POLITECNICO DI TORINO

EURECOM

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Personal Data Detection in Free Text

Supervisors:

Dr. Francesco Di Cerbo

Prof. Paolo Papotti

Prof. Giuseppe Rizzo

Candidate:

Gabriele Gioetto

Academic Year 2022/2023
Torino

Abstract

An HR (Human Resource) department in a large organization receives inquiries/requests from employees on multiple topics, quite different from one another. As an example, an employee can send requests dealing with health conditions, compensation/taxation, events of life (marriage, death of a relative...).
These data can be used for many different queries that can be useful for analysis purposes (Example: 'How many people have had COVID during 2021'). However, HR tickets typically contain personal data, that cannot be processed without the consent of the data subject according to the European privacy regulation (GDPR).
To be able to process documents with personal data, we can identify the pieces of information that qualify as personal data in a communication and subsequently anonymize such information using the appropriate techniques. A significant part of this problem is represented by the complex nature of personal data according to GDPR: personal data are defined as '*any piece of information that can be connected to an identified or identifiable natural person*' It comprises obvious identifiers like social security numbers, email addresses but also, elements like 'the Italian intern working for SAP in South of France'. To the best of our knowledge, it does not exist a public dataset of HR tickets that can be used to train machine learning models, the main reason being the difficult nature of these types of data. Synthetic data, which are artificial data that are generated from original data and a model that is trained to reproduce the characteristics and structure of the original data, follow a data protection by design approach. To address the need for a large dataset of HR tickets, we created a taxonomy of tickets, we found real data that can be used as support to create synthetic tickets and developed Ticket Generator: an application that can produce as many tickets as needed

belonging to different categories, we released a dataset of previously created tickets and we showcase some possible use cases of the dataset.

Acknowledgements

qualcosa

About SAP

SAP Founded in 1972, SAP has grown to become the world's leading provider of business software solutions. SAP is market leader in enterprise application software. The company is also the fastest-growing major database company. Globally, more than 77% of all business transactions worldwide touch an SAP software system.

With more than 347.000 customers in more than 180 countries, SAP includes subsidiaries in all major countries. SAP is the world's largest inter-enterprise software company and the world's third-largest independent software supplier, overall. SAP solutions help enterprises of all sizes around the world to improve customer relationships, enhance partner collaboration and create efficiencies across their supply chains and business operations. SAP employs more than 98.600 people.

Security Research at SAP Labs France, Sophia Antipolis

Based at SAP Labs France Mougins, Security Research Sophia-Antipolis addresses the upcoming security needs, focusing on increased automation of the security life cycle and on providing innovative solutions for the security challenges in networked businesses, including cloud, services and mobile. This internship is based in the SAP Labs France Research Lab, in Sophia-Antipolis. The work will be performed in the context of the Research Program "Security & Trust.

Table of Contents

Security Research at SAP Labs France, Sophia Antipolis v

List of Figures ix

Acronyms xi

1 Introduction 1

1.1 GDPR 1

1.2 Synthetic data 2

1.3 Ticket Generation 3

2 Related Works 4

3 Method 6

3.1 Taxonomy 6

3.2 Datasets 8

3.2.1 Datasets preprocessing 9

3.2.2 Bayesian Network 10

3.3 Ticket Generation 12
3.4 GPT-J 19
3.5 Survey 28

Bibliography 30

List of Figures

3.1 Initial Taxonomy 6

3.2 Final taxonomy, reduced due to unavailability of data 7

3.3 Input Saliency: First token 16

3.4 Input Saliency: Subject 17

3.5 Input Saliency: HR 17

3.6 Neuron Activation Analysis 18

3.7 Non-negative Matrix Factorization on Activation Matrix . . 19

3.8 GPT-3 and GPT-J architectures compared 25

3.9 Byte-Pair Encoding Tokenization Example 26

3.10 Implementation of RoPE, Image taken from original paper . 28

3.11 Example of survey 29

Acronyms

Chapter 1

Introduction

With the advent of deep learning a lot of application have been needing more and more data. . However, more often than not, these data contains a large amount of sensitive and personal information that restricts its use according to the legal framework in place in many countries.
Even in companies' internal data, massive amounts of sensitive information are growing, limiting its processing and sharing. It is considered that, if the information reveals the identity of a person then it threatens the personal rights of this person, and can only be processed with special attention in compliance with the legal framework.
This is especially relevant in the context of HR tickets, which can contain not only personal information but also special categories of personal data, which need additional layers of protection according to GDPR.

1.1 GDPR

The General Data Protection Regulation is a privacy regulation that regulates the processing of personal data 'wholly or partly by automated means and to the processing other than by automated means'. The regulation applies to all citizens of the EU and to all data subjects in the EU, whether the processing is carried out inside or outside the EU.

Personal data are defined as 'any information relating to an identified or identifiable natural person'. The regulation states that 'an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person'. The natural person can be identified both by direct identifiers and quasi identifiers. The direct identifiers are information that directly identify the person, such as the telephone number, the social security number..., while the quesi identifiers are information that alone cannot identify a person, but if they are combined with other quasi identifier can affect a person's privacy. For example the job title could be not enough to identify a person, but combined with his/her company and his/her nationality could be.
Moreover, the GDPR treats some categories of personal data more carefully. These categories are called 'special categories' and include racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, genetic data, biometric data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural person's sex life or sexual orientation
The special categories of personal data cannot be generally processed, with some exceptions including 'the data subject has given explicit consent to the processing of those personal data for one or more specified purposes'.

1.2 Synthetic data

Synthetic data is artificial data that is generated from real data and has the same statistical distribution of the original data. This means that synthetic data and original data should deliver very similar results when undergoing the same statistical analysis.
Synthetic data has many benefits over real data: if you create a model that generates synthetic data you can generate how many data you need, you can infer certain properties to your data (for example it can be useful for bias and fairness research) and above all, synthetic data can respect the right to personal data protection. However, it is not always guaranteed that

synthetic data is privacy preserving: it has been shown that synthetic data can leak personal information [1].

