Master Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Analyzing Conversation Data in the Context of Addiction Counseling

Petr Karlík

Supervisor: doc. Ing. Daniel Novák, Ph.d.
May 2024

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Karlík Petr**	Personal ID number:	**491970**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Medical Electronics and Bioinformatics**		
Specialisation:	**Image processing**		

II. Master's thesis details

Master's thesis title in English:

Analyzing Conversation Data in the Context of Addiction Counseling

Master's thesis title in Czech:

Zpracování dialogových dat z oblasti poradenské praxe pro závislé

Guidelines:

The goal is to develop an assistance system tailored for social service personnel, enhancing their online communication efficacy with clients.
1. Conduct a comprehensive review of the current state of the art in Named Entity Recognition (NER) and Retrieval Augmentation methods within the domain of Natural Language Processing.
2. Identify socio-demographic and clinical entities within a database of conversational dialogues, focusing on extracting relevant information that can provide insights into the social and health-related characteristics of individuals engaged in these conversations.
3. Evaluate the performance of the proposed techniques by analyzing their accuracy, processing time, and the amount of computational resources they require.

Bibliography / sources:

[1] Raza S, Reji DJ, Shajan F, Bashir SR. Large-scale application of named entity recognition to biomedicine and epidemiology. PLOS Digit Health. 2022
[2] Wang, C., Ong, J., Wang, C. et al. Potential for GPT Technology to Optimize Future Clinical Decision-Making Using Retrieval-Augmented Generation. Ann Biomed Eng, 2023
[3] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Name and workplace of master's thesis supervisor:

doc. Ing. Daniel Novák, Ph.D.   Analysis and Interpretation of Biomedical Data FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **31.01.2024**   Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

doc. Ing. Daniel Novák, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to thank my supervisor, doc. Ing. Daniel Novák, Ph.D., for his guidance, valuable advice and patience. My thanks also belongs to the whole research team as their advice and remarks were greatly appreciated, following: Cheng Kang, MSc., Bc. Fabián Bodnár, Ing. Patrik Jankuv, Bc. Štěpán Bořek, Petr Stádník,Klára Losenická, Varvara Shorina.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, May 24, 2024

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 24. května 2024

Abstract

This work is a part of an ongoing platform development originally for national quitting line but now extended for general counseling. The main focus is on finding solution for automatization of arbitrary data extraction from within The dialogoues collected by The platform. Since this task falls within The many problems of natural language processing I first provide a general overview of NLP methods arguing The importance of neural nets. Subsequently I introduce key NN architectures which pioneered NLP. Following I present a summary of LLMs and choose six of these models to be tested within The NQL data framework. A manual selection of 40 dialogues is made from the NQL data, these samples are chosen based on the potential field information they contain and on their diarization and transcription quality, basic errors are manually corrected. Given The unlabelled nature of The data an unsupervised evaluation pipeline based on word similarity is proposed and used. Based on this evaluation Llama3-70B is argument to be The best performing model. With respect to computational efficiency Gemma-7B is The best performer.

Keywords: LLM, transformers, RNN, National quitting line, addictology, counseling, cosine similarity

Supervisor: doc. Ing. Daniel Novák, Ph.d.

Abstrakt

Tato práce je součástí probíhajícího vývoje platformy, která byla původně určena pro národní linku pro odvykání, ale nyní byla rozšířena pro obecné poradenství. Hlavní zaměření je na nalezení řešení pro automatizaci libovolné extrakce dat z dialogů shromážděných platformou. Protože tento úkol spadá do mnoha problémů zpracování přirozeného jazyka, nejprve poskytuji obecný přehled metod NLP a zdůrazňuji důležitost neuronových sítí. Následně představím klíčové architektury neuronových sítí, které byly průkopníky v NLP. Poté prezentuji souhrn LLM a vybírám šest z těchto modelů, které budou testovány v rámci datového prostředí NQL. Z dat NQL je manuálně vybráno 40 dialogů, tyto vzorky jsou vybrány na základě potenciálních informačních polí, která obsahují, a na základě kvality diarizace a transkripce; základní chyby jsou ručně opraveny. Vzhledem k nelabelované povaze dat je navržen a použit nesupervidovaný hodnotící proces založený na podobnosti slov. Na základě tohoto hodnocení je argumentováno, že model Llama3-70B je nejlépe fungující. S ohledem na výpočetní efektivitu je Gemma-7B nejlepším modelem.

Klíčová slova: LLM, transformery, RNN, Národní linka pro odvykání, adiktologie, kosinová vzdálenost

Překlad názvu: Zpracování dialogových dat z oblasti poradenské praxe pro závislé

Contents

Figures

Tables

Chapter 1
Introduction

This project is a part of an ongoing collaboration between the CTU and czech national qutting line (NQL). The cooperation started with the NQL's need for new advanced system to help their patients. However now it scaled beyond just the NQL system development. The aim is to develop an efficient platform powered by the state-of-the-art technology and help provide support for service lines anywhere.

With the recent growth in drug use [1] the time staff can use for targeted care growths more important. Unfortunately with this also comes more administration workload which needs to be processed by the respective workers. The recent boom of LLM systems[2] begs the question if some of these tasks couldn't be automated. We focus on identification and automatization of these tasks while keeping the solutions scalable and incorporatable into the NQL system.

This work aims at automatization of data extraction from the dialogues collected on the platform. This will greatly help to alleviate redundant rewriting job from the consultants thus allowing them to better tend to the patients.

Our approach involves using mainly pretrained LLMs and fine-tuning them to recognize entities that are crucial to achieve more effective support for NQL patients. We present a pipe-line for unsupervised evaluation on the unlabelled data gather by NQL.

The aim of this work is to provide an informed selection recommendation on a system usage to achieve the above mentioned extraction of client data.

Chapter 2

The platform under development

In this chapter we briefly introduce the system being developed for the NQL. We present its most important functionalities and in greater detail present those directly impacted by this work.

2.1 Motivation behind platform development

To fully understand the improvement a modern platform would bring we first should examine the problem the NQL and other similar agencies are trying to solve. While some studies suggest small positive effects of few drugs namely psylocybin and cannabis. Pilot studies were done and while suggesting some positive effects might be present, but generally authors agree further study in the field is recommended [3],[4],[5].

While the positives of certain drug usage are in the phase of research the negative impacts are fairly well studied from both medical and statistical point of view. Reviewing effects of drugs from medical perspective is beyond the scope of both this work and my education. To prove the above statement we refer the reader to these studies presenting the negative effects of drugs from medical side: Meta analysis of different effects of methamphetamine[6], cocaine toxicity:[7], amphetamines toxicity:[8].

Let's start with statistics on alcohol as it is one of the most common drugs. With the COVID pandemic it is warranted to ask for its effects on alcohol consumption. Studies in USA and Germany suggest an increase in alcohol consumption between adults [9], [10], [11]. Eventhough the extrapolation of studies done in USA may be incorrect, the fact that when we compare consumption per capita in Europe countries vs US the higher consumption in Europe is indisputable. In 2019 in the top 20 alcohol consumers only 2 countries we not European [12]. These fact further justify the extrapolation.

Further looking at figure 2.1 suggest an increase with covid years also, although this consumption growth is even more noticeable when inspecting data from USA 2.2. I use the above mentioned facts as a backbone to say the effect of the COVID pandemic on alcohol consumption was at most not positive, the data suggest the effect was mainly negative even-though statistics from years to come will clarify.

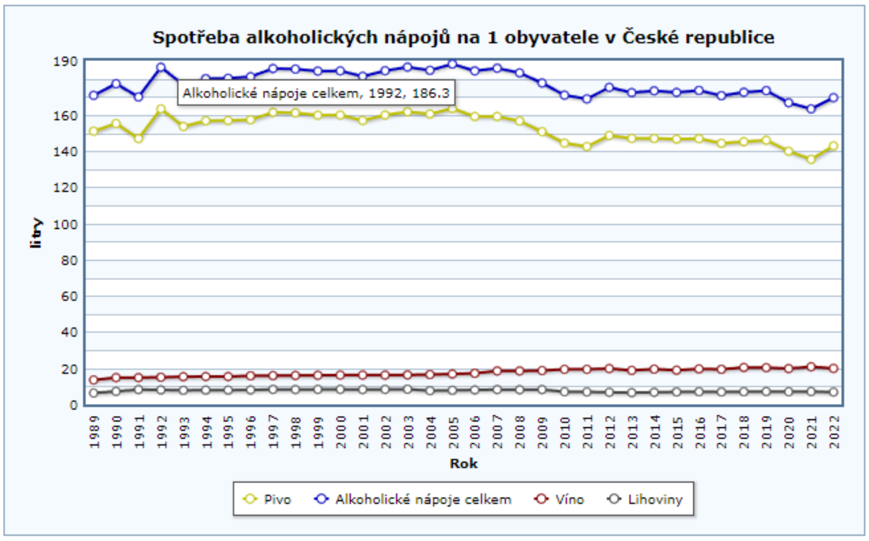


Figure 2.1: Evolution of alcohol consumption in CZ. Image source: [13]

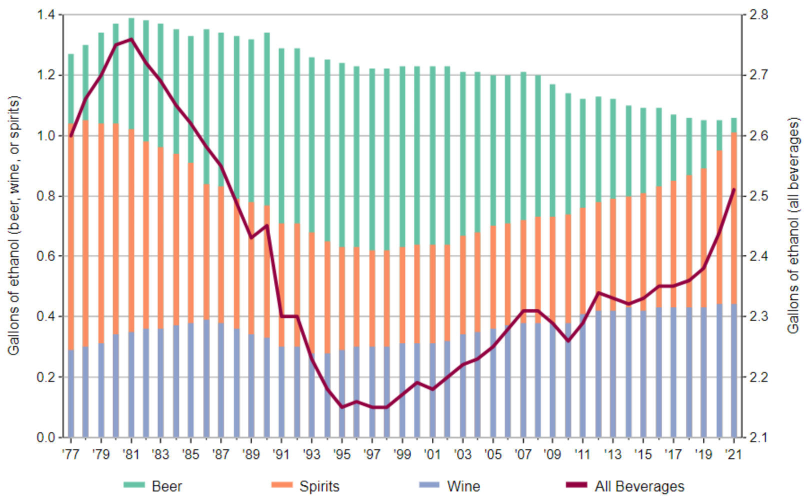
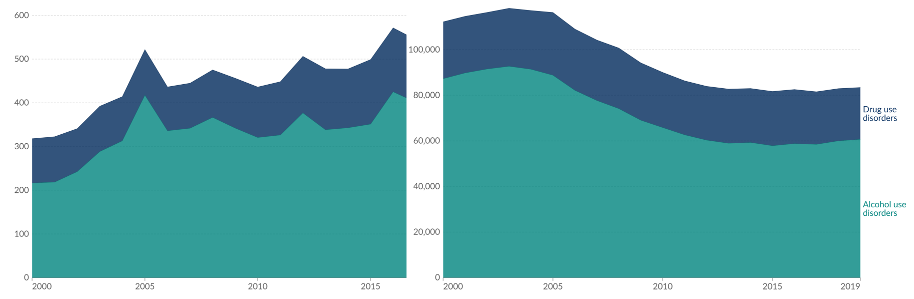


Figure 2.2: Alcohol consumption through years in USA. Image source: [14]

Further diving into the data in Czech republic we can observe the amount of alcohol and drug induced deaths through the past years in figure 2.3. In Europe the trend was generally a decrease in deaths with slowing/stopping nature in the past years, while in CZ the pattern is inverse, with the amount of deaths having a growing nature. This is especially troublesome given that the quality of healthcare in Czech republic steadily rises [15].

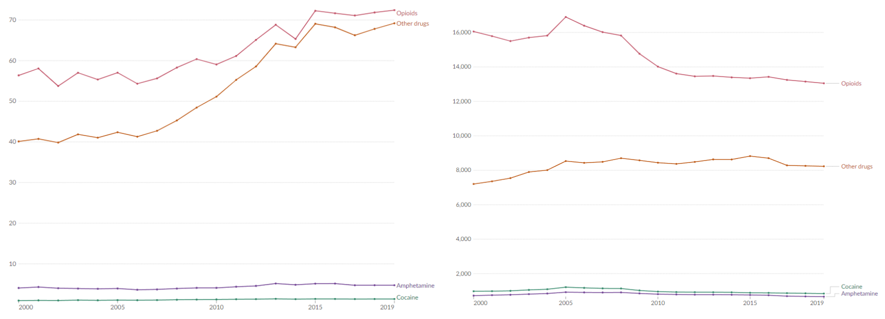
These results may seem confusing give figure 2.1 shows a relatively steady amount of alcohol consumed. So where do the deaths come from when healthcare is better and the consumption is the same. Since we can safely reason the amount of alcohol consumed to induce death threatening states is quite high the explanation I provide is a shift of consumption load from casual drinkers to those chronically addicted. This furthers the need for an intervention and subsequently bettering of the services provided by the national quitting line.



(a) : Czech republic **(b) :** Europe

Figure 2.3: Comparison of alcohol and drug induced deaths in Czech republic and Europe. Image source:[16]

An investigation into drugs purely yields the same results. As figure 2.4 suggests the trend in drug induced deaths in CZ slowly increases in the past years while the trend in Europe is steady in past years. Looking further back the patterns between these two seem to be even more inverse, Europe going down and CZ up.



(a) : Czech republic **(b) :** Europe

Figure 2.4: Comparison of drug induced deaths in Czech republic and Europe. Image source:[16]

On the other hand looking at the second most common drug, smoking. We witness a positive trend where both Europes and Czech republics deaths induced by smoking steadily decrease as we see in figure 2.5.

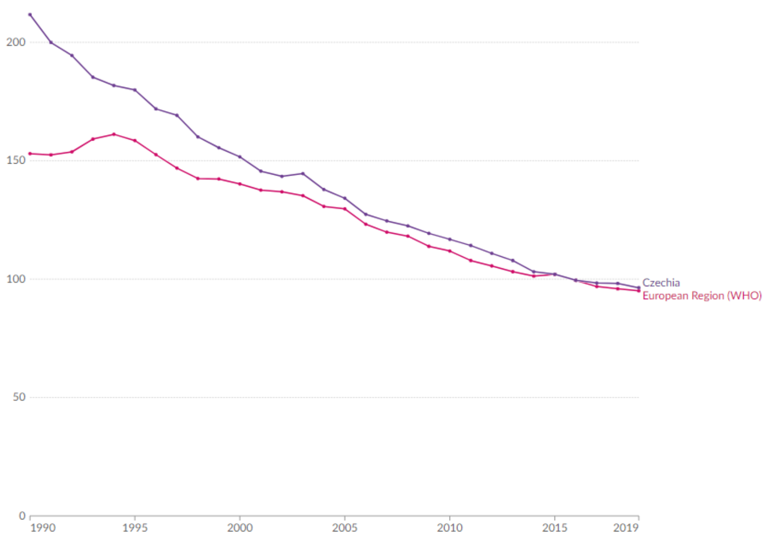


Figure 2.5: Smoking induced deaths per 100,000 people in EU and CZ. Image source:[17]

Having investigated both the positive and negative effects of drugs it is safe to assume drugs are harmful for most people in both social and physical aspects. These findings truly justify the need for improvement in accessibility of drug quitting help. Even-though this tendency is promising, its presence is only due to continuous work from governments and agencies like NQL.

