Large Language Models for identification of medical data in unstructured records

Presian Petkov[1][0009-0000-4956-2600], Emanuil Markov[2][], Latchezar Tomov[3][0000-0003-1902-6473]

[1] Graduate Student, B.IT., New Bulgarian University, 1618, Sofia, Bulgaria
npecko@protonmail.com
[2] Senior statistical programmer, IQVIA Solutions Bulgaria Ltd., Sofia, emospy@yahoo.com
[3] Department of Informatics, New Bulgarian University, 1618, Sofia, Bulgaria
lptomov@nbu.bg,
Medical Faculty, Sofia University St. Kliment Ohridski, 1 Kozyak Str., 1407 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 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. Popular Large Language Models

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

BERT

Bidirectional Encoder Representations from Transformers (BERT) is a language model, introduced in 2018 by researchers from 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]

It uses *WordPiece* [4] to convert each English word into an integer code. Its vocabulary has a size of 30,000. Any token that does not appear in its vocabulary is replaced with [UNK] (unknown). For pre-training, the Toronto BookCorpus [5] (800M words) and the English Wikipedia (2,500M words) were used. BERT was pre-trained simultaneously on two tasks. Language modeling, where 15% of the tokens were selected for prediction, and the training objective was to predict the selected token given its context. And next sentence prediction – given two spans of text, the model predicts if these two spans 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 be prompted and cannot generate text, while bidirectional models in general do not work effectively without the right side, thus being difficult to prompt, with even short text generation requiring sophisticated computationally expensive techniques [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

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

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 accuracy and logarithmic loss[2] that evaluate the model's performance on the remainder of the data 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 – 16GB, HDD; Total training time – 7 minutes). 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

Four models were trained for the first version of LLaMA. 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.

On July 18th, 2023, in partnership with Microsoft, Meta announced LLaMA-2, the next generation of the model. It comes in three sizes – 7, 13, and 70 billion parameters. The architecture remains largely unchanged, 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 for commercial use[3]. Despite this, due to some active limitations, the description of LLaMA as "being open-source" has been disputed by the Open-Source Initiative, who are in charge of maintaining the "Open-Source Definition". [10]

[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/

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 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, collected through *ShareGPT*. Preliminary evaluations using GPT-4 as a judge show that Vicuna-13B achieves more than 90% the quality of OpenAI's ChatGPT and Google's Bard, while simultaneously performing 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 WizardLM handling the dataset itself more deeply and broadly (given the fine-tuning and additional context from chat-type user interactions), as well as VicunaLM overcoming the limitations of single-turn conversations by introducing multi-round conversations. A 7% performance increase over VicunaLM is reported. [15]

Hermes is a state-of-the-art language model fine-tuned on over 300,000 instructions that was trained almost entirely on synthetic GPT-4 outputs. The result is an enhanced LLaMA-13B model that rivals GPT-3.5 Turbo in performance across a variety of tasks. This model stands out for its long responses, low hallucination rate, and absence of censorship mechanisms. [16]

GPT

Generative pre-training (GP) was a well-established concept in applications with machine learning [17]. In 2017, researchers at Google published a paper called "Attention Is All You Need", that featured the modern transformer architecture. [18] This advance led to BERT and XLNet, which are pre-trained transformers (PT), but were not developed to be generative (as they only have an encoder). In 2018, OpenAI published an article titled "Improving Language Understanding by Generative Pre-training", in which the first system with a generative pre-trained transformer – GPT-1 was introduced. [19] The successor GPT-2 has more parameters by a factor of 10 (1.5 billion), also pre-trained on BookCorpus and trained on a dataset of eight million web pages. 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 similarly noted that longer samples of GPT-2 writing tended to "stray off topic" and lack overall coherence. [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, a group of 31 engineers and researchers from OpenAI describe their accomplishments on the development of GPT-3. [22] GPT-4 was launched on the 14th of March 2023. OpenAI stated that GPT-4 is “more reliable, creative, and able to handle much more nuanced instructions than GPT-3.5”. [23] While undisclosed by OpenAI, based on the speed at which it was running and by George Hotz, it was estimated that GPT-4 has 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.

In order 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, i.e., the *content*, which in actuality is the generated text. The 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:”. To conclude, we limit the maximum number of output tokens 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.

It is fine-tuned to improve its Bulgarian language capabilities using multiple datasets, including Bulgarian web crawl data, a range of specialized Bulgarian datasets sourced by INSAIT Institute, and machine translations of popular English datasets. This Bulgarian data was augmented with English datasets to retain English and logical reasoning skills.

The model’s tokenizer has been extended to allow for a more efficient encoding of Bulgarian words written in Cyrillic. This not only increases throughput of Cyrillic text but also 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 is a family of large language models developed by Anthropic. Claude models are generative pre-trained transformers and have then been fine-tuned with “Constitutional AI” and Reinforcement Learning from Human Feedback.