1.3 Ticket Generation

We created a tool that can be used to create unlimited amount of synthetic HR tickets and we published a dataset of 16000 tickets. Each ticket has a category, sub-category and the entities of the ticket. The tickets are not created from scratch, but starting from a dataset of real data and some prompts that help the model generate the text.
We did a survey internal to the company to gather some real data, and we changed the tickets generation parameters in order to respect the real data with respect to different text metrics.
Finally, we showed different possible use cases for the dataset:

- Ticket Anonymization
- Ticket Classification
- Named Entity Recognition on tickets' entities
- Ticket Summarization

Chapter 2

Related Works

As far as we know, it has never been published a dataset of HR tickets written by company employees, due to the sensitive nature of the data. We believe this is the case not only because of the GDPR laws, but also because these data can be damaging both for the company and the employees in some cases (For example an employee criticising the working environment or leaking some personal information in a ticket).
However, there are some datasets that can resemble what we are trying to build, with similar language style and intents:

- *Consumer Complaint*: Consumer Financial Protection Bureau's online database of customer complaints about different financial products. The overall dataset contains over 600,000 complaints and each record has the complaint's text description, the product that is the cause of the complaint, the company which issues the product, and the category of the issue. In the dataset the personal information are masked.
- *Enron Mail*: The *Enron Mail* dataset contains contains about 0.5M emails of 150 senior management from the Enron corporation. This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation for fraud. The corpus is one of the only few publicly available mass collections of real emails easily available for study.
- *TAB*: the Text Anonymization Benchmark corpus[2] is a privacy-oriented

annotated text resource. The corpus comprises 1,268 English-language court cases from the European Court of Human Rights (ECHR) enriched with comprehensive annotations about the personal information appearing in each document, including their semantic category, identifier type, confidential attributes, and co-reference relations.

Chapter 3

Method

3.1 Taxonomy

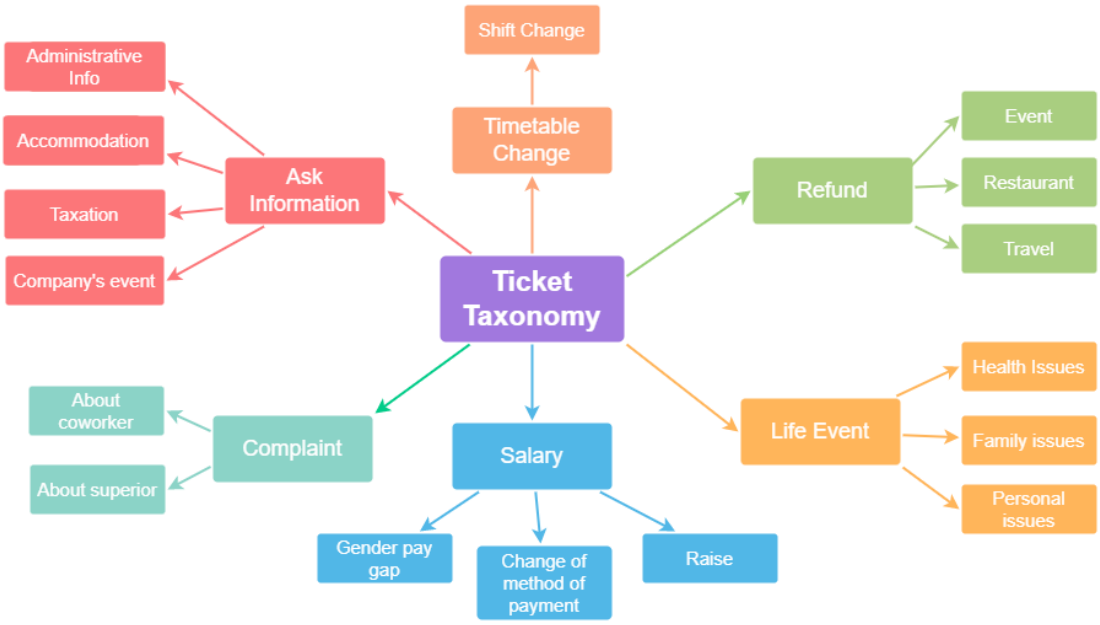


Figure 3.1: Initial Taxonomy

Usually HR tickets can belong to various categories, which can range from a request of shift change to a complaint about a superior.