Cigarette addiction is still a big topic and we have to keep in mind that proving the causality of death being induced purely by smoking can be hard even with modern approaches. The correlation between leading death causes [18] like hearth disease, cancer and smoking is undisputed [19], [20].

NQL main communication method is phone calls. This comes with many disadvantages. The anonymity of the caller is partially compromised due to their voice being known. Also studies [21],[22] show that texting/online-chatting is very popular especially between teenagers and young adults. The need to catch up to this trend grows with each day, forcing NQL to act, subsequently leading to the above mentioned cooperation.

2.2 General platform capabilities

The main idea is to provide a client management system with an integrated chatting platform. This platform is developed in a scalable manner providing configurable features so not only NQL can use it. The system should be capable of the following:

- Keeping client profile
- Allowing anonymous access to chatting (chatting without profile)

- Scheduling of consultations
- Allow gathering of custom data predefined by the operator (such as NQL)
- Managing the operator roles and assigning clients to them
- Generate summaries of chats

Some of these key capabilities will be powered by machine learning. In this work I mainly focus on making the data gathering procedure more efficient, trying to automate as much data collection as much as possible to relief operator workload. In the next section I dive more into how exactly this should be achieved.

2.3 Automating data gathering

In order to fulfill the patient card data like name, address, phone number etc. have to be gathered. NQL also needs to gather some more advanced data such as type of addiction, used medicament's, frequency of using, general health data (weight, height, any long-term illnesses). In this project I mainly focus on the easier data as these not only don't need such precision but also are very tedious for the operators to gather while the automatization seems reasonably difficult to achieve.

The main source of this data would be the chatting platform where we can scan each message and decide if it contains any valuable information and if it does we can either mark it for the operator so they can fill the respective field with one click not having to write anything or even better if our algorithm is good enough fill the card directly without any human effort. This problem can be classified as Named entity recognition(NER). NER tasks are generally performed to extract data with direct format like Name, Date, Adress etc. these entities are easily markable within the given text. Our aim is to be able to fill fields with a more abstract format like, financial situation, medical history, relationship status these fields cannot be directly marked within the text but have to be inferred from the context. This brings many difficulties with both design and evaluation of the proposed solutions.

2.4 Summary generation

Two types of summaries are useful:

- Summary of general state.
- Summary of each session.

Summary of general state grossly simplifies the transfer between operators. Each patient has its own operator but it might happen that their operator is not able to conduct a session to ensure quality of the session the new operator has to gather information from the previous operator. A summary of the patients current health and progress could alleviate the need for this transfer as the new operator could just read the summary.

Since every operator handles multiple patients thus having many consultations per day, sometimes without a break it is very helpful for them to have a session summary so they can better remember what was discussed. Summary also helps with the preparation for the next consultation.

2.4.1 How to do it?

In order for the summary to be useful the generator should have access to data relevant to the field. In our case addictology and general medicine since the patient's health state is very important. As a generator we use a LLM namely Mistral to achieve the data relevancy we use retrieval-augment generation (RAG). RAG is a method to augment response of a model with data from a vectorized database. RAG is also used to provide a platform of knowledge for the user where it is possible to query the model with questions to generate responses which are augmented by the data included in the RAG database. The user will also be able to add specialized data to the database to achieve higher relevancy of the responses to their given field of operation. I won't dive into more details as this is not the task I am primarily working on.

Chapter 3

Natural language processing approaches

Since most of the work done within this thesis utilizes some kind of neural network I delegate a big part of the chapter to talk about those. First we dive into the core building blocks of virtually every architecture used in my thesis then I present an introduction to several critical LLM architectures. I discuss their major characteristics like size, speed, applicability. Then I provide major use cases of these architectures across multiple fields with a focus on applications in the medical industry.

3.1 Recurrent neural networks

Recurrent neural networks(RNNs) were the first attempt to handle sequential data by maintaining a memory of past states(inputs). This allows them to capture temporal relationships within the data. Among the context of natural language processing this is very important as the inherent dependency of the currently processed text on its predecessors plays a huge role in understanding the underlying semantic meaning of the data being processed.

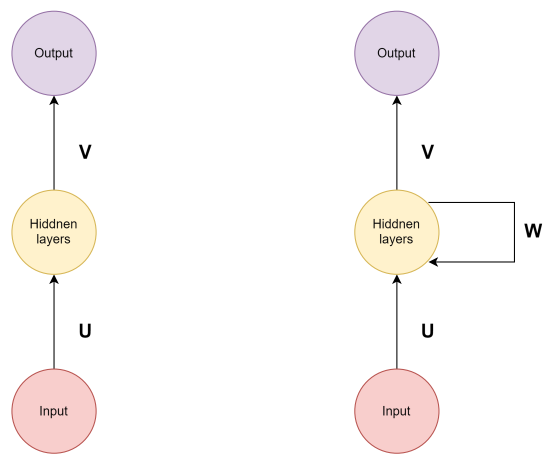
To understand how the memory sequential is incorporated into a neural net let us first compare two simple graph representations of RNN and classical feed forward network (FFN). Looking at figure 3.1 we notice both networks same three main building blocks:

1. Input layer

2. Hidden layers - commonly these multi-layer perceptron layers

3. Output layer

The only imminent difference we notice is how these blocks are connected, with RNN having one more connection (W). This is the recurrent connection and serves as the memory source for the network. In practice W is basically a backward connection between all the hidden layers.



(a) : FFN **(b) :** RNN

Figure 3.1: High level graph representations of recurrent a feed forward networks

Though the high level graph is helpful for demonstrating the main difference among the two types of network it doesn't provide a good insight towards how the sequential data is actually processed. To demonstrate this let us unpack the RNN in time. In figure 3.2 we observe this unpacking in action.

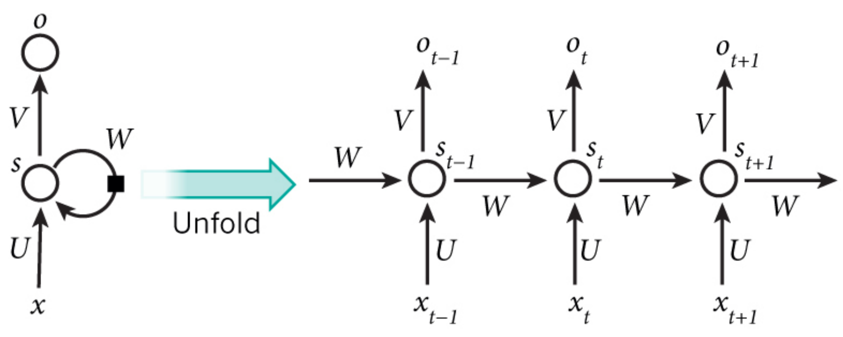


Figure 3.2: Illustration of RNN unpacking in time. Image source: [23]

We notice that we can think about RNN's as multiple feed forward networks connected in a sequential manner. While this intuition is correct the training phase of the network changes slightly due to the intersequential connection (parameters encapsulated by W), this change is especially noticeable when calculating gradient for the backward pass of backpropagation algorithm. We will discuss this in the next section while also superficially diving into the mathematics [24][25][26][27].

3.1.1 Training recurrent neural networks

As mentioned above RNN's can be learned using the widely used back-propagation algorithm [28]. Backpropagation algorithm can be fundamentally divided into two parts:

1. Forward pass
2. Backward pass

In the forward pass part the data is fed to the network to get it's output. Then a loss function is applied to calculate the performance of the network on this batch of data. Let us now formalize this in a mathematical sense for our general RNN structure:

$$s_n = f_a(W s_{n-1} + U x_n) \tag{3.1}$$

$$o_n = V s_n \tag{3.2}$$

Where:

- s_n is the state in time n
- o_n is the output the network in time n
- x_n is the input in time n
- W, U, V are the parameter matricies
- f_a is an arbitrary activation function such as *tanh* or *sigmoid*

Before we move to the backward pass we also have to define a general loss for the whole sequence. For this we annotate a loss in moment n of the sequence as $\boldsymbol{\ell}(y_n, o_n)$, ($y_n$ is the ground truth). Then the total loss is average over the sequence:

$$\mathcal{L}(y_0 ... y_N, o_0 ... o_N) = \frac{1}{N} \sum_{n=0}^{N-1} \boldsymbol{\ell}(y_n, o_n) \tag{3.3}$$

For the backward pass gradients of the loss function with respect to all parameters are needed. I will only superficially derive the general formula without any concrete activation functions as this can get quite complicated and I only want to demonstrate the general difference to the classical backpropagation.

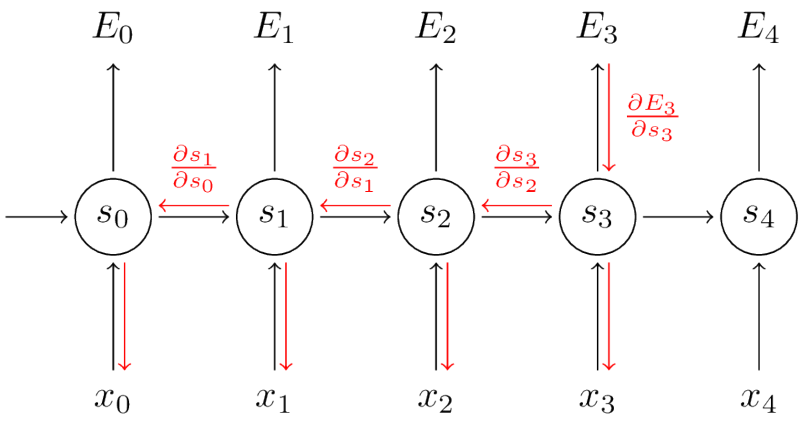
Going back to figure 3.2 we can see 3 different parameter groups (U, V, W). We will need gradient of the function $\mathcal{L}$ with respect to each one of these. Let's start with W as this might be the most complicated one. Taking the derivative of 3.3:

$$\frac{\partial \mathcal{L}(y_0 ... y_N, o_0 ... o_N)}{\partial W} = \frac{1}{N} \sum_{n=0}^{N-1} \frac{\partial \boldsymbol{\ell}(y_n, o_n)}{\partial W} \tag{3.4}$$

Using the chain rule we can rewrite the derivative of $\boldsymbol{\ell}(y_n, o_n)$ as follows:

$$\frac{\partial \boldsymbol{\ell}(y_n, o_n)}{\partial W} = \frac{\partial \boldsymbol{\ell}(y_n, o_n)}{\partial o_n} \frac{\partial o_n}{\partial s_n} \frac{\partial s_n}{\partial W} \tag{3.5}$$

Here we run into the issue since $\frac{\partial s_n}{\partial W}$ is dependent on s_{n-1} which is also dependent on W. In other words there are multiple paths to W unlike in classical FFN's. In figure 3.3 we observe 4 different paths leading to E_3.



E_0
E_1
E_2
E_3
E_4
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4

Figure 3.3: Possible gradient paths through the network graph. Image source: [29]

The chain rule for multivariable function with one independent variable says following [30]: Given three functions $g(x)$, $h(x)$ and $f(g(x), h(x))$ for the derivative of f with respect to x stands the following:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g(x)}\frac{\partial g(x)}{\partial x} + \frac{\partial f}{\partial h(x)}\frac{\partial h(x)}{\partial x} \tag{3.6}$$

Since s_n is a function of W and s_{n-1} we can define $g(W) = W$ and use 3.6:

$$\frac{\partial s_n(g(W), s_{n-1}(W))}{\partial W} = \frac{\partial s_n}{\partial g(W)}\frac{\partial g(W)}{\partial W} + \frac{\partial s_n}{\partial s_{n-1}(W)}\frac{\partial s_{n-1}(W)}{\partial W} \tag{3.7}$$

Given $g(W) = W$ we simplify 3.7:

$$\frac{\partial s_n(g(W), s_{n-1}(W))}{\partial W} = \frac{\partial s_n}{\partial W} + \frac{\partial s_n}{\partial s_{n-1}(W)}\frac{\partial s_{n-1}(W)}{\partial W} \tag{3.8}$$

It might seem confusing that the first term on the right looks like it equals the left side of the equation but note that this derivative is taken while treating s_{n-1} as a constant. Let me demonstrate on an example. Suppose:

$$x2 = Wx_1 + B$$
$$x1 = Wx_0 + B$$

First let's just substitute x_1 and calculate the derivative normally:

$$x_2 = W(Wx_0 + B) + B = W^2x_0 + WB + B \tag{3.9}$$

Thus the derivative:

$$\frac{\partial x_2}{\partial W} = 2Wx_0 + B \tag{3.10}$$

Now using 3.8 let's first calculate the needed derivatives, x_n^* means we treat the x_1 as a constant.

$$\frac{\partial x_2^*}{\partial W} = x_1, \ \frac{\partial x_2}{\partial x_1} = W, \ \frac{\partial x_1}{\partial W} = x_0$$

Now to combine and substitute using the 3.8 template:

$$\frac{\partial x_2}{\partial W} = x_1 + W x_0 = W x_0 + B + W x_0 = 2W x_0 + B \tag{3.11}$$

Thus achieving the same result as in 3.10.

