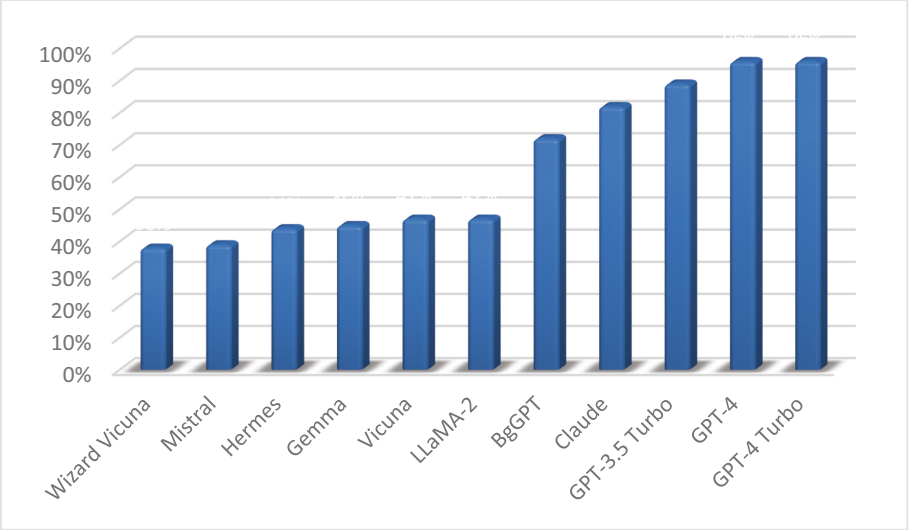

Fig. 3. F1-Scores Macro Average (Percentage)

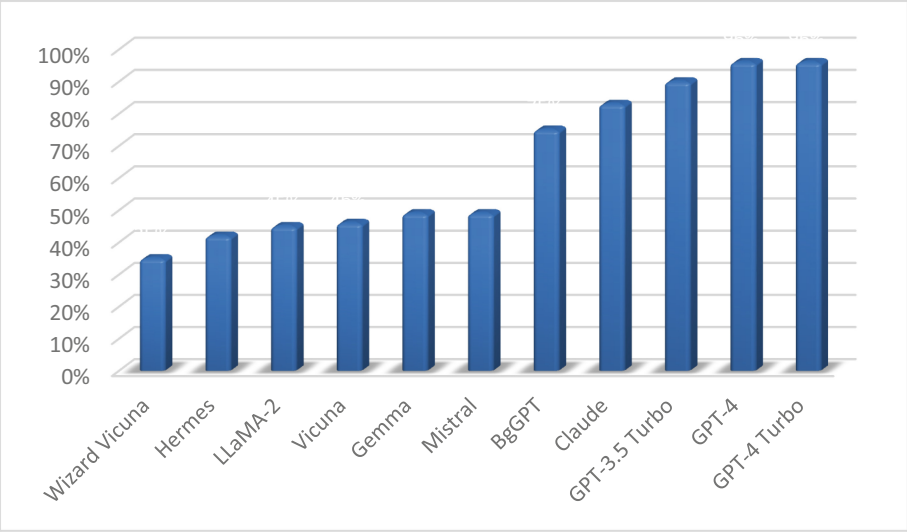

Fig. 4. F1-Scores Weighted Average (Percentage)

4 Conclusions

The task of extracting medical data from unstructured records is complex and challenging. In the general case, regular expressions are no match for complexity and usage of LLM is necessary. We managed to develop a method for systematical extraction of medical data in a particular case, related to diabetes and smoking status and we compared multiple models. We discovered some models that have results satisfactory for our very high level of desired precision, such as GPT-4.5 turbo. This performance is comparable of higher than with more traditional approaches for the same dataset, that require structuring of data and generating xml schemes [31, 32]. Our next challenge is to find smaller and free of charge models, that can run on a local machine under the cost of 5000$ with a query per second as a speed. Our data cannot be uploaded and shared with the owners of the models, se we need to run a local instance and to run it efficiently and cheaply enough for our budget. This is the goal for the next phase of our research.

Acknowledgement. This study is supported by the Strategic fund of New Bulgarian University.

This study is financed by the European Union-NextGenerationEU, through the National Recovery and Resilience Plan of the Republic of Bulgaria, project № BG-RRP-2.004–0008.