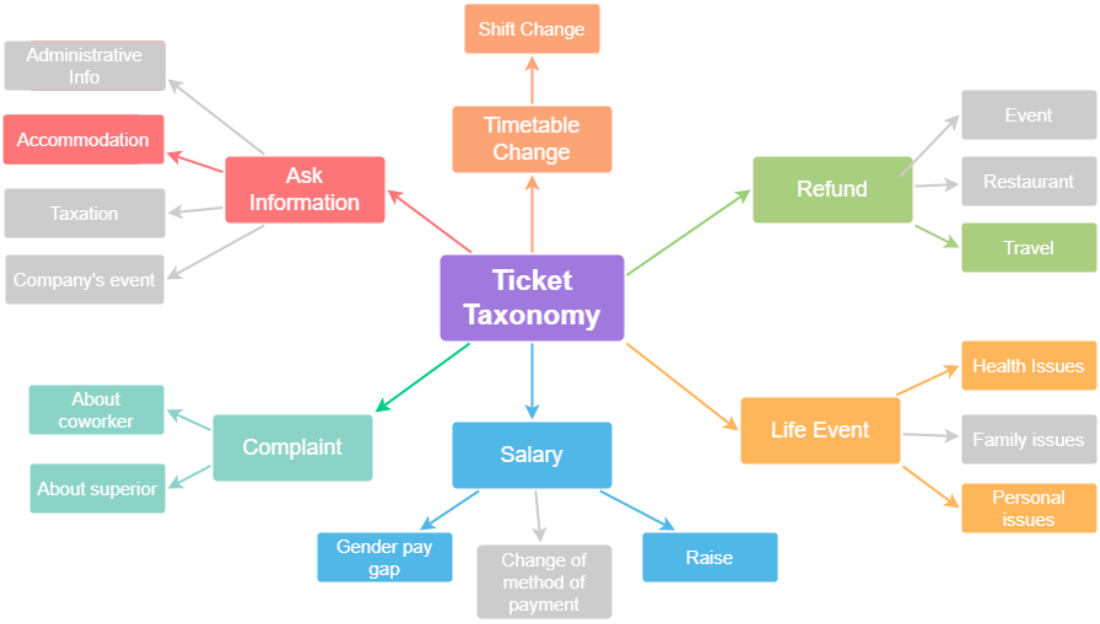


Figure 3.2: Final taxonomy, reduced due to unavailability of data

We built a taxonomy of tickets, which is structered in categories and sub categories. Each sub category has its own variables that are used as inputs for the ticket generation. The variables are sampled from the existing datasets mentioned before.
For example to create a request of sick leave, we pass as inputs the reason and the number of days of sick leave requested. Moreover, each category has distinct templates and prompts. The taxonomy (Shown in Figure 3.1) has been built following the advise of an HR expert from SAP, however the final taxonomy (Shown in Figure 3.2) presented here is a subset of the original one due to the unavailability of public data on certain topics (Ex. *Work benefits*)
The final complete taxonomy and the complete list of all variables used for each category/sub-category can be seen in the Table 3.1

Category	Sub-category	variables
Ask Information	Accomodation	location, duration
Complaint	About coworker	complaint, reason
	About superior	complaint, reason
Timetable change	Shift change	reason_of_change, old_date, new_date
Salary	Salary raise	old_salary, new_salary, increase,work_title
	Gender pay gap	wage_gap
Life Event	Health issues	disease, number_of_days_of_sick_leave
	Personal issues	issue, number_of_days
Refund	Travel	from, to, date_travel

Table 3.1: Table of all defined categories and sub-categories with their respective variables

3.2 Datasets

In order to generate tickets, we decided to use real data as a starting point to make them as much realistic as possible. Another reason to use real data is that it makes the dataset useful for use cases such as anonymization.
The dataset that we used are all public and available online. In some cases where no datasets were available, we created them manually from personal experience.
The datasets are:

- *Absenteeism at work Data Set*: contains records of work absences, with the reason of the absence (almost always a disease) and the number of hours of absence. It is used to create the requests of days off for health reasons
- *National Occupational Employment and Wage Estimates United States*: estimates of wages in the US calculated with data collected from employers in all industry sectors in metropolitan and nonmetropolitan areas in every state and the District of Columbia. It is used to create the requests of salary raise.
- *List of events of life*: list of major events in life. It is used to create the requests of time off due to personal reasons.
- *Gender pay gap in the UK*: dataset of employers with 250 or more

employees, comparing men and women's average pay across the organizations. It is used to create the requests of explanation for the wage gap amongst genders.

- *OpenFlights database*: datasets of airports and flights all over the world. It is used for the requests of refund of travels.

- *Geonames all cities with a population over 1000*: datasets of all cities of the world with a population over 1000 people. It is used for the requests of information about accommodation.

3.2.1 Datasets preprocessing

The *Absenteeism at work Data Set* is the only dataset that contains data about people that is not already grouped and averaged. This means that there is a record in the dataset for each employee request, which contains the personal information of the employee, the reason of the absence and the time of absence in hours. For privacy reasons, we used a Bayesian Network. A Bayesian network is a probabilistic graphical model that measures the conditional dependence structure of a set of random variables based on the Bayes theorem. The features that we have used to build the Bayesian network are the *reason for absence*, the *month of absence* and the *time of absence*.
Using the *Absenteeism at work Data Set* we learn conditional probability distributions from data, to which we add a Laplace noise for differential privacy. Then we can sample new data that follow the original distributions, but that are not equal to the original ones.
The absence reasons in the dataset are given as ICD(International Classification of Diseases) codes, to make them more human readable, we picked for each ICD code the corresponding more frequent diseases.
In the *National Occupational Employment and Wage Estimates United States* dataset, we sample employees based on the number of people employed in a certain field. Therefore for example since 'Retail Sales Workers' consists of the 5.4% of the total occupations, then the sampled employee will have the 5.4% of possibility to have as occupation 'Retail Sales Worker'.
The current salary of the employee is calculated adding a Gaussian noise to the average salary of the employee's occupation, and then the salary raise requested is picked randomly between 5%-10%.

The ranges of wage gap, used in the tickets regarding explanation for the gender wage gap in the company, are sampled randomly from the dataset *Gender pay gap in the UK*, adding a Gaussian noise for privacy reasons.
To sample the cities for the requests of accomodation, we randomly sample from all the cities with more than 100,000 inhabitants from the country of residence of the synthetic employee. To calculate the duration of the accommodation we pick a random number of months between 1 and 12.
To get data for the category type 'refund travel', we sample randomly one flight from all the flights leaving from the country of the synthetic employee. The data are taken from the *OpenFlights database*.
The complaints about coworkers and superiors and the life events that can affect the work life of a person were written by myself, using primarely personal experience and some help from internet blogs.

3.2.2 Bayesian Network

A bayesian network is a machine learning method which combines a probabilistic graphical model with Bayesian inference to infer the likelihood of certain events or outcomes. It is used to find relationships between variables and to identify which variables are most influential in predicting a certain outcome or event.
The bayesian network is based on the Bayesian theorem:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

Formally, the Bayesian network is a directed acyclic graph G = (V,E) with

- A feature for each node i belonging to V
- A conditional probability distribution for each edge, so the edge from feature i to j represents $p(x_j|x_i)$

The base version of a Bayesian network works with discrete variables, however there are also implementations that consider also continuous variables [3]