Let's now get back to equation 3.8. We notice another term ($\frac{\partial s_{n-1}(W)}{\partial W}$) which has both direct and indirect dependency on W. But what we can do now is reapply the same expansion this moves us again one move back in time. In equation 3.12 we can see how two of these steps back in time would look.

$$\begin{aligned}\frac{\partial s_n}{\partial W} &= \frac{\partial s_n}{\partial W} + \frac{\partial s_n}{\partial s_{n-1}}\frac{\partial s_{n-1}}{\partial W} + \frac{\partial s_n}{\partial s_{n-1}}\frac{\partial s_{n-1}}{\partial s_{n-2}}\frac{\partial s_{n-2}}{\partial W} \\ &= \frac{\partial s_n}{\partial W} + \frac{\partial s_n}{\partial s_{n-1}}\frac{\partial s_{n-1}}{\partial W} + \frac{\partial s_n}{\partial s_{n-1}}\frac{\partial s_{n-1}}{\partial s_{n-2}}\frac{\partial s_{n-2}}{\partial W} \\ &+ \frac{\partial s_n}{\partial s_{n-1}}\frac{\partial s_{n-1}}{\partial s_{n-2}}\frac{\partial s_{n-2}}{\partial s_{n-3}}\frac{\partial s_{n-3}}{\partial W}\end{aligned} \tag{3.12}$$

In a better notion the pattern observed from 3.12 can be written using sums and product like this:

$$\frac{\partial s_n}{\partial W} = \frac{\partial s_n}{\partial W} + \sum_{i=1}^{n-1}\left(\prod_{j=i+1}^{n}\frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial s_i}{\partial W} \tag{3.13}$$

If we plug 3.13 into 3.5 and then into 3.4 we get the equation for calculating the gradient with respect to W. The gradient with respect to U can be derived in a similar fashion, we again define the total and partial losses in a same way:

$$\frac{\partial \mathcal{L}(y_0...y_N, o_0...o_N)}{\partial U} = \frac{1}{N}\sum_{n=0}^{N-1}\frac{\partial \boldsymbol{\ell}(y_n, o_n)}{\partial U} \tag{3.14}$$

$$\frac{\partial \boldsymbol{\ell}(y_n, o_n)}{\partial U} = \frac{\partial \boldsymbol{\ell}(y_n, o_n)}{\partial o_n}\frac{\partial o_n}{\partial s_n}\frac{\partial s_n}{\partial U} \tag{3.15}$$

We again observe the same issue as s_n depends on U directly and through s_{n-1}. Doing the same tedious derivation we will arrive to basically the same equation as 3.4:

$$\frac{\partial s_n}{\partial U} = \frac{\partial s_n}{\partial U} + \sum_{i=1}^{n-1}\left(\prod_{j=i+1}^{n}\frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial s_i}{\partial U} \tag{3.16}$$

Which we would again plug into 3.15 and 3.17 respectively to get the formula for the gradient. Obtaining the gradient with respect to V is easy, we can directly write the total loss as:

$$\frac{\partial \mathcal{L}(y_0...y_N, o_0...o_N)}{\partial U} = \frac{1}{N} \sum_{n=0}^{N-1} \frac{\partial \ell(y_n, o_n)}{\partial o_n} \frac{\partial o_n}{\partial U} \tag{3.17}$$

Here no dependency on previous states is present allowing us to calculate the gradient directly without any complications.

Having derived the backpropagation for RNN allows us to closely inspect it's properties. I would like to point the readers attention especially to the products present in equations 3.13, 3.16. The continuous multiplication leads to amplification of the vanishing and exploding gradient problems. The number of multiplications is dependent on the sequence length and it is notoriously difficult to train RNN's on this type of data [31],[32], [33]. This is not the only problem imposed by the gradient multiplications, inherently the network tends to forget. As the gradients are propagated they loose magnitude thus as the sequence elongates the effect of the firs elements in the sequence decreases. In the next section we introduce several RNN architectures diving deeper into how these architectures try to solve the problems associated with the basic RNN architecture.

3.1.2 Different RNN based architectures

LSTM

Long short-term memory or LSTM networks use an idea of so called memory unit to control what information the model stores for later. This control is done through three different gates.

- Forget gate
- Input gate
- Output gate

As the names suggest each of these gates regulates distinct type information. To govern this flow each gate uses a sigmoid activation paired with multiplication to act as a weighting block. The memory block structure can be observed in figure 3.4. First let me explain the notation.

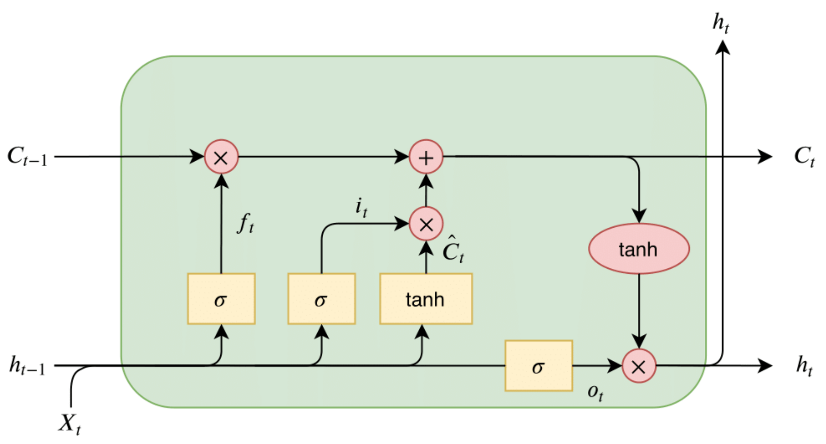


Figure 3.4: Memory block architecture of LSTM's. Image source:[34]

- C_t - Cell state representation in time t,
- h_t - Hidden state in time t, we can think about this as a mediator layer between the external inputs and the cell state. This can also be thought of as the network output since it's a function of the previous state, cell state and the input.
- i_t - The input gate in time t
- f_t - The forget gate in time t
- o_t - The output gate in time t

Note that these are abstractions, mostly we can think about the notation as being matricies, the sizes of these matricies are defined by the concrete architecture input and output sizes. The operations in red are element-wise. Everything in yellow is a learnable network which has the depicted activation function on its output.

Forget gate

The conceptual functionality of the forget gate is to give the cell state an ability to forget previous information. The input to this gate consists of the current network input x_t concatenated with the previous hidden state h_{t-1}. The result of the concatenation is dot producted with a parameter matrix of the forget gate, this matrix is learnable, finally a sigmoid function is applied to get a weight vector which is then multiplied with the previous cell state C_{t-1} to perform the forgetting. Mathematically this is pretty simple:

$$f_t = \sigma(W_f \cdot ([h_{t-1}; x_t])) \tag{3.18}$$

Normally a bias term is added but for simplicity sake I just assume its included in the x_t as this is how one would implement it in practice.

Input gate

The purpose of the input gate is to facilitate addition of new information gathered from the current input x_t. To achieve this a proposal cell state $\hat{C}_t$ is calculated. A learnable network is used for this. In figure 3.4 we see a *tanh* function on the outputs of the network, note that this activation function could be actually be arbitrary as no defined output range is needed. To produce i_t another network is learned, unlike in the previous case the *sigmoid* activation is mandatory here as we need the outputs to serve as weights for the proposal cell state $\hat{C}_t$.

Writing the equations for this we get:

$$i_t = \sigma(W_i \cdot ([h_{t-1}; x_t])) \tag{3.19}$$

$$\hat{C}_t = tanh(W_c \cdot ([h_{t-1}; x_t])) \tag{3.20}$$

Now the equations required to get the cell state update are finalized. As depicted in figure 3.4 the update is given by simple summation:

$$C_t = f_t C_{t-1} + i_t \hat{C}_t \tag{3.21}$$

Output gate

Now we are left with updating the hidden state and that is the objective of the output gate. The hidden state update is dependent on cell state C_t calculated using the equation 3.20 and again on the concatenation $[h_{t-1}; x_t]$ which are processed by another learnable network. The update is facilitated by multiplication of the network output o_t and the *tanh* activated cell state C_t.[35][36] Mathematically:

$$o_t = \sigma(W_o \cdot ([h_{t-1}; x_t])) \tag{3.22}$$

$$h_t = o_t \cdot tanh(C_t) \tag{3.23}$$

This architecture allow for a more sophisticated network memory management. It was successfully used in many applications though now is mostly outperformed by transformer based networks. Some of these applications include language modeling [37], speech recognition [38], handwritten text transcription [39], in combination with convolutional neural network in medical imaging [40], and finally various named entity recognition tasks such as CoNLL benchmark[41], drug recognition [42], NER in twitter messages [43].

Bi-Directional LSTM

As discussed above the RNN/LSTM are used mainly for sequential data but given the nature of information flow through the network they both capture inter-sequential relationships only in one direction. Bi-directional LSTM tries to solve this problem by essentially stacking two LSTM networks, this stacking is depicted in figure 3.5. The result is then given by the normalized

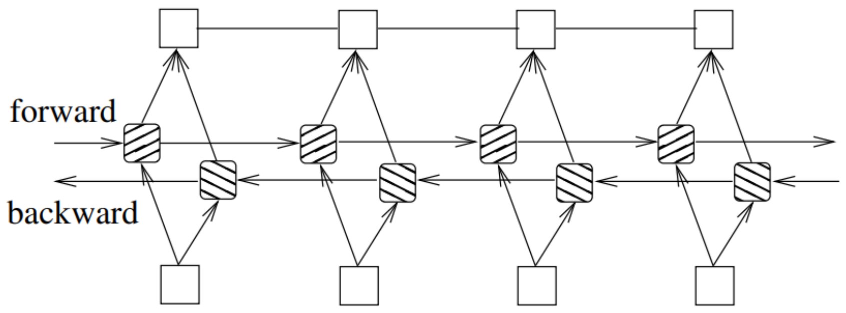


Figure 3.5: Bi-directional LSTM architecture diagram. Image source: [44]

sum of the network's outputs. This simple yet effective modification pioneered sequential data processing especially in the case of text data [44][45][46].

Many of the above mentioned LSTM applications use this approach. As we will discover in later sections many other architectures take inspiration from this bi-directional approach as well as adapting some core concepts of RNN architectures.

3.2 Transformers

Transformers are a big step forward from architectures based on reccurent layers, instead of using these transformers depend on so called attention mechanism. This mechanism provides variety of benefits making transformers completely overtake virtually every other architecture for a majority of tasks, especially natural language processing. Almost every model used in this work utilizes some version of this architecture so I deem important to introduce the internal mechanisms of transformers thoroughly thus dedicating whole section.

3.2.1 Architecture

Transformers mostly utilize an encode-decoder architecture. What differentiates them is the attention layers inside both the encoder and decoder which allow updating of the embedded vectors of tokens with respect to context. Tokenization must be done beforehand for simplicity we can think about tokens as words in text but it is not always the case. Tokens can be anything the important part is they are the entity which is input to the embedding layer of the encoder. Let us now understand how the attention works.

Attention

The main three building blocks of attention layer are queries Q, keys K and values V. Before explaining how these interact with each other let's look at them separately to get some concrete idea. To make the explanation simpler we will track one embedding of a token as it goes through the network. To denote this I will use lower-case letters. Also note all vectors are treated as columns.

Query

To get intuition for what queries represent we can think about them as abstractions of questions. In practice every query is a vector calculated from the embedding. The calculation is done using a learnable parameter matrix W_Q. This matrix is shared for the whole attention block and should somehow learn what "questions" are most relevant to a given embedding. So a query q_1 for embedding e_1 would be calculated as follows:

$$q_1 = W_Q e_1 \tag{3.24}$$

Key

Keys should somehow abstract the answers for the question posed by queries. In practice the calculation of key is very similar to queries. We again have a learnable matrix W_K and simply multiply each embedding with this matrix:

$$k_1 = W_K e_1 \tag{3.25}$$

Value

As mentioned in the intro the attention layer tries to find how to adjust the embedding to better represent the context of the sequence. The value vectors do that. Again there is a learnable value matrix used to calculate value for each embedding:

$$v_1 = W_V e_1 \tag{3.26}$$

To fully understand the meaning of value we have to provide context to how the introduced building block actually interact. We do this in the next paragraph.

Block interactions

The result of the interaction should be a vector of shift to apply onto the initial embedding vector. Let's call this vector Δe_1. Remember we want to shift the embedding vector with respect to the sequence context. This context is conceptualized by the query key pair. In other word we want to move the embedding vector while considering queries with the most relevant key. Transforming these thoughts into the math world, since keys and queries exist within a space of the same dimension we can measure the relevance between these two by how much the direction they point to differ.

Matrix multiplication is perfect for this as for vectors pointing in the same direction its values are high and in opposite they are negative thus quantifying the similarity in an unbounded manner. The unbound nature is not very practical as it can lead to numerical issues while also not really representing any mathematical measure per say. To solve this we convert the output to probability using $softmax$ function on the result.

In our case of one embedding the query will be a vector but we need to measure the similarity to all keys so let's denote the matrix of keys as K and put the vectors $k_1...k_N$ so each column of K is one of these vectors, we also need a matrix of all values let's call it V and fill it in the same fashion like K.

$$\Delta e_1 = softmax(q_1^T K)V^T \tag{3.27}$$

Now the update to the embedding is simply given by addition with Δe_1:

$$e_1^{new} = e_1 + \Delta e_1 \tag{3.28}$$

Defining matricies ΔE and Q in similar fashion like K and V we can write equations 3.30 and 3.27 for all embeddings:

$$\Delta E = softmax(Q^T K)V^T \tag{3.29}$$

subsequently the update:

$$E^{new} = E + \Delta E \tag{3.30}$$

Let's now discuss the dimension of the learnable matricies as they define the number of parameters the model has. The notation is following:

- d_{QK} - dimension of the query and key vectors
- d_e - dimension of the embedding space

From this we can infer the parameter matricies sizes. W_Q and W_K both map from space of dimension d_e to a space of dimension d_{QK} so W_Q and W_K are both of dimension $d_{QK} \times d_e$. The value matrix W_V maps between two spaces of dimension d_e so it's size is $d_e \times d_e$.

Generally the embedding space has much higher dimension than the key query space. This makes the matrix W_V really large. But since the matrix is square a low rank approximation using two matricies can be made. The inner dimension of the two matricies is usually set to match the query key space dimension. This greatly reduces the number of parameters while keeping the embedding space large enough to capture most of the semantic meaning. Also doing this can save computation as we can perform the weighting on smaller vector. Let me demonstrate in a mathematical framework.

We will need to define the low rank approximation of V. This is simply:

$$V = W_U W_D E \tag{3.31}$$

Where W_U and W_D are the low rank approximation matricies. The dimensions of W_U and W_D are $d_e \times d_{QK}$ and $d_{QK} \times d_e$ respectively. Now we substitute for V into 3.29:

$$\Delta E = softmax(Q^T K)(W_U W_D E)^T \tag{3.32}$$

And using the transpose rule for matrix multiplication we get:

$$\Delta E = softmax(Q^T K) E^T W_D^T W_U^T \tag{3.33}$$

Now we see that the dimensions of the multiplication $E^T W_D^T$ are $N \times d_{QK}$ and as mentioned above generally $d_{QK} << d_e$. In simple terms using equation 3.33 allows us to calculate the weighting on a much smaller vectors.

Multi-headed attention

It is no surprise that in the transformer there are many of these attention blocks. Just like FFN's have many neurons in each layer transformer has many attention blocks in each layer. We call this layer a multi-head attention layer as each attention block is known as a head. In practice this is many attention blocks ran in parallel. Running them in parallel allows us to merge the output operations across all the attention blocks. What we can do is split out W_U from equation 3.33 each head will now produce an output of dimension $N \times d_{QK}$. Now we do two concatenations:

1. All W_U matricies across all heads. We get a big matrix of dimensions $d_e \times n_{heads} \cdot d_{QK}$
2. The split heads outputs across the d_{QK} dimension to get big matrix of size $N \times n_{heads} \cdot d_{QK}$

Multiplying these two big matricies equals to taking all the proposed changes ΔE from each head and summing them. From this we can calculate the new embedding vectors using 3.30.[47], [48], [49].