Constitutional AI is an approach developed by Anthropic for training AI systems, particularly language models like Claude, to be harmless and helpful without relying on extensive human feedback. The method, detailed in the paper "Constitutional AI: Harmlessness from AI Feedback" involves two phases: supervised learning and reinforcement learning. In the supervised learning phase, the model generates responses to prompts, self-critiques these responses based on a set of guiding principles (a "constitution"), and then revises the responses. The reinforcement learning phase involves training the model with AI-generated feedback, where the AI evaluates responses according to the constitutional principles. This approach enables the training of AI assistants that are both helpful and harmless, and that can explain their objections to harmful requests, enhancing transparency and reducing reliance on human supervision. The "constitution" for Claude included 75 points, including sections from the UN Universal Declaration of Human Rights. [26]

The model demonstrated proficiency in various tasks but had certain limitations in coding, math, and reasoning capabilities. It was released as two versions, Claude and Claude Instant, the latter being a faster, less expensive, and lighter version with an input context length of 100,000 tokens. Claude 2 was the next major iteration. It was available to the public, unlike the first ones, which were only available to selected users, approved by Anthropic. It expanded the context window, and offered the ability to upload documents that Claude can process. Claude 2 faced criticism for its stringent ethical alignment that reduced usability. Users were refused assistance with requests, deemed benign, i.e., programming inquiries. With Claude 2.1, the number of tokens the chatbot could handle was increased to 200,000. 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, including the ability to realize it is being artificially tested during needle in haystack 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 is a family of lightweight, state-of-the-art open models built from the same research and technology used to create the Gemini models. Developed by Google DeepMind and other teams across Google, Gemma is inspired by Gemini, and the name reflects the Latin gemma, meaning “precious stone.”

Model weights 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 size is released with pretrained and instruction-tuned variants. 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, similar to a chat bot. It is reported that Gemma surpasses significantly larger models on key 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 model to be trained with an 8k context length and a fixed cache size, providing a theoretical attention span of 128k tokens.
- Grouped Query Attention – Enables faster inference and a smaller cache size.
- Byte-fallback BPE tokenizer – Ensures that characters are never mapped to out-of-vocabulary tokens.

The model has demonstrated superior performance compared to other models 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 approximately 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 a compelling alternative that balances cost, accessibility, and robust AI 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.

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*
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. 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. 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 the fields of 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 Positives and False Negatives are considered Type I and Type II errors, respectively.

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 to provide a better understanding of what is the tendency of the model 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 for the models we tested. The values shown are an average taken from all models.

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

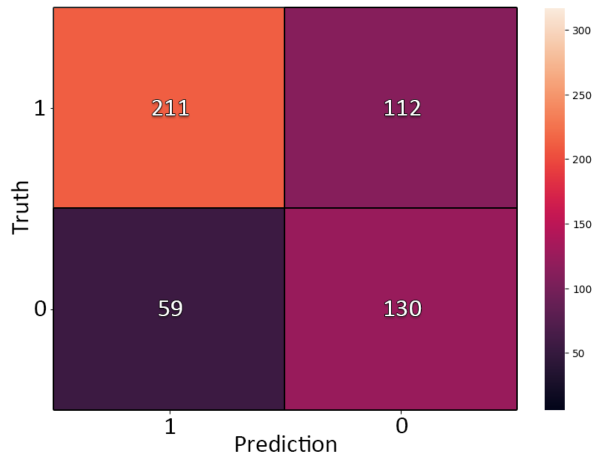

Fig. 1. Confusion Matrix (Average for the eleven chosen models)

Final comparison

We can observe below a comparison of the F1-Scores for the accuracy of the different models.

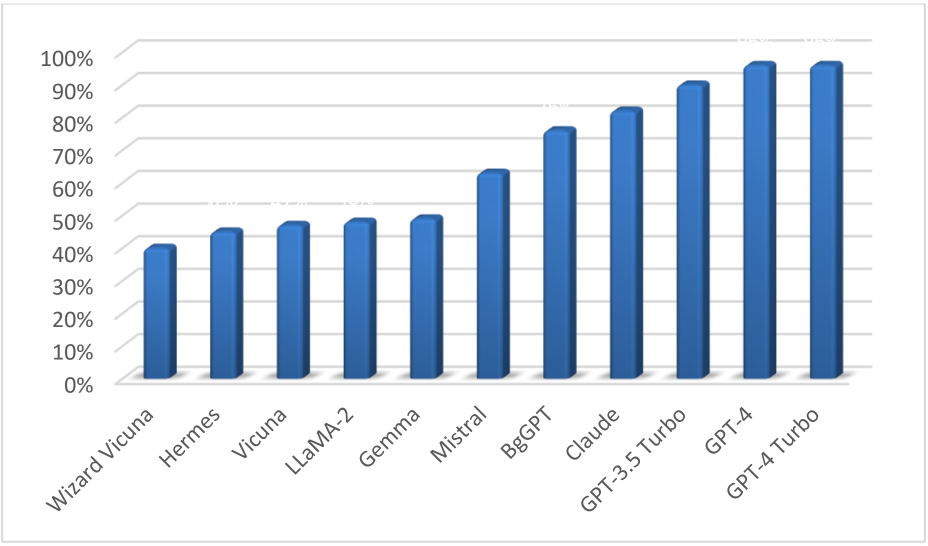

Fig. 2. F1-Scores Accuracy (Percentage)