Building a Bayesian network starting from the *Absenteeism at work Data Set* is relatively easy, we calculate the likelihood distribution $p(x_i|x_j) \forall x_i, x_j \in D$, where D is the set of features. As a prior we used a Dirichlet distribution, mainly because it is the conjucate prior of the categorical distribution.
We then added pseudocounts to the observed counts in the data used to calculate $p(x_j|x_i)$. This technique is used to diminish the overfitting of data. The values we used for pseudocount is $\gamma = 1$.
Since we learn the conditional probability distribution from our data, the structure of the network or the conditional probabilities may therefore leak some information on an individual in the dataset. In order to provide strong privacy guarantees and minimize the re-identification risk, we leverage the notion of differential-privacy: we perturb the data adding a noise sampled from a Laplace distribution

$$z \sim Laplace\left(0, \frac{2 \cdot n_{features}}{\gamma \cdot \epsilon}\right)$$

where ϵ is the privacy budget for differential privacy, which controls the anonymization level.
Differential privacy is a rigorous mathematical definition of privacy. An algorithm is said to be differentially private if by looking at the output of an algorithm A performed on a dataset X, one cannot tell whether any individual's data was included in the original dataset or not. In other words, the guarantee of a differentially private algorithm is that its behavior hardly changes when a single individual joins or leaves the dataset. The mathematical definition of differential privacy is:

$$Pr[A(X) \in Z] \leq e^{\epsilon} \cdot Pr[A(X') \in Z]$$

where A is an algorithm and X' is a neighbour dataset of X. A dataset is neighbour of another dataset if they differ by only one record.
Once the private Bayesian network is built, we can sample new values for all the nodes in the graph. These generated values will have the same distribution and preserve the consistency and statistical properties of the original dataset up to the noise addition which acts as a de-identification barrier.

3.3 Ticket Generation

For each HR ticket, we create a synthetic employee. For all tickets' categories, the employee has some common features: *name*, *first name*, *last name*, *nationality*, *country*, *email*, *company*, *company's email* and *ticket date*.
All these information are created exploiting the Python library *Faker*. The nationality and the company's country are selected from the extendible list { USA, Germany, Italy, Spain, France }. All other information are created accordingly to the country picked. So for example if the country of birth of the employee is Italy, then the generated name will be Italian.
Then, once the employees are generated, the information specific to the ticket category, created starting from the open datasets as mentioned before, are concatenated to the general information of the employees.
For each ticket category there are distinct templates. In each template there is an initial part that contains the general information of the employee, such as name, surname, company..., then some prompts correlated to the category of the ticket and then the textual prompt.
Here's a couple of examples of templates:

Request of time off due to health reason:

> From: ${email}
> To: ${company email}
> First name: ${first name}
> Last name: ${last name}
> Company: ${company}
> Date: ${ticket date}
> Ticket category: ${category}
> Ticket sub-category: ${sub category}
> Date start absence: ${date start absence}
> Reason absence: ${reason}
> Subject: Request for sick leave for ${number_of_days}
>
> Dear Sir/Madame, my name is ${name} and I work at ${company}. I am requesting <*generate*>. I hope <*generate*>.

Request of refund of travel:

> From: ${email}
> To: ${company email}
> First name: ${first name}
> Last name: ${last name}
> Company: ${company}
> Date: ${ticket date}
> Ticket category: ${category}
> Ticket sub-category: ${sub category}
> Date Travel: ${date_travel}
> From: ${airport_from}, ${from}
> Destination: ${airport_to}, ${to}
>
> Hello, my name is ${name}. I am writing this mail to ask a refund for the travel <*generate*>

The variables are replaced with the features of the employee, whereas the <*generate*> are replaced with text generated by a generative model. The part generated by the generative model are created in a recursive way. This means that the first <generate> is replaced with text generated automatically using as prompt everything that precedes it. Then the second <generate> will have as prompt the entire ticket, including the text generated previously by the model. The model is forced to generate some text, if no text is generated in an iteration, the process is repeated until the model gives a non empty output.
A general schema of the generation is showed in **??**

Templates

The model takes in a prompt, or context, and generates text from it. The prompts typically take the form of a few sentences or a paraphrase, and the model generates sentences that fit the context of the prompt. By manipulating the prompt, users can generate text of various tones, topics, and styles.

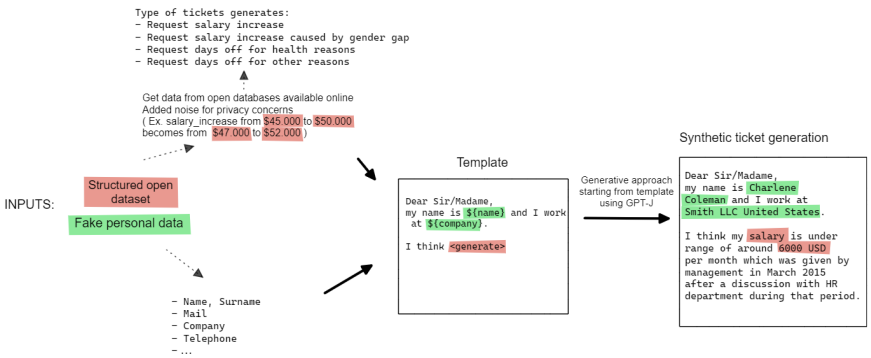


Figure 3.3: Schema of Ticket Generation

The GPT model is also capable of completing tasks such as question answering, machine translation, and summarization in an unsupervised manner[4]. By providing a prompt with the task and context, the model can generate accurate results that address the specific context and task. Tasks such as summarization, for instance, require a prompt to provide the model with the information it needs to summarize the text accurately.
Changing the prompt of GPT can change the tone, topics, and style of the generated text. Depending on the prompt, GPT can generate text ranging from creative stories to technical summaries. The tone and style of the text can range from humorous to academic, depending on the prompt. As the prompt changes, the model will also adjust to reflect the context of the prompt.
This is why we took inspiration from the e-mail format, as HR tickets are created in a working environment where a formal language is used and often they resemble mails in the tone and in the topics.
In particular, the Enron Emails dataset is included in the Pile dataset, the dataset which GPT-J is trained on.
The emails of the Enron dataset have usually a well-structured prompt, here's an example:

> Message-ID: <16593073.1075858228177.JavaMail.evans@thyme>
> Date: Wed, 12 Jan 2000 00:29:00 -0800 (PST)
> From: carrie.hollomon@enron.com
> To: phillip.love@enron.com
> Subject: Workhours
> Mime-Version: 1.0

Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Hello ...