Cross-attention

Following the original transformer paper [47] many others differentiate between self-attention and cross-attention. So far we assumed keys, queries and values are all calculated from the same embeddings but this doesn't have to be the case. In cross-attention block these are not calculated from the same embeddings. Simple intuition why this might be helpful could be demonstrated on the example of text translation.

In this case we can calculate the queries from emebedding in one language and the keys from the other. This makes perfect sense as the query key pair resembles the translation pattern quite well. Most commonly the cross-attention is between the encoder and decoder block of the whole architecture as we will see in the next section where we present the original transformer of [47].

Example architecture

Obviously the attention layer is only one component of the whole network. Let's now inspect the original transformer of [47]. The whole structure can be seen on figure 3.6. The left part is the encoder on the right the decoder. Let's

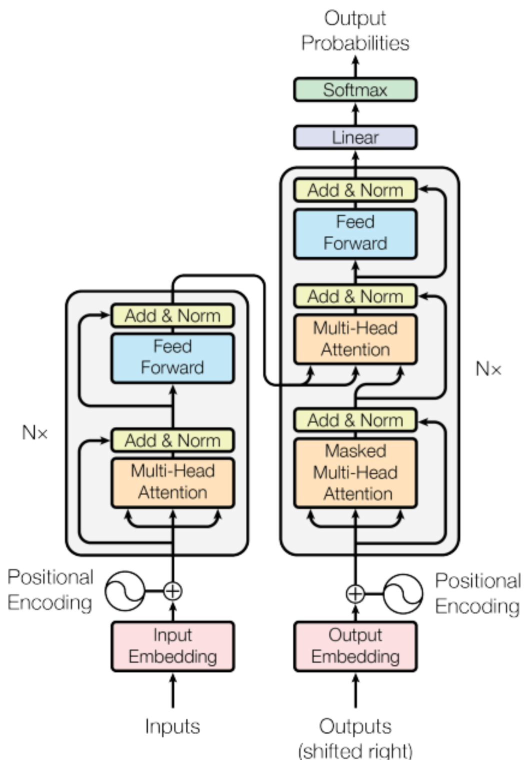


Figure 3.6: Original transformer architecture. Image source: [47]

now discuss the blocks. On each multi-head attention we see an add&norm block. This block is inspired by the residual skip architecture [50]. It simply adds the input of the multi-head attention block to its output. The residual networks are known to increase performance of very deep networks as they propagate the input even to the weights hidden deep in the network [51].

After, a normalization is performed, as its positive effect on both the performance and mostly calculation load needed when training is widely established [52], [53], [54]. A classical simple FFN with another residual skip is then used to further increase the models capacity. Let's now talk about the attention blocks. The block in the encoder is just a normal multi head attention. But the blocks in the decoder differ slightly. We see the upper one being connected to the encoder. This is exactly the case of cross-attention, the keys and values come from the encoder while the queries are made by the decoder. In a simple sense the keys and values are the low level representations present in any encoder decoder structure.

The last head, also in the decoder, has masking present. The decoders access to the output creates a simple problem for the attention block where when calculating the query key similarity we could actually make keys corresponding to future embeddings influence past queries. This backward flow of information is not desired as it basically allows the network to learn how to cheat. Also in practice it never has access to the future so this flow makes no sense. A simple fix is to set everything under the diagonal of $Q^T K$ to $-\infty$ as when $softmax$

is calculated the $-\infty$ turns into 0 to make the future-present similarity 0. This is what masking refers to as we mask the lower triangle of the matrix to the further layers of the model.

We also notice that unlike in the RNN there is no inherent encoding of position given by the architecture. The transformer processes the sequence without any recurrent connections thus omitting the position of the token in the sequence. To solve this a vector of the same dimension as the embedding is added. This vector is calculated using a function of the sequence position. This function can be arbitrary as long as it is dependent on the sequence position and the output dimension matches d_e to allow for the vector addition. In our example a sine based function is used.

3.3 Large language models overview

3.3.1 BERT

Bidirectional Encoder Representations from Transformers (BERT)[55], is a natural language processing (NLP) architecture introduced by Google in 2018. Unlike previous transformer models that mostly processed text in a unidirectional manner, BERT employs a bidirectional approach taking inspiration from above mentioned bi-directional LSTM. The training process also differentiates BERT from its predecessors. BERT generally uses two learning strategies, masking and next sentence prediction. A loss function combining the two strategies is subsequently optimized.

Masking

The model is fed a sentence with a masked token a tries to predict this token, note here the model will predict the whole sentence but we only care about the masked word prediction. This procedure leads to very efficient usage of the available data since each word in the sentence can be masked yielding number of examples from one sentence equal to its length in words. Note that the tokens are not used directly but rather their vector embeddings as illustrated in 3.7

Next sentence prediction (NSP)

In NSP the training data are divided into pairs of sentences 50% of these pairs are random the other 50% are subsequent sentences. The model's goal is to predict if the given sentences are in fact subsequent or a random pair.

3.3.2 Applications

BERT was applied to most natural language processing tasks such as Named entity recognition (NER), relation extraction, question answering, sentence

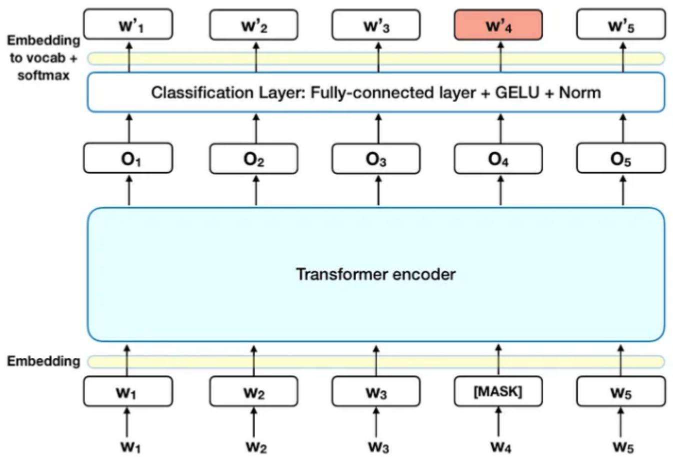

w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Figure 3.7: Masked training diagram [56]

embedding [57], [58], [59]. The network is mostly trained in unsupervised manner on a large corpus of data from one field, for example biomedical[58] or scientific[59] data. Results show superior performance of the models augmented with the unsupervised context learning within its field. These models mostly outperform bare BERT and are comparable to state-of-the-art solutions within their respective field. NLP is a fast evolving field so even results old only few years can be outperformed by now.

Big advantage of BERT is its size, the base model has only 110million parameters. Compared to other language processing models this is very lightweight. For example GPT-4 [60] has 1.7 trillion parameters, mistral [61] 7.3 billion.

3.3.3 Llama2

Llama2 [62] is the predecessor of Llama3 both being LLMs developed by Meta. I mainly mention it here just for the sake of comparison to Mistral discussed below. LLama2 uses classical learning approach, self-supervised pretraining followed by fine-tuning step including multiple inputs but mainly using human feedback for evaluation of chat generation also known as reinforcement learning, the whole cycle is illustrated in Figure 3.8. The main difference from other LLMs is the presence of safety reward model. This "loss" function gives the network feedback on usage of personal data trying to keep this information safe. Most other open source LLM don't use this and probably would be safe to assume private LLMs don't use this either. Even though this model is mostly outperformed one scientific benchmarks by openAI models (GPT-3/4) it is open-source and notably not trained on any data which Meta might have. This is important as the amount of data greatly influences the performance of the model. Also the open license makes it more usable for a

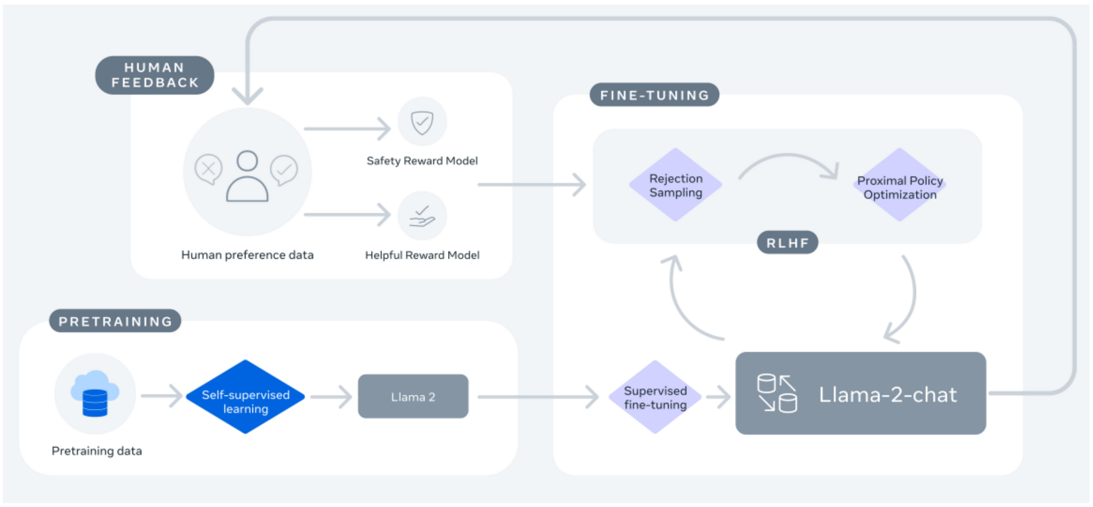


Figure 3.8: Training cycle of Llama2 [62]

project like this.

3.3.4 Llama3

The brand new successor of llama2 comes in two versions an 8billion parameter smaller model and a large model consisting of 70 billion parameters, also a model of 400B+ is currently being trained. Both a larger dataset and token vocabulary (128k tokens) were used for training. To provide some context the dataset was 7 times larger then the one used for Llama2 training. The novel dataset also includes 5% of data in other languages than English, this is done to better the performance on multilingual tasks. As far as the origin of data used goes, meta claims only publicly available sources were used, avoiding using 3rd party licensed data like GPT models.

The architecture is decoder only based. Looking back at figure 3.6 we could achieve this architecture by only taking the right side (decoder) and switching out the "output embedding" branch for the normal input. Obviously the subsequent architecture differs and is mostly deeper with many more layers. Rather than using the basic attention we already discussed Llama3 employs grouped query attention [63]. In simple terms the queries are divided into a given number of groups. Instead of each query having its own key each group now has its own key. This leads to a substantial speed up in learning and an increase in memory efficiency of the model. This is important for both learning and inference.

Performance

Unlike the llama2 model the performances of both the small and large versions are competitive if not better than other popular state-of-the-art models. A human preference evaluations done by meta can be seen in 3.9

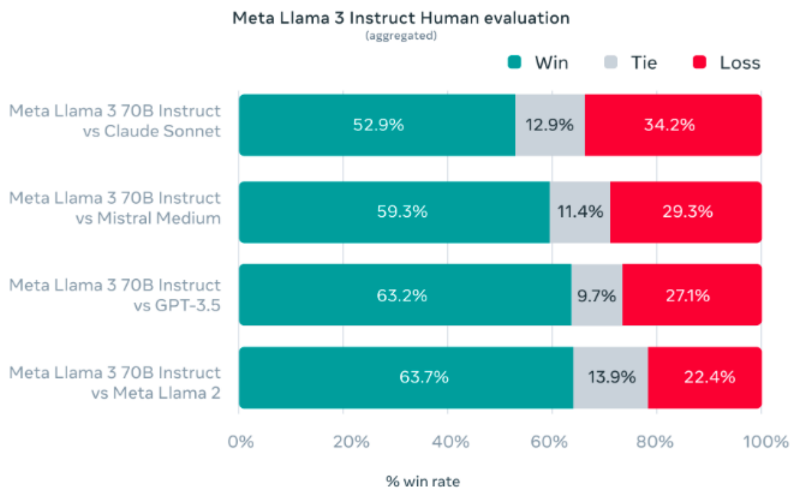
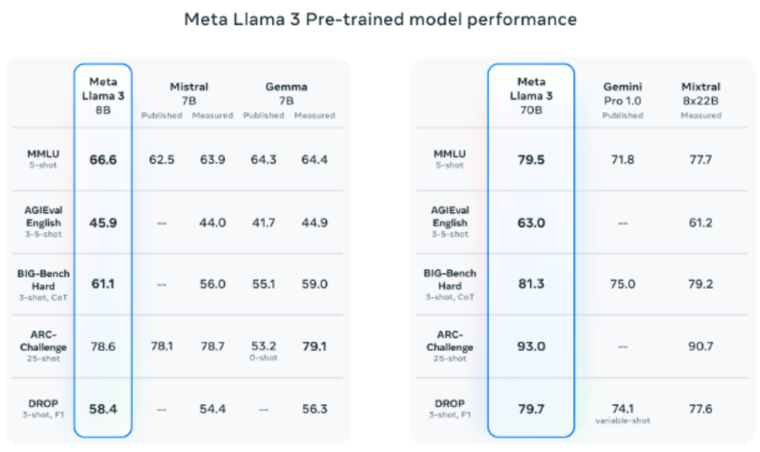


Figure 3.9: Llama3 human evaluation versus popular models. Image source: [64]

Meta also provides comparison performances on some benchmark datasets of different tasks:



Meta Llama 3 Pre-trained model performance

	Meta Llama 3 8B	Mistral 7B Published	Mistral 7B Measured	Gemma 7B Published	Gemma 7B Measured
MMLU 5-shot	66.6	62.5	63.9	64.3	64.4
AGIEval English 3-5-shot	45.9	--	44.0	41.7	44.9
BIG-Bench Hard 3-shot, CoT	61.1	--	56.0	55.1	59.0
ARC-Challenge 25-shot	78.6	78.1	78.7	53.2 0-shot	79.1
DROP 3-shot, F1	58.4	--	54.4	--	56.3

	Meta Llama 3 70B	Gemini Pro 1.0 Published	Mixtral 8x22B Measured
MMLU 5-shot	79.5	71.8	77.7
AGIEval English 3-5-shot	63.0	--	61.2
BIG-Bench Hard 3-shot, CoT	81.3	75.0	79.2
ARC-Challenge 25-shot	93.0	--	90.7
DROP 3-shot, F1	79.7	74.1 variable-shot	77.6

Figure 3.10: Benchmark performance comparisons done by Meta. Image source: [64]

Both of these results presented by meta suggest very good performance of the model. Especially with comparison to much larger models like GPT-3.5. Next I introduce results of independent evaluators.

First let's look at Huggingface leaderboard [65],[66],[67],[68],[69],[70],[71] as it provides a good insight into performance on commonly used benchmarks. Looking at the average across all the benchmarks in the top 10 performers a llama3-70B based models take 4 spots in the top 15 it is 8.

Another popular benchmark is the ChatBot arena [72]. An elo system driven ranking approach similar to one in chess. An elo is assigned to each model, the higher the elo the "stronger" the model is compared to the others. The elo is based on 1v1 clashes where the models outputs are compared by many people to asses the winner, each model takes part in many of these "duels" to obtain its elo. The instruct fine-tuned version of Llama3 70B takes shared 7th place (with Bard(Gemini pro)) on the overall leader-board, being beaten by only proprietary licensed models namely, 4 iterations of GPT-4 [60], Claude 3 Opus [73] and Gemini 1.5 pro[74].