References

1. Petkov, P.: Extraction of Health and Socio-Economic Indicators from Medical Text and Analysis of the Results. Bachelor thesis, New Bulgarian University (2024). https://u.pcloud.link/publink/show?code=XZwKhe0ZeC9g3Pl3ngp9ae42o8tiNmSqlsfk
2. Large Language Model. https://en.wikipedia.org/wiki/Large_language_model#List

3. Rogers, A., Kovaleva, O., Rumshisky, A.: A primer in BERTology: what we know about How BERT Works. Transactions of the Association for Computational Linguistics **8**, 842–866 (2020). arXiv:2002.12327
4. Wu, Y., et al.: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation (2016). arXiv:1609.08144v2
5. Zhu, Y., et al.: Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books (2015). arXiv:1506.06724
6. Summary of the models —transformers 3.4.0 documentation. https://huggingface.co/transformers/v3.4.0/model_summary.html
7. Patel, A., et al.: Bidirectional Language Models Are Also Few-shot Learners (2022). arXiv:2209.14500 [cs.LG]
8. Touvron, H., et al.: LLaMA: Open and Efficient Foundation Language Models (2023). arXiv:2302.1397138
9. Touvron, H.T., Martin, L., et al.: LLaMA-2: Open Foundation and Fine-Tuned Chat Models (2023). arXiv:2307.09288
10. Edwards, B.: Meta Launches LLaMA-2, a Source-Available AI Model that Allows Commercial Applications [Updated] (2023). Ars Technica
11. Gerganov, G.: https://github.com/ggerganov/llama.cpp
12. Lin, J., et al.: AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration (2023). arXiv:2306.00978v2
13. Contextual Temperature for Language Modeling. https://openreview.net/pdf?id=H1x9004YPr
14. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/
15. WizardVicunaLM. https://github.com/melodysdreamj/WizardVicunaLM
16. Nous-Hermes-13b. https://huggingface.co/NousResearch/Nous-Hermes-13b
17. Deng, L.: A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing | Cambridge Core. Apsipa Transactions on Signal and Information Processing. Cambridge.org. **3**, e2 (2014)
18. Vaswani, A., et al.: Attention is all you need. (PDF). Advances in Neural Information Processing Systems. Curran Associates, Inc. **30** (2017)
19. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving Language Understanding by Generative Pre-Training (PDF). OpenAI. p. 12 (2018)
20. Piper, K.: An AI Helped us Write this Article (2019). Vox
21. Vincent, J.: OpenAI's New Multitalented AI Writes, Translates, and Slanders (2019). The Verge
22. Brown, T.B.et al.: Language Models are Few-Shot Learners (2020). arXiv:2005.14165
23. Wiggers, K.: OpenAI Releases GPT-4, a Multimodal AI that it Claims is State-of-the-Art (2023). TechCrunch
24. Schreiner, M.: GPT-4 Architecture, Datasets, Costs and More Leaked (2023). THE DECODER
25. BgGPT-7B-Instruct-v0.1. https://huggingface.co/INSAIT-Institute/BgGPT-7B-Instruct-v0.1
26. Bai, Y., et al.: Constitutional AI: Harmlessness from AI Feedback (2022). arXiv:2212.08073
27. Edwards, B.: Anthropic's Claude 3 Causes Stir by Seeming to Realize When it was being Tested (2024). Ars Technica
28. Gemma: Introducing New State-of-the-Art Open Models. https://blog.google/technology/developers/gemma-open-models/
29. Announcing Mistral 7B. https://mistral.ai/news/announcing-mistral-7b/
30. Mixtral 8x7B: What You Need to know about Mistral AI's Latest Model. https://medium.com/@raouf.chebri/mixtral-8x7b-what-you-need-to-know-about-mistral-ais-latest-model-1f95b8dd8ebe

31. Nikolova, I., Tcharaktchiev, D., Boytcheva, S., Angelov, Z., Angelova, G.: Applying language technologies on healthcare patient records for better treatment of bulgarian diabetic patients. In: Agre, G., Hitzler, P., Krisnadhi, A.A., Kuznetsov, S.O. (eds.) Artificial Intelligence: Methodology, Systems, and Applications. AIMSA 2014. Lecture Notes in Computer Science, **8722**. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10554-3_9
32. Nikolova, I., Boytcheva, S., Angelova, G., Angelov, Z.: Combining structured and free textual data of diabetic patients' smoking status. In: Dichev, C., Agre, G. (eds.) Artificial Intelligence: Methodology, Systems, and Applications. AIMSA 2016. Lecture Notes in Computer Science, **9883**. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44748-3_6