To mimic the format of the emails of the Enron Dataset, we kept the From, To, Date and Subject rows. Instead we removed the Mime-Version, the Content-type and Content-Transfer-Encoding, because after conducting some experiments it was evident that they did not help achieving better results, on the contrary in some cases they were worse.
In addition to the standard information, we added explicitly some rows with additional information specific to the category, rather than including them only in the subject or in the initial prompt.

In order to generate tickets that were dissimilar and covered a wide range of topics/tokens, we preferred giving to GPT small text prompts and let the model generate most of the text getting the information from the email-like prompt. This approach allowed to improve the diversity of the dataset and, consequently, the usefulness of it.

Small text prompt example of a request of shift change:

Dear Sir/Madame, my name is ${name}. I wanted to <*generate*>

Long text prompt example of a request of shift change:

Dear Sir/Madame, my name is ${name} and I work at ${company}. I wanted to ask to change the shift from ${old_date} to ${new_date} in order to be able to <*generate* >

Architecture analysis

To understand better how the model was behaving and why it gave certain types of outputs rather than others, we used the python library Ecco, which creates interactive visualization that show at which layers of the architecture the final token has been decided, which input tokens contribute the most for a prediction, ...

In particular, we used two different methods:

- Input Saliency: used to show how much did each input token contribute to producing the output token
- Neuron Activation Analysis: used to examine underlying patterns in neuron activations using non-negative matrix factorization

Input Saliency

To get a better grasp of the most useful information of the prompt, and to understand how we could modify it to achieve better results, we used Gradient * input (Shrikumar et. al). This technique calculates the partial deratives off the output of the model and multiply them with the input itself. Then the inputs with the highest scores are the one that influenced the most the generation of the new tokens

$$score = x_i \bigtriangledown f(x_i)$$

where f is the architecture output.
The main problem with the Gradient * input method is that only one input is considered. Integrated Gradients solve this issue, computing the average gradient while the input varies along a linear path.

$$score = (x_i - x_i') \int_{\alpha=0}^{1} \bigtriangledown f(x' + \alpha(x - x'))\, d\alpha$$

Nevertheless, we used the Gradient * input method for computational issues (The Integrated Gradients took too much RAM of the GPUs).
In the images under we underline some of the considerations we have done while defining the prompts.

Neuron Activation Analysis

The Feed Forward Neural Network layer is one of the major components inside a transformers block. To better understand how the neurons of different layers were 'activated' and how the neurons contributed towards

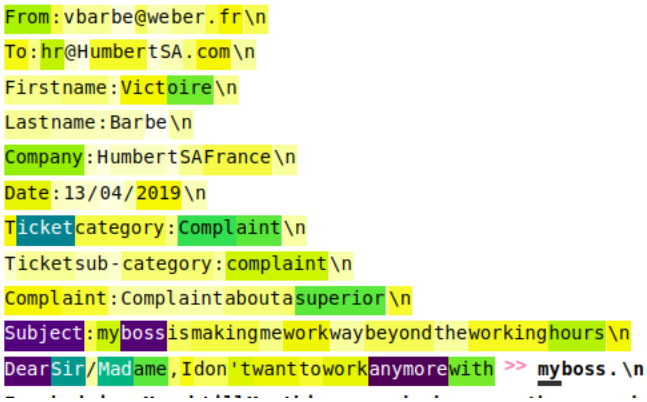


Figure 3.4: Input Saliency: First token

To create the first token, the model focus on the elements that resembles the content of an email, in particular on the fact the it is indeed a 'Ticket'

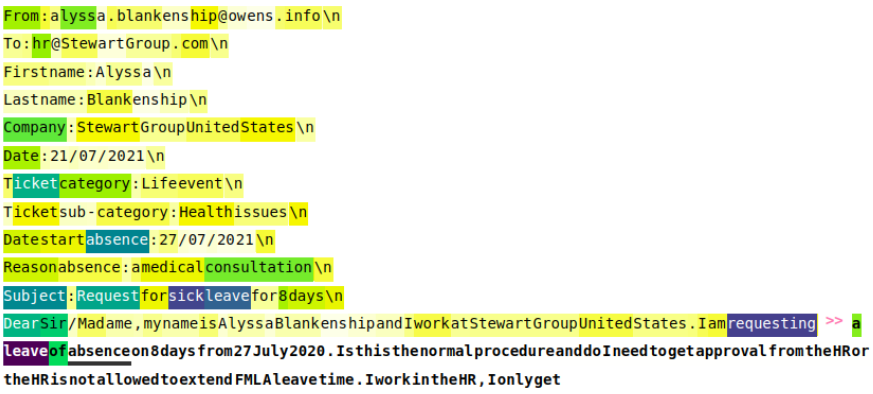


Figure 3.5: Input Saliency: Subject

Clearly stating the subject of the ticket helps the model creating tokens that are on the right topic (In this example 'days of absence')

each generated token, we exploited the Factor Analysis provided by Ecco. Firstly, Ecco calculates the activation scores of each neurons over all layers, and then uses Non-negative Matrix Factorization3.7 to do dimensionality reduction on the matrix of activations, which will be reduced to a matrix $M \times T$, where M is a parameter we decide and T is the number of tokens, starting from a matrix $(n \cdot N) \times T$, where n is the number of neurons per layer and N is the numbes of layers. In GPT-J $n = 16384$ and $N = 28$. Then, for each factor, which are the dimensionality reduced layers, we visualize their activated neurons. From the example in the Figure 3.6, we