All these results are very impressive given that Llama3 is public for two months.

3.3.5 Mistral

Mistral is a new model which demonstrates that the number of parameters is not everything. The base version has 7B parameters and outperforms the above mentioned much larger Llama2 (13B parameter version) on basically all benchmarks. It is also open-source making it a great option to test in our project. We already use Mistral to perform multiple tasks such as above mentioned summary generation but also text-completion.

3.3.6 Mixtral

The mixtral models are developed by the same authors as the mistral ones. These models aim to provide accessible inference computational load while maximizing performance on different tasks. They adapt an old idea of mixture of experts [75]. A quite simple idea of training so called expert networks and one gate network, this gate network acts like a selector of which of the expert networks will be used for inference on a given input. In practice the gating network calculates weight for each of the expert and then the experts with the highest weight are selected. The amount of selected experts is a hyperparameter of the model. This approach allows the network to maintain high amount of parameters in training providing greater context scope while enabling the inference to run only on a subset of all the models parameters [76].

Performance

Mixtral models come in two sizes 8x7B and 8x22B. Both having 8 experts of the two respective parameter sizes. We again use the hugging face leaderboard and Chatbot arena [72] as these are the most popular and together cover both human evaluation and benchmark performance metrics. On the benchmark leaderboard from huggingface mixtral 8x22B takes 6th place beating all the llama3-70B based models while also having the least inference cost. The performance on human evaluated benchmarks is rather worse, the larger version of mixtral shares 21st place in Chatbot arena with the medium sized version of Mistral.

3.3.7 Gemma

The gemma models developed by google come in two versions both relatively small, 2B and 7B parameters. These models were trained on 3T and 6T tokens respectively. On human evaluated benchmarks it has around 60% winrate against mistral-7B-v0.2 since I deem the human evaluation more important than concrete supervised benchmark evaluation I decide to use the larger gemma instead of mistral, except for in the fine-tuning part. From now on when I refer to Gemma I mean the 7B version [77].

Chapter 4

Methodology and results

This chapter presents the data gathered by the NQL, describing the difficulties encoutered while processing it. Later we implement an API capable of extracting entities from text mainly adapting work of [78]. We also build the foundation for future work and present ideas about evaluating accuracy on unlabeled data. We don't show any statistically significant results since this work is mainly exploratory and statistics will be subsequently included in my diploma thesis.

4.1 Using NER to extract basic entities from czech text

There are many approaches to Named entity recognition. Classical approaches include Hidden Markov Models [79], Decision trees [80], SVM [81] and more statistically based approaches leveraging entropy maximization [82]. Eventhough these approaches pioneered the research of Named Entity recognition problems, with the rise of neural networks and mainly transformer architectures capable of context based learning and inference NER tasks like many others became solvable using these methods yielding superior performance over the traditional ones mentioned above. This lead me to subsequent research and usage of LLMs for solving NER problem presented previously. To explore the possibilities I adapted the work done by [78] to solve my task at hand. Next I briefly introduce their work and then provide details about implementation of an API capable of producing model output and parsing it.

Authors of [78] use a bi-directional LSTM architecture and treat the problem as a seq2seq task, each token is treated as a sequence and the output labels are predicted also as a sequence a special label is used for end of word. To further augment the data, pretrained networks (BERT + Flair) are used to calculate contextual embedding of the data to further improve the classification.

My work was the implementation of an API capable of calling the model and parsing both input and output. The simple API was implemented in

Flask [83]. Flask is a python framework mainly for back-end web development. I chose it due to its simplicity implementing an API of this size doesn't require a framework like Django which generates a whole project, flask enables me to program everything in one simple code file.

This API served mainly as an introductory work to extract entities like name, adress, phone number, omitting the more context based entities like financial situation etc. In the next sections I evaluate models on the more difficult task of having arbitrary fields possible.

4.2 The data used

NQL provided around 1500 hours of dialog from phone call consultation sessions. This data was transcribed and provided in form of text files, a version translated to English was also provided. An effort was made to test the above implemented system one this data but the transcriptions were deemed bad. I independently read about 250 conversations and they proved to be hard to understand even for a human. I selected 39 best examples from these conversations, to use for further processing. I corrected the diarization and some basic mistakes in these dialogues.

For fine-tuning an automatically generated data was used. The generation was provided by GPT-3.5 and GPT-4 models. Both these models were asked to create artificial dialogues between a therapeut and a patient, a list of form fields was also given. After the generation the models were tasked to fill the form provided. This data was subsequently used to fine-tune other models. In other words the output of GPT models was used as ground truth.

4.3 Fine-tuning

Fine-tuning of mistral 7B and mixtral 8x7B was employed on the data generated by the GPT models. Both these models are relatively small but still a 4bit quantization had to be used. Mixtral had to be trained on A100 with GPU memory of 40Gb as it wouldn't fit on my GPU with 12Gb memory. The next section introduces the fine-tuning approach used.

4.3.1 QLoRa

Quantized Low rank adaptation [84] is and efficient way to fine-tune large networks while utilizing a fraction of memory needed for their full training. To achieve this goal it combined the original LoRa [85] with NormalFloat quantization. To fully understand QLoRA let's briefly introduce LoRa in the next section.

LoRa

Let's start by defining the objective of the fine-tuning. We want to tweak the current models weights W to improve the performance on a specialized task. More rigorously we are searching for such ΔW, a change to apply to W, which minimizes the loss on our task. Given the nature of neural networks layers the W can be thought of as a matrix, in general this matrix is very large and subsequently the ΔW matrix will be large too. LoRa leverages a low rank approximation of ΔW just like the value matrix in transformers. Mathematically:

$$\Delta W = w_1 w_2 \tag{4.1}$$

where:

- ΔW - matrix of dimensions $n \times k$
- w_1 and w_2 - matricies of dimensions $n \times r$ and $r \times k$ respectively

The justification for this is following. The W matrix is large it almost certainly doesn't have full rank and thus contains redundant information, the objective is then to get rid of the linearly dependent rows/columns which contain this redundancy. The main problem is we don't have access to the rank of the matrix W and also the network contains multiple of these matricies and most certainly each has different rank and we want to generalize. What we then do is make the rank a hyperparameter of the fine-tuning. More concretely we make the dimensions of the estimate matricies the hyperparameter. This corresponds to the dimension r of w_1 and w_2 from 4.1.

The training step of the fine-tuning is then done only on these low-rank approximations w_1 and w_2 greatly reducing the number of parameters needed to be stored on the GPU memory. Once these are trained ΔW is calculated and added to the models W to obtain the fine-tuned version.

To bring this further, an idea from the adapters[86] based fine-tuning can be employed. Instead of inserting adapters we can simply only modify matricies W of certain layers.

NormalFloat quantization

Normally the quantization assumes uniform distribution of the values to be quantized. In neural net we generally don't have uniform distribution of the weight values. This information can be leveraged to retain more information when performing quantization. As the name suggest NormaFloat quantization assumes normal distribution. Note that the weights have to be normalized to zero mean and standard deviation of 1.

QLoRa

QLoRa is basically LoRa on a NormalFloat quantized model. In our case we quantize down to 4 bit integer.

4.4 Measuring text similarity

Measuring text similarity is important for comparison of the model outputs. Use of classical distances such as Levenshtein distance is not viable given the nature of some form fields. Since our problem is not concretely defined and arbitrary form fields should be acceptable we need to find a measure capable of capturing similarity between outputs grossly varying in format. There are two broad approaches to text similarity. First leveraging the strings directly by either calculating some kind of a distance such as Levenshtein distance, Hamming distance or a more elaborate approach using some kind of corpora or a knowledge base. Second and more modern approach tries to extract semantic meaning from the given text with a mathematical transformation. The acquisition of these transform is defined by the concrete approach, this could be deep learning from a dataset or employing a dimensionality reduction technique on corpus matrix. In the next sections we will dive deeper into these methods.[87],[88].

4.4.1 Basic string distance and similarity measures

Hamming distance

Hamming distance is pretty simple it is just the amount of non-corresponding characters between the two strings. It is only defined for strings of the same length. One might ask why we even mention it and the reason is following. If we don't think about the distance in a correspondence manner but rather imagine how many and which operations we would need to transform one string to the other we can come to interesting conclusions. If we define a "replace" operation (switch one character in a string to another) hamming distance just becomes the minimum number of "replace" operations needed to transform one string to match the other. The framework of counting the minimum operations needed to achieve equality between strings is a powerful tool and as we will see in next sections helps define other more complex string distances.

Levenshtein distance

There is one big flaw to Hamming distance and that is the inability to deal with string of different lengths. Levenshtein distance solves this problem by adding two new operations:

- Insert - places a new character into the string
- Delete - removes a character from a string

Having defined the two new operations alongside "replace" calculating the distance becomes again a problem of finding the least amount of operations needed to transform one string to the other. [89]

Damerau–Levenshtein distance

This string metric introduces the last operation we will discuss. The operation is know as:

- Transposition - switch the position of two adjacent characters.

Damerau's main work focus was on spell checking. He found that majority of errors in spell checking could be corrected by one of the four operations used in his distance measure. This distance also saw usage in different applications genetics [90] and was inspiration for other metrics using the transposition operation.

The next few measures are represented in and inverse manner to distance also known as similarities. Most of them are normalized to output number between 0 and 1 but in general it stands that higher the number the bigger the similarity. Note that these are not metrics in mathematical sense anymore since most of them break the triangle inequality condition.

Jaro similarity

Jaro distance utilizes only one operation just like hamming distance, namely the "transposition".

DICE similarity

DICE similarity is mostly known and vastly defined on sets. Given two sets X and Y DICE similarity is defined as:

$$DICE = \frac{2|X \cap Y|}{|X| + |Y|} \tag{4.2}$$

We can define the intersection of sets as the number of matching characters between the strings and the cardinality of the sets as the lengths. Doing this we get:

$$DICE_s = \frac{2 \cdot match(S_1, S_2)}{len(S_1) + len(S_2)} \tag{4.3}$$

4.4.2 Deep-learning based approaches

As previously discussed the need for a measure able to compare more abstract semantic meaning within the text is warranted. A deep learning based approaches generally use vector embedding and then some vector distance measure to achieve a more contextually relevant results. This approach is what I will use for evaluation.

Embedding models

A BERT based architectures from [91] are used for calculation of text embeddings. Authors also provide a whole library for sentence similarity which is used to calculate the embedding [57]. With the library also multiple pre-trained models are available all being pretty light-weight in the amount of parameters making the computational load not that heavy. I choose the all-MiniLM-L6-v2 this models has around 23M parameters. It is a fine-tuned version of nreimers/MiniLM-L6-H384-uncased trained on around 1B sentence pairs from multiple datasets, and it maps the input to a 384 dimensional vector space.

Vector distance measure

There are two commonly used vector similarity/distance measures, euclidean distance and cosine similarity. With the authors of the embedding models using cosine similarity in the training process the choice is straight forward. Also in comparison cosine similarity is normalized and doesn't suffer from curse of high dimensionality like euclidean distance [92]. Cosine similarity between two vectors is defined as follows:

$$cossim(x, y) = \frac{x^T y}{||x||\,||y||} \tag{4.4}$$

It is basically a normalized vector product. It measures the how far angle wise the vectors are from each other.

4.5 Evaluation pipeline

4.5.1 Using the models

Most of the models tested are too large to be run locally, thus some hosting had to be used. Huggingface [48] provides hosting and library API for some models, this was used for fine-tuning of the Mistral models. The GPT models have a paid API provided by openai. For the Llama3, Gemma and eventually even mixtral a new hosting called GroqCloud was used, since it is in beta phase all the models were provided free of charge but under rather strick restrictions on amount of tokens processed per minute. This is the second big reason for only processing a small part of the data provided.

4.5.2 Reference models

Since the unlabelled nature of the data a two models of the models with the best expectation on performance were chosen as a reference. These models were GPT-4 and Llama3-70B. These were selected for two main reasons, one they are developed by different companies thus the training data might differ more than if I used two GPT based models and two because these models have outstanding results on human evaluated benchmarks. All the testing was done with respect to each reference model we discuss the effects of this scheme are discussed further down the work.

4.5.3 Form field and data filtering

For evaluation an example 9 form fields were chosen, but in general any form field should be acceptable. These fields were chosen:
gender, name, age, phone number, type of addiction, length of addiction, frequency of use, medical history, financial situation
It happens that in the given dialogue the information about the field is not present thus the models are not able to fill it. This occurs quite often and as will be demonstrated in the results section the models generally answer correctly that the given information could not be retrieved. In order to remove a bit of this bias in the results a filtering is applied and the results are then calculated on both the unfiltered and filtered data. The filtering is following, when the reference models output is that the information about the given field is not present in the text we remove this entry from further processing.

4.5.4 Model output format

The reliable output format of the model is very important as if the format is not stable the subsequential parsing becomes more difficult and sometimes impossible. All the models were asked to deliver the results in json format. Most models also add some kind of comment to the json output so further parsing was implemented. A sample json output after parsing could look like this.

```json
{
  "gender": "not included",
  "name": "Veronika",
  "age": "not included",
  "phone number": "not included",
  "type of addiction": "nicotine",
  "length of addiction": "2.5 years",
  "frequency of use": "30 heatsticks a day",
  "medical history": "Throat issues",
  "financial situation": "not included"
}
```

4.6 Results

4.6.1 Fine-tuning results

The data generated by the GPT model was divided into a test and training splits respectively. Fine-tuning of Mistral 7B and Mixtral 8x7B was performed yielding 100% accuracy on the test set. Subsequently the models were used on the NQL data and both failed to produce results in a usable format. Thus forcing us to remove them from further testing.

The suspected reasons for the poor performance are a big bias in the data used for fine-tuning. The desired objective was mostly to make the models learn the correct output format, but instead the fine-tuning mostly allowed to models to fixate on wrong aspects of the dataset. These aspects were a simple and repeating nature of the dialogues and a quite short length compared to the NQL data.

Especially the length posed a big problem as when the fine-tuned models were presented with the longer samples from the selected NQL dialogues they were inable to process the text and outputted results of no value. To fix this I tried cutting the longer dialogues to segments, this helped and the models yielded a correctly formatted output but with completely made up answers. Given the performance of the bare models I deemed the fine-tuning unhelpful.