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 the 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. 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 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).
2. “Large Language Model”, https://en.wikipedia.org/wiki/Large_language_model#List

3. Rogers, Anna; Kovaleva, Olga; Rumshisky, Anna (2020). "A Primer in BERTology: What We Know About How BERT Works". Transactions of the Association for Computational Linguistics. 8: 842–866. arXiv:2002.12327
4. Wu, Yonghui; Schuster, Mike; Chen, Zhifeng; V. Le, Quoc; Norouzi, Mohammad; Macherey, Wolfgang; Krikun, Maxim; Cao, Yuan; Gao, Qin; Macherey, Klaus; Klingner, Jeff; Shah, Apurva; Johnson, Melvin; Liu, Xiaobing; Kaiser, Łukasz; Gouws, Stephan; Kato, Yoshikiyo; Kudo, Taku; Kazawa, Hideto; Stevens, Keith; Kurian, George; Patil, Nishant; Wang, Wei; Young, Cliff; Smith, Jason; Riesa, Jason; Rudnick, Alex; Vinyals, Oriol; Corrado, Greg; Hughes, Macduff; Dean, Jeffrey (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". arXiv:1609.08144v2
5. Zhu, Yukun; Kiros, Ryan; Zemel, Rich; Salakhutdinov, Ruslan; Urtasun, Raquel; Torralba, Antonio; Fidler, Sanja (2015). "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books". 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, Ajay; Li, Bryan; Mohammad Sadegh Rasooli; Constant, Noah; Raffel, Colin; Callison-Burch, Chris (2022). "Bidirectional Language Models Are Also Few-shot Learners". arXiv:2209.14500 [cs.LG].
8. Touvron, Hugo; Lavril, Thibaut; Izacard, Gautier; Martinet, Xavier; Lachaux, Marie-Anne; Lacroix, Timothée; Rozière, Baptiste; Goyal, Naman; Hambro, Eric; Azhar, Faisal; Rodriguez, Aurelien; Joulin, Armand; Grave, Edouard; Lample, Guillaume (2023). "LLaMA: Open and Efficient Foundation Language Models". arXiv:2302.1397138
9. Touvron, Hugo Touvron; Martin, Louis; et al. (2023). "LLaMA-2: Open Foundation and Fine-Tuned Chat Models". arXiv:2307.09288
10. Edwards, Benj (July 18, 2023). "Meta launches LLaMA-2, a source-available AI model that allows commercial applications [Updated]". Ars Technica.
11. Gerganov, Georgi, https://github.com/ggerganov/llama.cpp
12. Lin, Ji; Tang, Jiaming; Tang, Haotian; Yang, Shang; Dang, Xingyu; Gan, Chuang; Han Song (2023). "AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration". 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, Li (January 22, 2014). "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.
18. Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz; Polosukhin, Illia (2017). "Attention is All you Need" (PDF). Advances in Neural Information Processing Systems. Curran Associates, Inc. 30.
19. Radford, Alec; Narasimhan, Karthik; Salimans, Tim; Sutskever, Ilya (June 11, 2018). "Improving Language Understanding by Generative Pre-Training" (PDF). OpenAI. p. 12.
20. Piper, Kelsey (February 14, 2019). "An AI helped us write this article". Vox.
21. Vincent, James (February 14, 2019). "OpenAI's new multitalented AI writes, translates, and slanders". The Verge

22. Brown, Tom B.; Mann, Benjamin; Ryder, Nick; Subbiah, Melanie; Kaplan, Jared; Dhariwal, Prafulla; Neelakantan, Arvind; Shyam, Pranav; Sastry, Girish; Askell, Amanda; Agarwal, Sandhini; Herbert-Voss, Ariel; Krueger, Gretchen; Henighan, Tom; Child, Rewon; Ramesh, Aditya; Ziegler, Daniel M.; Wu, Jeffrey; Winter, Clemens; Hesse, Christopher; Chen, Mark; Sigler, Eric; Litwin, Mateusz; Gray, Scott; Chess, Benjamin; Clark, Jack; Berner, Christopher; McCandlish, Sam; Radford, Alec; Sutskever, Ilya; Amodei, Dario (May 28, 2020). "Language Models are Few-Shot Learners". arXiv:2005.14165.
23. Wiggers, Kyle (March 14, 2023). "OpenAI releases GPT-4, a multimodal AI that it claims is state-of-the-art". TechCrunch.
24. Schreiner, Maximilian (July 11, 2023). "GPT-4 architecture, datasets, costs and more leaked". THE DECODER.
25. “BgGPT-7B-Instruct-v0.1”, https://huggingface.co/INSAIT-Institute/BgGPT-7B-Instruct-v0.1
26. Bai, Yuntao; Kadavath, Saurav; Kundu, Sandipan; Askell, Amanda; Kernion, Jackson; Jones, Andy; Chen, Anna; Goldie, Anna; Mirhoseini, Azalia (December 15, 2022), Constitutional AI: Harmlessness from AI Feedback, arXiv:2212.08073
27. Edwards, Benj (March 5, 2024). "Anthropic's Claude 3 causes stir by seeming to realize when it was being tested". 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