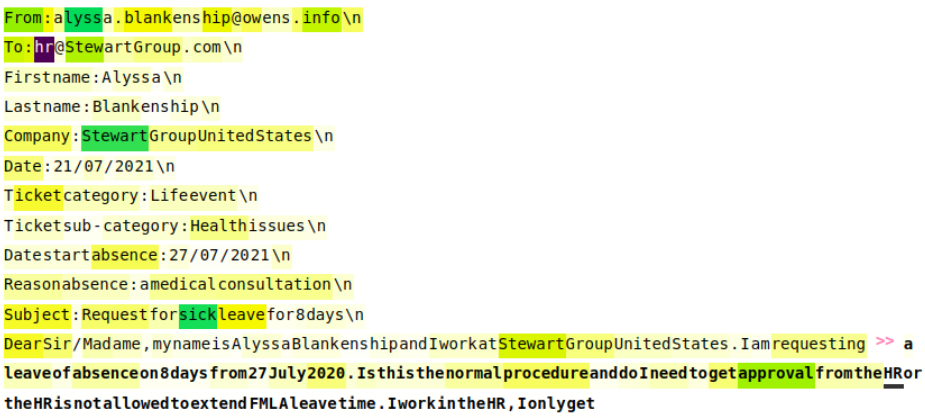


Figure 3.6: Input Saliency: HR

Clearly stating that we are writing to hr can help the model know the context, and consecutively use a certain language, specific words...

can show for each factor what their main contributions are:

1. Punctuation
2. Mail prompt (From, To, First Name, Last Name, Company...)
3. New lines
4. Dates
5. First token, common across various GPT factors
6. Medical terms (generally terms related to the tickets' topic)
7. People's names and contacts
8. Company's name and contacts
9. Ticket's subject (Category, Sub-category and additional info)
10. Email presentation ('Dear Sir/Madame...')

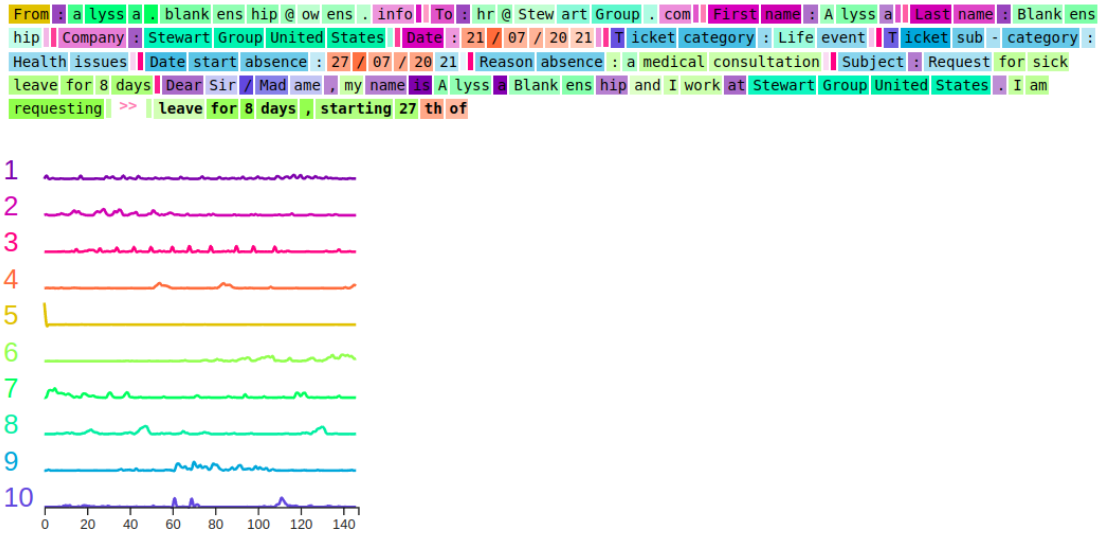


Figure 3.7: Neuron Activation Analysis

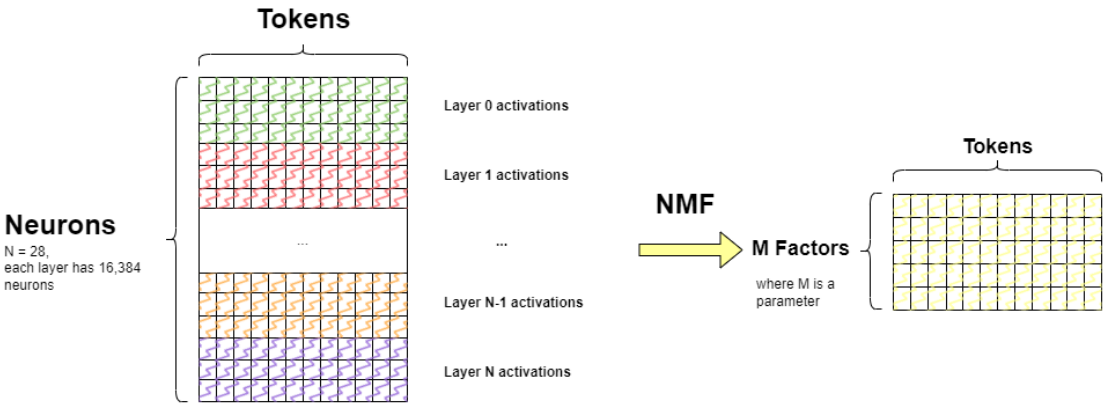


Figure 3.8: Non-negative Matrix Factorization on Activation Matrix

3.4 GPT-J

The generative model used to create the tickets is GPT-J, an open source 6 billion parameter, autoregressive text generation model trained on The Pile dataset released by EleutherAI.
The Pile dataset[5] is an 825 GiB English text corpus composed by 22 diverse subsets, which can be grouped in 5 categories:

- Academic (*ArXiv*, *PubMed Central*, ...)
- Internet (*Wikipedia*, *StackExchange*, ...)
- Prose (*Bibliotik*, ...)
- Dialogue (*Subtitles*, ...)
- Misc (*Github*, ...)