4.6.2 Field-wise performance

To asses the performance of the different models on each field an average cosine similarity over both the unfiltered and filtered data was calculated for each model on every field. For visual clarity and evaluation purposes the fields were divided into three categories by an estimated semantic context abstraction needed to fill the field. The categories contain the following fields:

1. Gender, Name, Age - the easiest category, Gender might need more context but when name is present can be inferred easily

2. Phone number, type of addiction, length of addiction - The medium category, requires a basic understanding of the dialog context but answers are generally simple

3. Frequency of use, medical history, financial situation - The hardest category requires understanding of the text while also requiring answer of no predefined structure

Inspecting figures 4.1-4.4 suggest a decreasing trend in cosine similarity values through the different categories, calculating average values on each reference for every category can be seen in table 4.1. These values suggest that the difficulty of category 1 and 2 are on the same level as the averages are close. This can be the case due to the phone number field as its presence has mostly binary nature, either the models are very close to the reference or they don't find the phone number at all. The dataset doesn't contain many examples with phone numbers included and if they do the transcription isn't very good. This may affect the evaluation accuracy between these two categories.

	Llama3-70B	GPT-4
1. category	0.72	0.72
2. category	0.76	0.69
3. category	0.55	0.41

Table 4.1: Average cosine similarities over both data types for each category

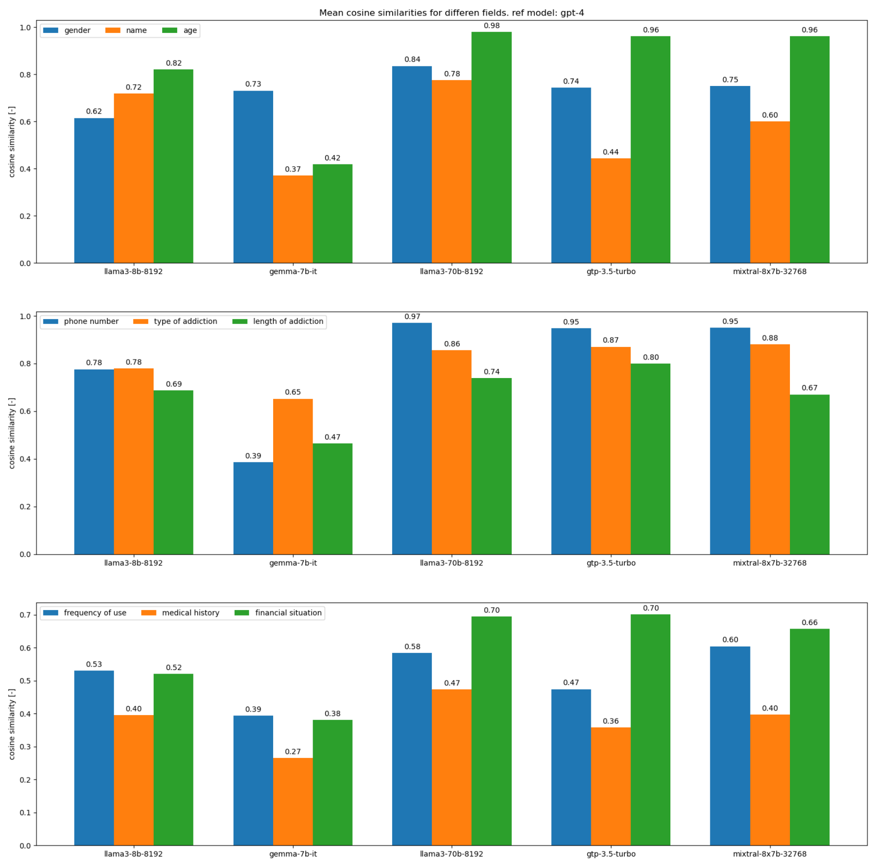


Figure 4.1: Unfiltered data cosine similarities, fields comparison. Reference model: GPT-4

As expected the average values in the 3rd category are smaller than in the two easier categories. Main reason for this is obviously the more abstract and thus harder semantic meaning of the fields. But also the evaluation method plays a bigger role here as now it is more likely the correct outputs in the fields won't be close in some string distance sense(4.4.1) but more in their semantic meaning. Comparing semantically close sentences is generally harder task even for the embedding based approach. This fact contributes to generally lower similarity values within the 3rd category.

Again an exploratory analysis of figures 4.1-4.4 begs the question if some models performance is dependent on category. This is justified as one might think the smaller models might underperform on the harder categories. For

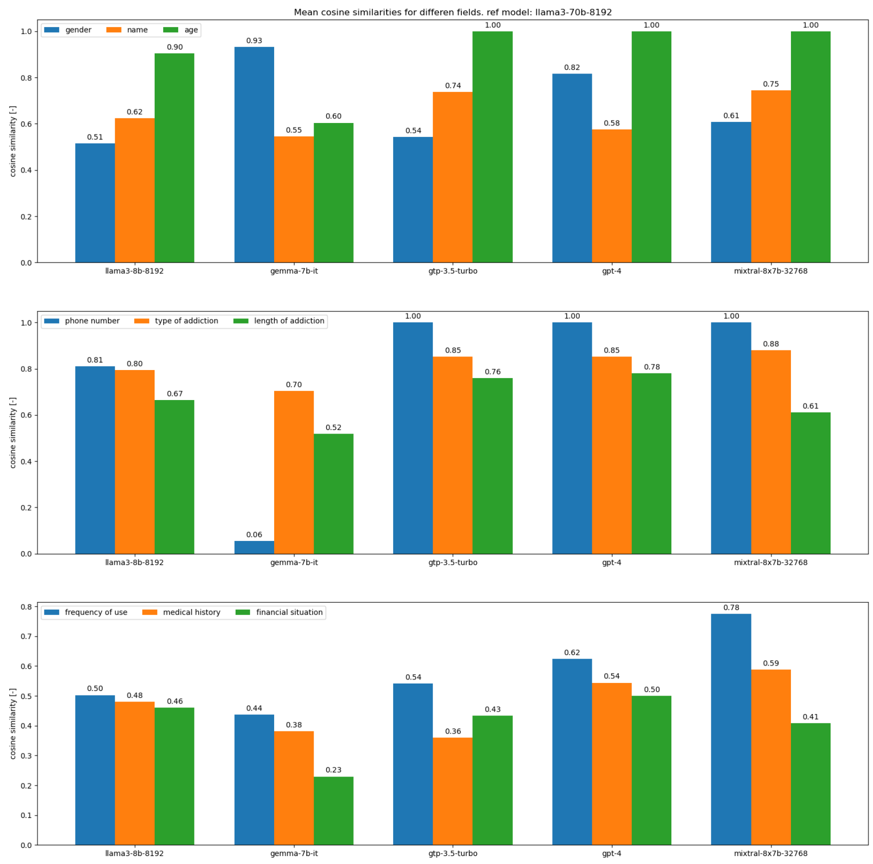


Figure 4.2: Filtered data cosine similarities, fields comparison. Reference model: Llama3 70B

this a separation of models was made based on their size. Smaller models being Llama3-8B and Gemma-7B. Larger models the rest.

	Smaller models		Bigger models		Differences	
Reference	Llama3-70B	GPT-4	Llama3-70B	GPT-4	Llama3-70B	GPT-4
1. category	0.65	0.67	0.77	0.76	0.12	0.09
2. category	0.60	0.61	0.86	0.74	0.26	0.13
3. category	0.47	0.35	0.61	0.46	0.14	0.11

Table 4.2: Table of average cosine similarities over both data types on smaller and larger models across all categories along with the respective differences between small and large models

An average performances of small and large models on each category on both references are summarized in table 4.2. Looking at the difference columns we notice the values are relatively stable with one exception.

The performance of large models with reference to Llama3 in second category is significantly higher than the one of smaller models. Looking at figures 4.2 and 4.4 we notice a very poor performance of the Gemma model on the phone number field. In reality this is due to Llama3-70B not detecting the phone numbers. This might seem like a mistake but in reality after inspecting the transcriptions it is impossible to extract the phone numbers from the transcriptions that include them as they are not fully transcribed. I wouldn't put this as a mistake of either model since it is mostly originated in the poor nature of the data.

With this understanding I categorize this exception as irrelevant and observing the other differences we notice no major change with respect to the category.

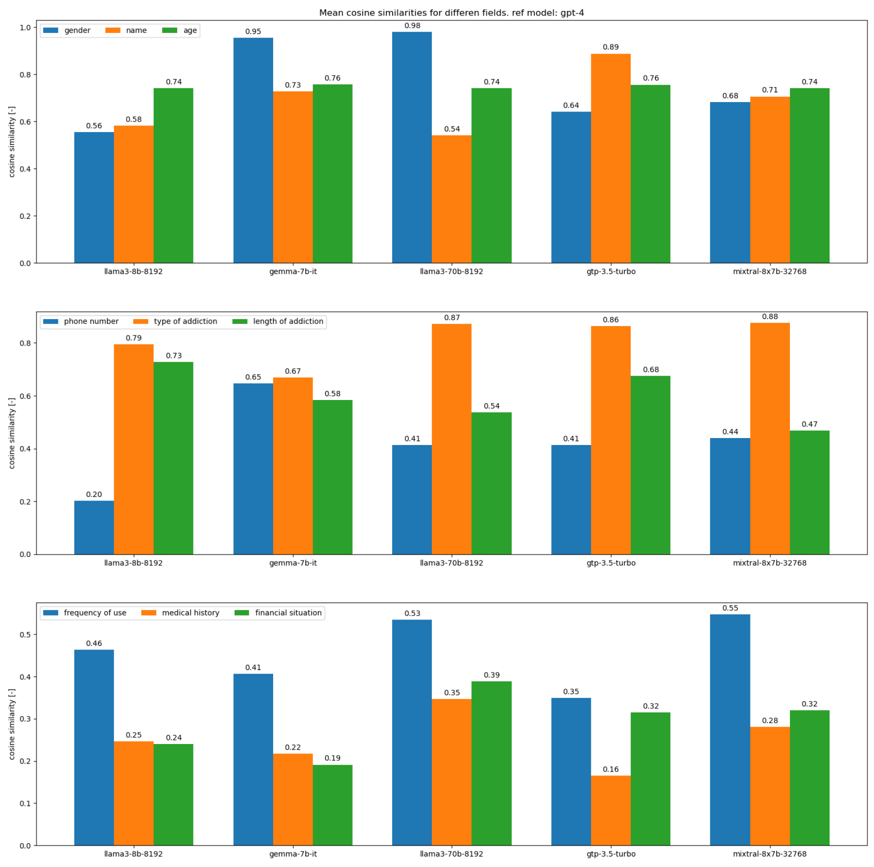


Figure 4.3: Filtered data cosine similarities, fields comparison. Reference model: GPT-4

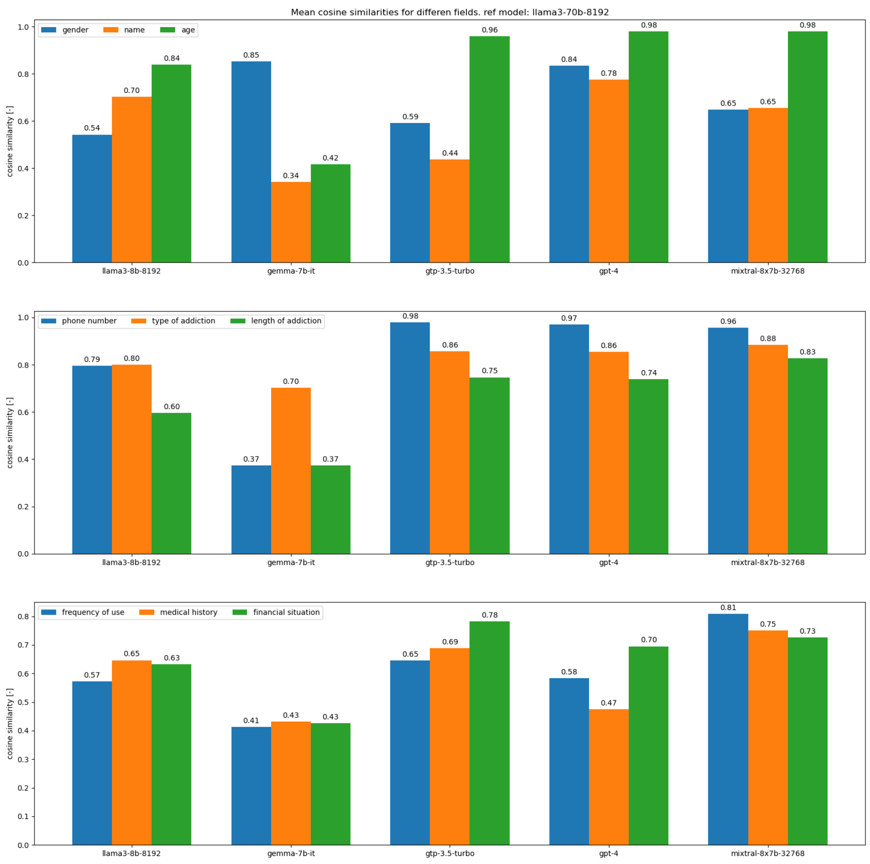


Figure 4.4: Unfiltered data cosine similarities, fields comparison. Reference model: Llama3 70B

4.6.3 Output reliability

A simple count of mistakes was used to asses the reliability of the models outputs. A mistake was counted when it wasn't possible to extract the given field from the output using simple parsing, this means in one output a model can make more mistakes. Most common mistakes were either complete disobedience of the requested format or missing out on quatation marks.

	Llama3-8b	gemma-7b-it	Llama3-70b	gpt-3.5-turbo	gpt-4	mixtral
Output mistakes	12	2	0	4	0	5

Table 4.3: Table of mistakes in output of each model

As table 4.3 indicates the worst performer by far is the small Llama3 model. The main reason for this is its complete disobedience on one sample thus having 9 mistakes from it. The same case is with the mixtral where all the mistakes occured within one faulty sample. The only models without any mistakes were the reference models, even gpt-3.5 had few mistakes, these were mainly missing quatataion marks. It could be argued the reliability of gpt-3.5 is worse than mixtral as it makes mistakes in different samples when mixtrals mistake is only one faulty sample. To conclude these results are quite concerning as the reliability is a huge factor in model selection for the application.

4.6.4 Effects of data filtering and reference selection on results

In order to compare the performance with respect to each reference an average of all data and fields was calculated for both the reference models. These averages can be seen in table 4.4 and suggest around 7% higher average on the Llama3-70B reference. Using a voting scheme for evaluation this would imply better performance of Llama3-70B as the tested models tend to agree with it more.

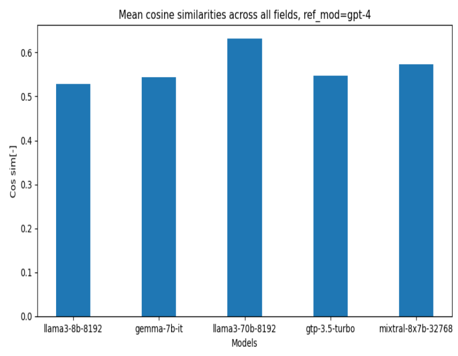
	Llama3-70B	GPT-4
Average over reference	0.68	0.61
Unfilt. vs filt data difference	0.035	0.098

Table 4.4: Comparison of different average metrics over reference models and data types

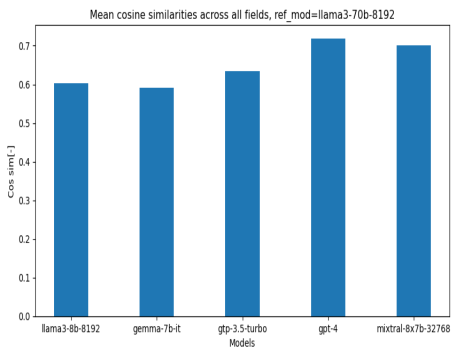
Inspecting figures 4.5, 4.6 one would assume a generally lower cosine similarity values on filtered data, the calculated differences in table 4.4 confirm around 6% difference. This discrepancy is mainly due to models having easier time performing on fields about which no information is included in the text. In general it seems confirming no occurrence of information about a given is a simpler task. The filtered dataset removes some of these samples thus generally decreasing the average performance of all the models.

4.6.5 General model performance

First looking at the average cosine similarities on the filtered data in figure 4.5 the highest performing model with respect to the reference is in each case the other reference model. This further justifies the usage of these models as references. The other models perform about the same with respect to GPT-4 while when using Llama3-70B as the ground truth model we see mixtral slightly outperform the other models.

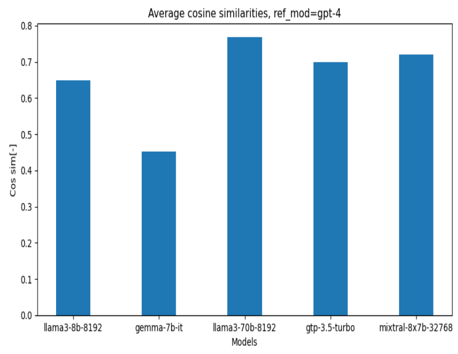


(a) : Reference: GPT-4

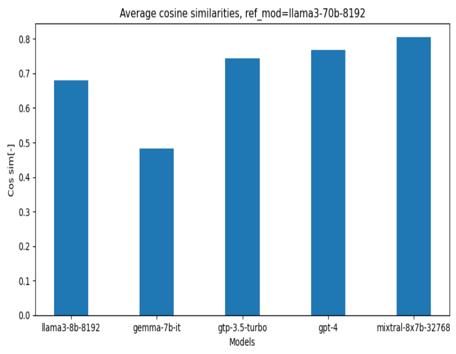


(b) : Reference: Llama3 70B

Figure 4.5: Comparison of average cosine similarities across all fields on filtered data for different reference models



(a) : Reference: GPT-4



(b) : Reference: Llama3 70B

Figure 4.6: Comparison of average cosine similarities across all fields on unfiltered data for different reference models

On the unfiltered data in figure 4.6 we notice that GPT-4 is actually outperformed by mixtral. As explained above this is probably due to the phone number field not being recognized by Llama3-70B while being found by the GPT-4 model thus lowering its performance with respect to Llama3-70B.

We also notice a substantial decrease in performance of the Gemma model. As discussed Gemma is the smallest model and its ability to follow instructions completely is thus limited. This leads to differently phrased output to describe the inability to fill the field from the given text. Sometimes instead of saying "not included" Gemma outputs and abbreviation with similar meaning, for example "Not provided". Since the embeddings of completely same strings are mapped to equal vectors the models which match completely with reference have higher performance. This problem only occurs on unfiltered data as in the filtered case we never compare samples where the reference model doesn't find the information.

4.6.6 Cosine similarity scores distributions

So far only all the results observed were averages over the data. To better understand how the scores are distributed we can inspect one sample histogram for each model. In figure 4.7 we see histograms of cosine similarities averaged over the fields for each model with GPT-4 as a reference, these were calculated on filtered data. Since the histograms have generally similar pattern the other cases are included in the appendix.

The general pattern is obvious, when the models agree with the reference the cosine similarity tend to be close to 1. The disagreement is harder to grasp, generally there tends to be a smaller peak around 0.2. The value mean value of a dissimilarity will never be exactly 0 as the model compares from both semantic and basic string perspective and two sentences will share characters even when their meanings are completely different.

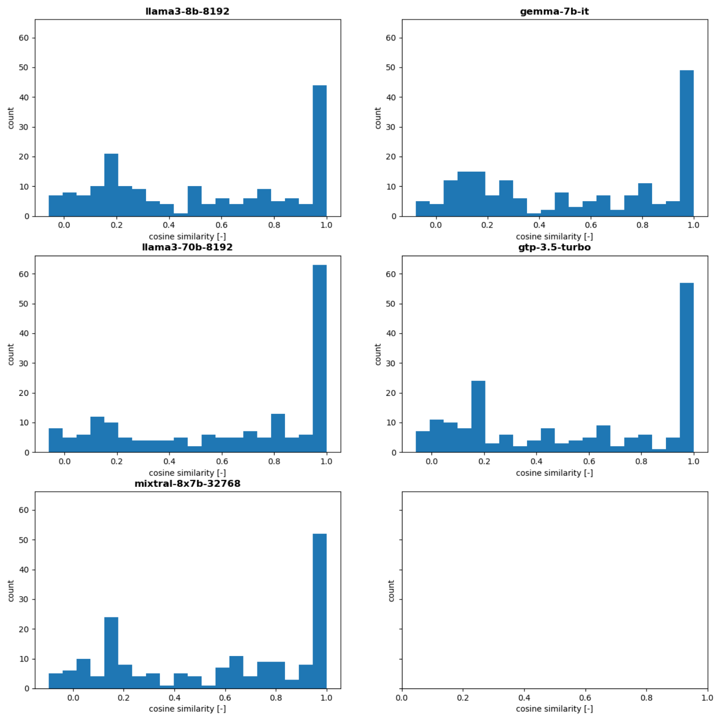


Figure 4.7: Histograms of cosine similarity averages across all fields. Reference model: GPT-4

Chapter 5
Conclusion

This work is a part of an ongoing development of a platform originally designed for NQL but counting on scalability to general counseling. I introduce the platform and argue its importance in the current society. Then I identify processes with potential for either partial or full automation. Subsequently I choose the task of form filling from dialogues obtained by the counseling feature. I clasiffy the problem as an extension of the well studied named entity recognition problem within the realm of natural language processing. Subsequently I introduce key deep learning methods which revolutionized natural language processing. In depth I discuss recurrent neural networks and transformer architectures as these are the pilot architectures. Following the theoretical exploration of the internal mechanisms of these architectures, I argue the importance of Large language models and their impact and usability for NLP tasks.

Subsequently I introduce the key large language models. I discuss they concrete architectures, training details, the differences in size and provide a summary of their performance on both human evaluated and common benchmark based metrics. To determine the usability of these models on our task an unsupervised evaluation pipeline based on calculating cosine similarity with respect to chosen reference models is proposed. Next a representative subset of the data provided by NQL is created by manually selecting and refurbishing relevant dialogues. Without any success a fine-tuning method is applied but subsequently removed from further evaluation for poor performance.

Following, six models are tested on the selected data using the proposed pipeline. A difficulty categorization of selected evaluation form fields is performed. The performance of the models is then assessed both in an overall and category-wise manner. Output reliability is also reviewed.

Based on these finding I propose the usage of Llama3-70B model for the platform to solve the problem at hand. If there is a need for computational efficiency thus forcing the selection of smaller model I recommend using Gemma over Llama-3-8B as the performance discrepancy is not big and Gemma provides better output reliability. If disk space is not a problem mixtral is a good option for a model with smaller inference cost. Generally I would not recommend GPT-3.5 as its performance is not satisfactory and the cost for hosting is rather high. If cost is not a problem GPT-4 is on par with the previously recommended Llama3-70B.

Bibliography

[1] P. Chomynová, K. Grohmannová, Z. Dvořáková, B. Orlíková, Z. Rous, and T. Černíková, "Souhrnná zpráva o závislostech v české republice 2022," in *Praha: Úřad vlády České republiky* (P. Chomynová, ed.), 2023.

[2] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A survey of large language models," 2023.

[3] K. K. Kedzior and L. T. Laeber, "A positive association between anxiety disorders and cannabis use or cannabis use disorders in the general population-a meta-analysis of 31 studies," *BMC psychiatry*, vol. 14, pp. 1–22, 2014.

[4] A. K. Davis, F. S. Barrett, D. G. May, M. P. Cosimano, N. D. Sepeda, M. W. Johnson, P. H. Finan, and R. R. Griffiths, "Effects of psilocybin-assisted therapy on major depressive disorder: a randomized clinical trial," *JAMA psychiatry*, vol. 78, no. 5, pp. 481–489, 2021.

[5] M. W. Johnson and R. R. Griffiths, "Potential therapeutic effects of psilocybin," *Neurotherapeutics*, vol. 14, pp. 734–740, 2017.

[6] J. C. Scott, S. P. Woods, G. E. Matt, R. A. Meyer, R. K. Heaton, J. H. Atkinson, and I. Grant, "Neurocognitive effects of methamphetamine: a critical review and meta-analysis," *Neuropsychology review*, vol. 17, pp. 275–297, 2007.

[7] I. Riezzo, C. Fiore, D. De Carlo, N. Pascale, M. Neri, E. Turillazzi, and V. Fineschi, "Side effects of cocaine abuse: multiorgan toxicity and pathological consequences," *Current medicinal chemistry*, vol. 19, no. 33, pp. 5624–5646, 2012.

[8] M. Carvalho, H. Carmo, V. M. Costa, J. P. Capela, H. Pontes, F. Remião, F. Carvalho, and M. d. L. Bastos, "Toxicity of amphetamines: an update," *Archives of toxicology*, vol. 86, pp. 1167–1231, 2012.

[9] A. Koopmann, E. Georgiadou, I. Reinhard, A. Müller, T. Lemenager, F. Kiefer, and T. Hillemacher, “The Effects of the Lockdown during the COVID-19 Pandemic on Alcohol and Tobacco Consumption Behavior in Germany,” *European Addiction Research*, vol. 27, pp. 242–256, 04 2021.

[10] C. Barbosa, A. J. Cowell, and W. N. Dowd, “Alcohol consumption in response to the covid-19 pandemic in the united states,” *Journal of Addiction Medicine*, vol. 15, no. 4, pp. 341–344, 2021.

[11] E. R. Grossman, S. E. Benjamin-Neelon, and S. Sonnenschein, “Alcohol consumption during the covid-19 pandemic: a cross-sectional survey of us adults,” *International journal of environmental research and public health*, vol. 17, no. 24, p. 9189, 2020.

[12] “Alcohol consumption per capita.” https://www.cia.gov/the-world-factbook/field/alcohol-consumption-per-capita/country-comparison/.

[13] “Graf - spotřeba alkoholických nápojů na 1 obyvatele v České republic,” Dec 2023. https://www.czso.cz/csu/czso/graf-spotreba-alkoholickych-napoju-na-1-obyvatele-v-ceske-republice.

[14] “National institute on alcohol abuse and alcoholism 2023,” Apr 2023. https://www.niaaa.nih.gov/publications/surveillance-reports/surveillance120.

[15] E. Comission, “State of health in the eu - czechia,” 2021. https://health.ec.europa.eu/system/files/2021-12/2021_chp_cs_english.pdf.

[16] H. Ritchie and M. Roser, “Alcohol consumption,” *Our World in Data*, 2022. https://ourworldindata.org/alcohol-consumption.

[17] O. W. i. D. IHME, “Global burden of disease study (2019) – processed by our world in data. “deaths that are from all causes attributed to smoking per 100,000 people, in both sexes aged age-standardized”,” 2019.

[18] S. Dattani, F. Spooner, H. Ritchie, and M. Roser, “Causes of death,” *Our World in Data*, 2023. https://ourworldindata.org/causes-of-death.

[19] L. Wilhelmsen, “Coronary heart disease: epidemiology of smoking and intervention studies of smoking,” *American heart journal*, vol. 115, no. 1, pp. 242–249, 1988.

[20] N. S. Godtfredsen, E. Prescott, and M. Osler, “Effect of smoking reduction on lung cancer risk,” *Jama*, vol. 294, no. 12, pp. 1505–1510, 2005.

[21] M. K. Underwood and S. E. Ehrenreich, “The power and the pain of adolescents’ digital communication: Cyber victimization and the perils of lurking.,” *American Psychologist*, vol. 72, no. 2, p. 144, 2017.

[22] A. Lenhart, R. Ling, S. Campbell, and K. Purcell, “Teens and mobile phones: Text messaging explodes as teens embrace it as the centerpiece of their communication strategies with friends.,” *Pew internet & American life project*, 2010.

[23] Denny, “Recurrent neural networks tutorial, part 1 – introduction to rnns,” Aug 2015.

[24] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” *arXiv preprint arXiv:1506.00019*, 2015.

[25] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, “How to construct deep recurrent neural networks,” *arXiv preprint arXiv:1312.6026*, 2013.

[26] I. Sutskever, *Training recurrent neural networks.* University of Toronto Toronto, ON, Canada, 2013.

[27] L. R. Medsker, L. Jain, *et al.*, “Recurrent neural networks,” *Design and Applications*, vol. 5, no. 64-67, p. 2, 2001.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[29] Denny, “Recurrent neural networks tutorial, part 3 – backpropagation through time and vanishing gradients,” Aug 2015.

[30] G. Strang, *Calculus v1.* OpenStax, 2020.

[31] A. Sherstinsky, “Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network,” *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[32] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[33] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 1310–1318, PMLR, 17–19 Jun 2013.

[34] S. Hesaraki, “Long short-term memory (lstm),” Oct 2023.

[35] A. Graves, *Long Short-Term Memory*, pp. 37–45. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling.," in *Interspeech*, vol. 2012, pp. 194–197, 2012.

[38] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, "Performance evaluation of deep neural networks applied to speech recognition: Rnn, lstm and gru," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 4, pp. 235–245, 2019.

[39] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[40] J. Chen, L. Yang, Y. Zhang, M. Alber, and D. Z. Chen, "Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[41] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Transactions of the association for computational linguistics*, vol. 4, pp. 357–370, 2016.

[42] D. Zeng, C. Sun, L. Lin, and B. Liu, "Lstm-crf for drug-named entity recognition," *Entropy*, vol. 19, no. 6, p. 283, 2017.

[43] N. Limsopatham and N. Collier, "Bidirectional lstm for named entity recognition in twitter messages," in *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pp. 145–152, 2016.

[44] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[45] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[46] J. P. Chiu and E. Nichols, "Named Entity Recognition with Bidirectional LSTM-CNNs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 07 2016.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[48] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, "Transformers: State-of-the-art

natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

[49] N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah, "A mathematical framework for transformer circuits," *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[51] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[52] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[53] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Transactions on nuclear science*, vol. 44, no. 3, pp. 1464–1468, 1997.

[54] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[55] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[56] R. Horev, "Bert explained: State of the art language model for nlp," Nov 2018.

[57] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[58] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[59] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," *arXiv preprint arXiv:1903.10676*, 2019.

[60] OpenAI, :, J. Achiam, and S. A. et. al, "Gpt-4 technical report," 2023.

[61] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023.

[62] H. Touvron and L. M. et.al, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[63] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, "Gqa: Training generalized multi-query transformer models from multi-head checkpoints," *arXiv preprint arXiv:2305.13245*, 2023.

[64] "Ai at meta$_2$024,″ *Apr*2024.

[65] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou, "A framework for few-shot language model evaluation," Sept. 2021.

[66] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," 2018.

[67] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?," 2019.

[68] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," 2021.

[69] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," 2022.

[70] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "WINOGRANDE: an adversarial winograd schema challenge at scale," 2019.

[71] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," 2021.

[72] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez, and I. Stoica, "Chatbot arena: An open platform for evaluating llms by human preference," 2024.

[73] A. Anthropic, "The claude 3 model family: Opus, sonnet, haiku," *Claude-3 Model Card*, 2024.

[74] G. Team, M. Reid, and N. S. et. al., "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," 2024.

[75] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[76] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, *et al.*, “Mixtral of experts,” *arXiv preprint arXiv:2401.04088*, 2024.

[77] G. Team, T. Mesnard, and C. H. et. al., “Gemma: Open models based on gemini research and technology,” 2024.

[78] J. Straková, M. Straka, and J. Hajič, “Neural architectures for nested ner through linearization,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Stroudsburg, PA, USA), pp. 5326–5331, Association for Computational Linguistics, 2019.

[79] S. Morwal, N. Jahan, and D. Chopra, “Named entity recognition using hidden markov model (hmm),” *International Journal on Natural Language Computing (IJNLC) Vol*, vol. 1, 2012.

[80] G. Szarvas, R. Farkas, and A. Kocsor, “A multilingual named entity recognition system using boosting and c4. 5 decision tree learning algorithms,” in *Discovery Science: 9th International Conference, DS 2006, Barcelona, Spain, October 7-10, 2006. Proceedings 9*, pp. 267–278, Springer, 2006.

[81] Z. Ju, J. Wang, and F. Zhu, “Named entity recognition from biomedical text using svm,” in *2011 5th international conference on bioinformatics and biomedical engineering*, pp. 1–4, IEEE, 2011.

[82] H. L. Chieu and H. T. Ng, “Named entity recognition with a maximum entropy approach,” in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pp. 160–163, 2003.

[83] M. Grinberg, *Flask web development: developing web applications with python.* " O’Reilly Media, Inc.", 2018.

[84] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” 2023.

[85] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.

[86] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-efficient transfer learning for nlp,” in *International conference on machine learning*, pp. 2790–2799, PMLR, 2019.

[87] F. Šarić, G. Glavaš, M. Karan, J. Šnajder, and B. D. Bašić, “Takelab: Systems for measuring semantic text similarity,” in ** SEM 2012: The First Joint*

Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012), pp. 441–448, 2012.

[88] J. Wang and Y. Dong, "Measurement of text similarity: a survey," *Information*, vol. 11, no. 9, p. 421, 2020.

[89] M. Vijaymeena and K. Kavitha, "A survey on similarity measures in text mining," *Machine Learning and Applications: An International Journal*, vol. 3, no. 2, pp. 19–28, 2016.

[90] K. A. Majorek, S. Dunin-Horkawicz, K. Steczkiewicz, A. Muszewska, M. Nowotny, K. Ginalski, and J. M. Bujnicki, "The RNase H-like superfamily: new members, comparative structural analysis and evolutionary classification," *Nucleic Acids Research*, vol. 42, pp. 4160–4179, 01 2014.

[91] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 11 2020.

[92] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings 8*, pp. 420–434, Springer, 2001.

Appendix A

Cosine similarity histograms

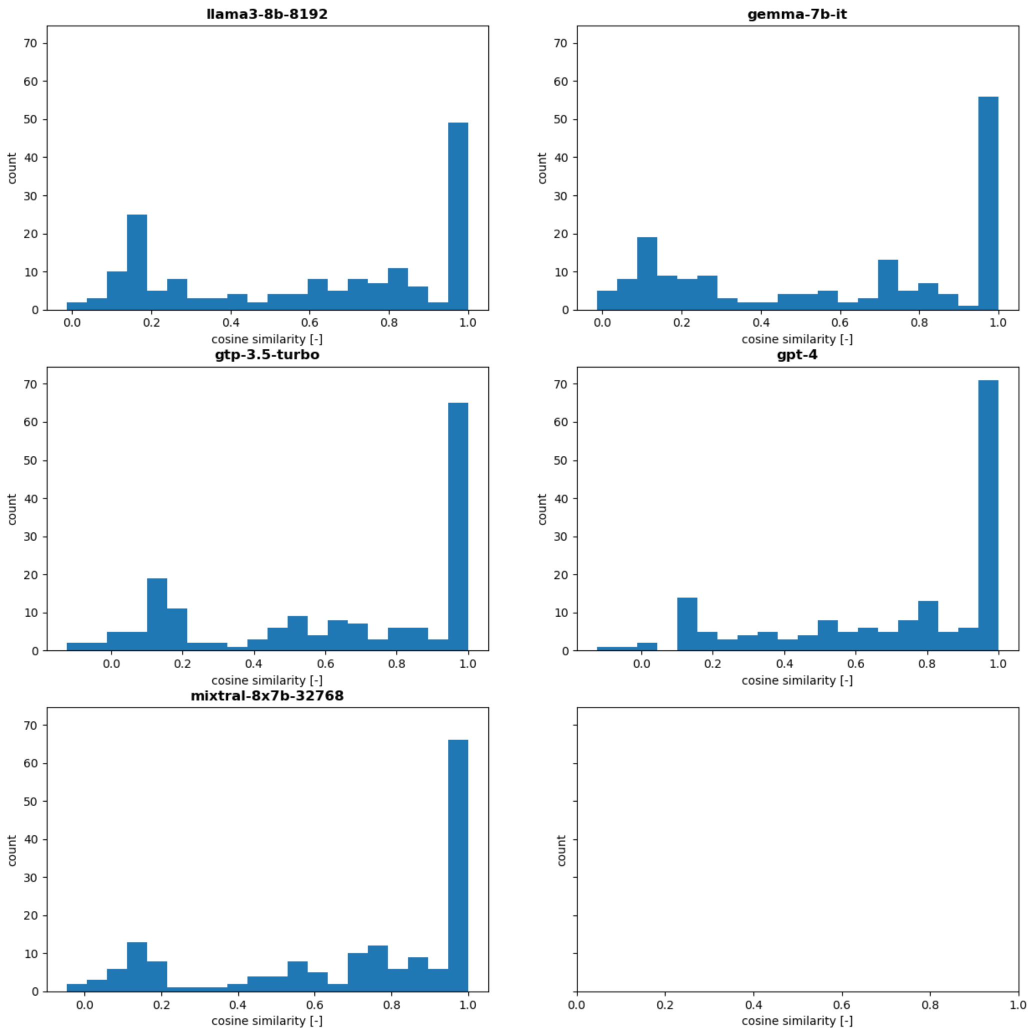


Figure A.1: Histograms of cosine similarity averages across all fields on filtered data. Reference model: Llama3 70B

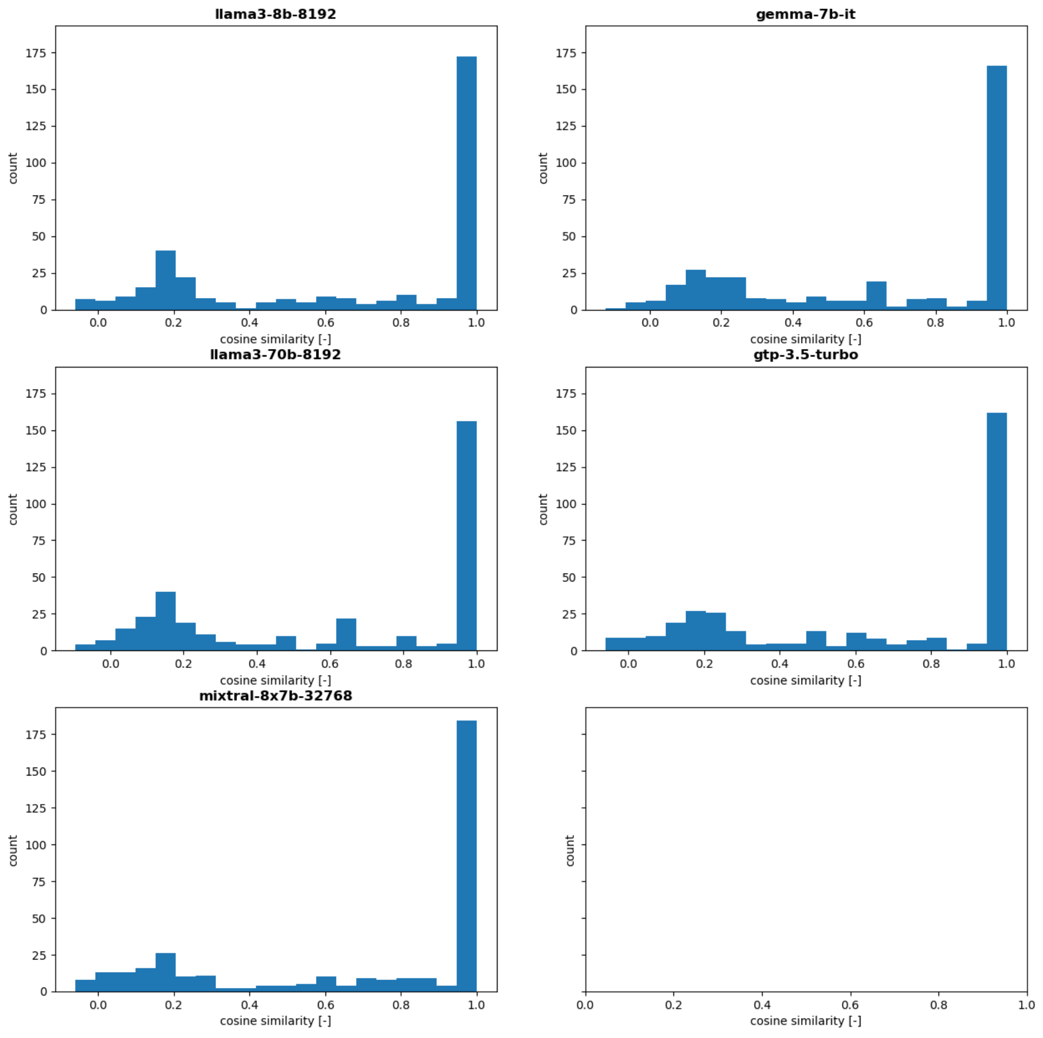


Figure A.2: Histograms of cosine similarity averages across all fields on unfiltered data. Reference model: GPT-4

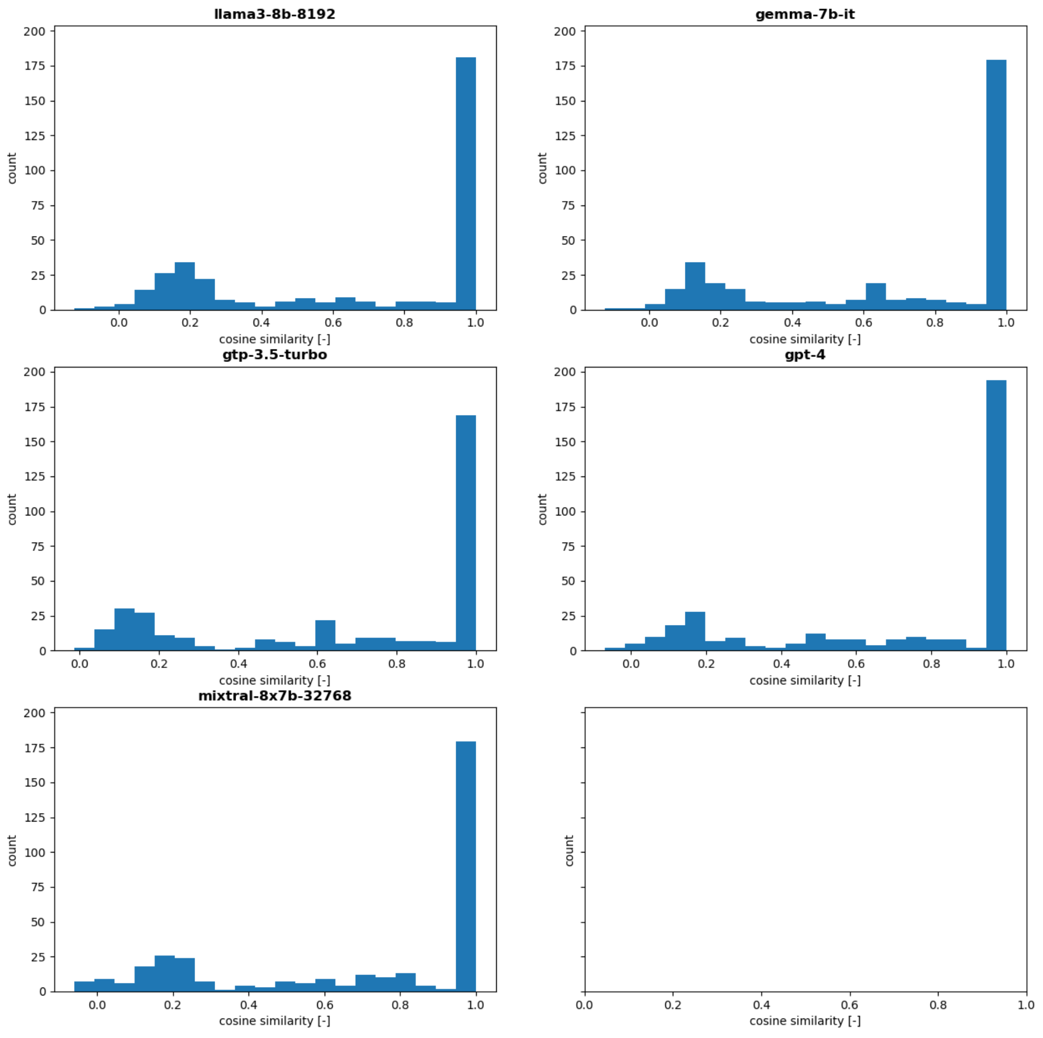


Figure A.3: Histograms of cosine similarity averages across all fields on unfiltered data. Reference model: Llama3 70B

Appendix B
Used software

B.1 Spell-checking

Overleaf built-spell check, Chat-GPT

B.2 Transalation and synonym generation

Chat-GPT, Abstract was translated to czech language.

Appendix C

Source code

Given the evaluation on sensitive data the code will only be made available upon request to the author. Source codes used to generate figure are not included.