The parameters used for the generation of the next token are:

- *min_length*: minimum number of words created by a gpt generation
- *max_length*: the max length of the sequence to be generated.
- *top_k*: the k most likely next words are filtered and the probability mass is redistributed among only these k words.
- *top_p*: the next words are sampled from the smallest possible set of words whose cumulative probability exceeds the probability p.
- *temperature*: the value T used to module the logits distribution. The higher the value of T the higher the entropy of the logits distribution will be
$$p_i = \frac{exp(x_i/T)}{\sum_j exp(x_j/T)}$$
- *repetition_penalty*: the parameter θ for repetition penalty. 1.0 means no penalty
Given a list of generated tokens G,
$$p_i = \frac{exp(x_i/T \cdot I(i \in G))}{\sum_j exp(x_j/T \cdot I(j \in G))}$$
$$I(c) = \theta \text{ if } c \text{ is True else } 1$$
So the logits distribution of the token change based on if the token has already been generated before.

- *length_penalty*: *length_penalty* > 0 promotes longer sequences, while *length_penalty* < 0 encourages shorter sequences.

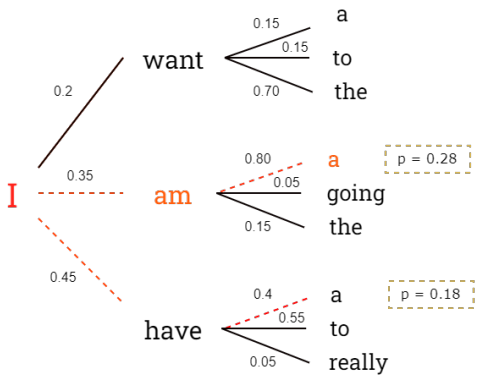
- *no_repeat_ngram_size*: If set to int > 0, all ngrams of that size can only occur once

  Dear Marie, I would like to have some days off. I

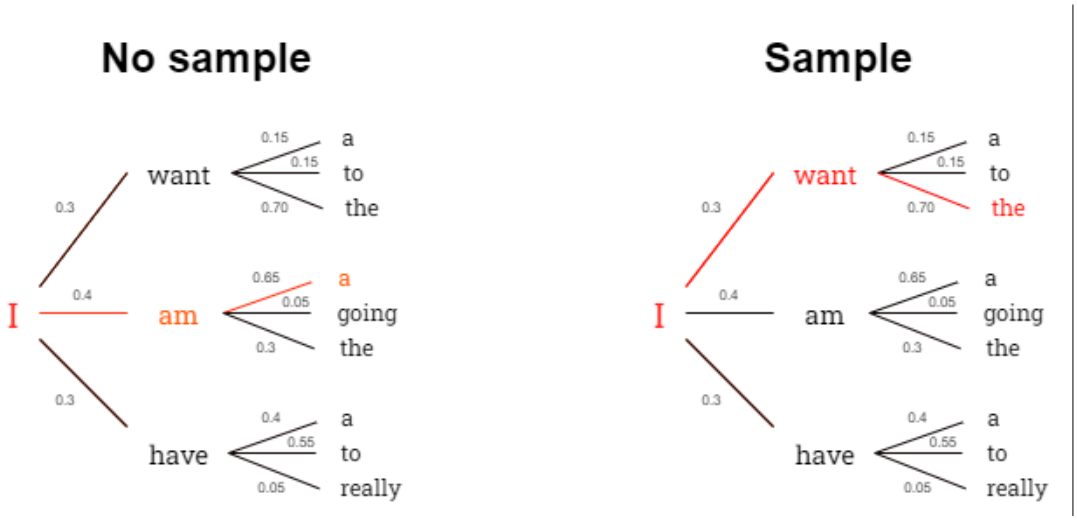
  Dear Marie, I would like to have some days off. I

  Ex: *no_repeat_ngram_size* $= 2$

- *num_beams*: Number of beams for beam search. They beams are the number of 'paths' that are considered when choosing the next token



- *do_sample*: If set to 'False' greedy decoding is used (the most probable token is always chosen). Otherwise, sampling is used (the next token is chosen sampling from the distribution of possible next tokens)



- *bad_words*: List of words that are not allowed to be generated by the model.
- *force_words*: List of words that must be generated in the generation.

The default values assigned to the parameters used are shown in Table 3.2

Parameter	Value
min length	0
max length	50
top k	50
top p	0.85
repetition penalty	1.2
temperature	1
length penalty	1
no repeat ngram size	0
num beams	1
do sample	True
bad words	[ ]
force words	[ ]

Table 3.2: Parameters of GPT-J model

The GPT architecture is based on the original Transformers paper, which introduced two types of transformers blocks: the encoder block and the

decoder block. GPT is assembled by a stack of decoder blocks, which are composed by:

- Normalization Layers: a normalization layer [6] normalize all inputs of a neural network across their features. It has been shown that Layer normalization enables smoother gradients, faster training, and better generalization accuracy [7]

  x: data sample
  d: dimension of data sample
  y: output of LayerNorm
  ϵ: small number added for stability

$$u = \frac{1}{d}\sum_{i=1}^{d} x_i$$

$$\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x\_ui)^2$$

$$\hat{x_i} = \frac{x_i - u}{\sqrt{\sigma^2 + \epsilon}}$$

$$y = \gamma\hat{x_i} + \beta$$

  where γ and β are parameters that the model learns.

- Masked Self-Attention Layer: attention is a mechanism that allows neural networks to assign a different amount of weight to each token in a sequence and process each token as a weighted average of all other tokens.
  In practice three matrixes are calculated:

  - Q (Query): the representation of the current token
  - K (Key): the representation of all the other tokens, which are matched with the current token
  - V (Value): the representation of all the words, used for the weighted-average

  The Q, K and V matrixes are initialized as:

  - $Q = W_q X + b_q$

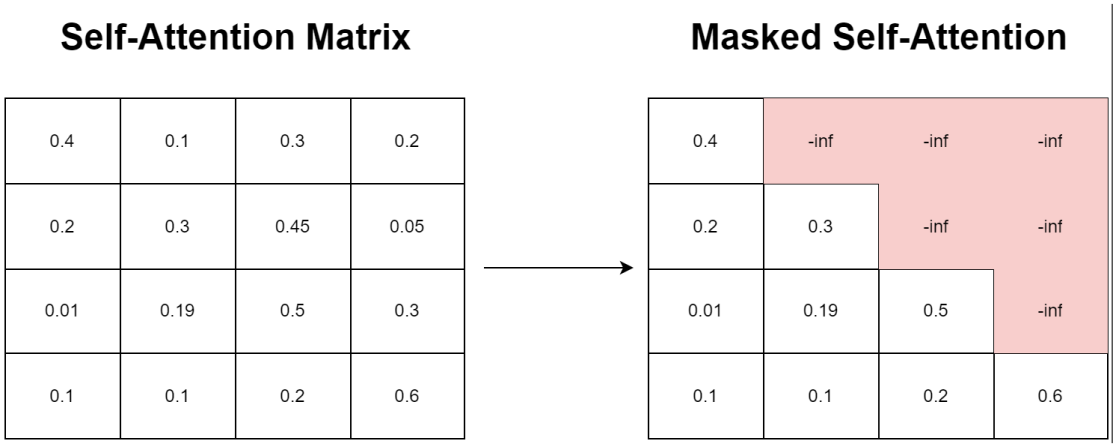
- $K = W_k X + b_k$
- $V = W_v X + b_v$

where X is the input matrix and the other matrixes are randomly initialized and learned by the model. Finally, the attention score is calculated with

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where d is a normalization factor equivalent to the embeddings dimension The masked self-attention is a modified version of self-attention where all the tokens that appear after the current one are set to 0, in order not to let the model being influenced by any information regarding the tokens at the next positions. This is fundemental when training generative models such as GPT, whose scope is to predict the successive tokens.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T + Mask}{\sqrt{d}}\right)V$$



- Feed Forward Neural Network Layer: Used to add non-linearity to the transformer block

GPT models use a Byte-Pair Encoding tokenization. The Byte-Pair Encoding algorithm starts by building a vocabulary with all the single characters of the corpus we are training on. Then, at each step of the algorithm, the two tokens which appear consecutevily the most in the words

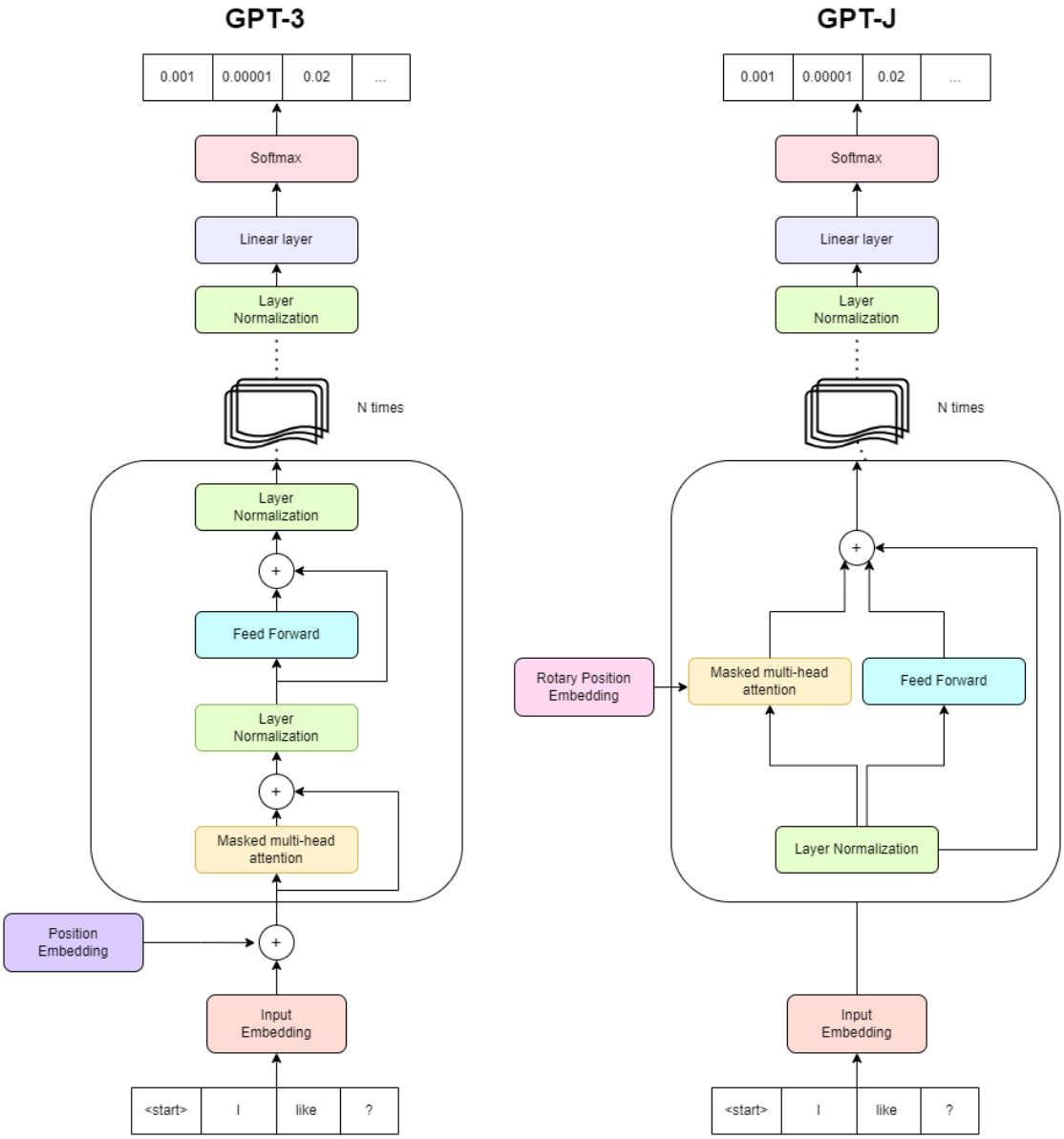


Figure 3.9: GPT-3 and GPT-J architectures compared

of the corpus are unified and create a new token. This process continue until the desired vocabulary lenght is satisfied. An example taken from the site [8] is shown in the Figure 3.9

Compared to GPT-3, GPT-J[9] has two minor architectural differences (shown in Figure 3.8):

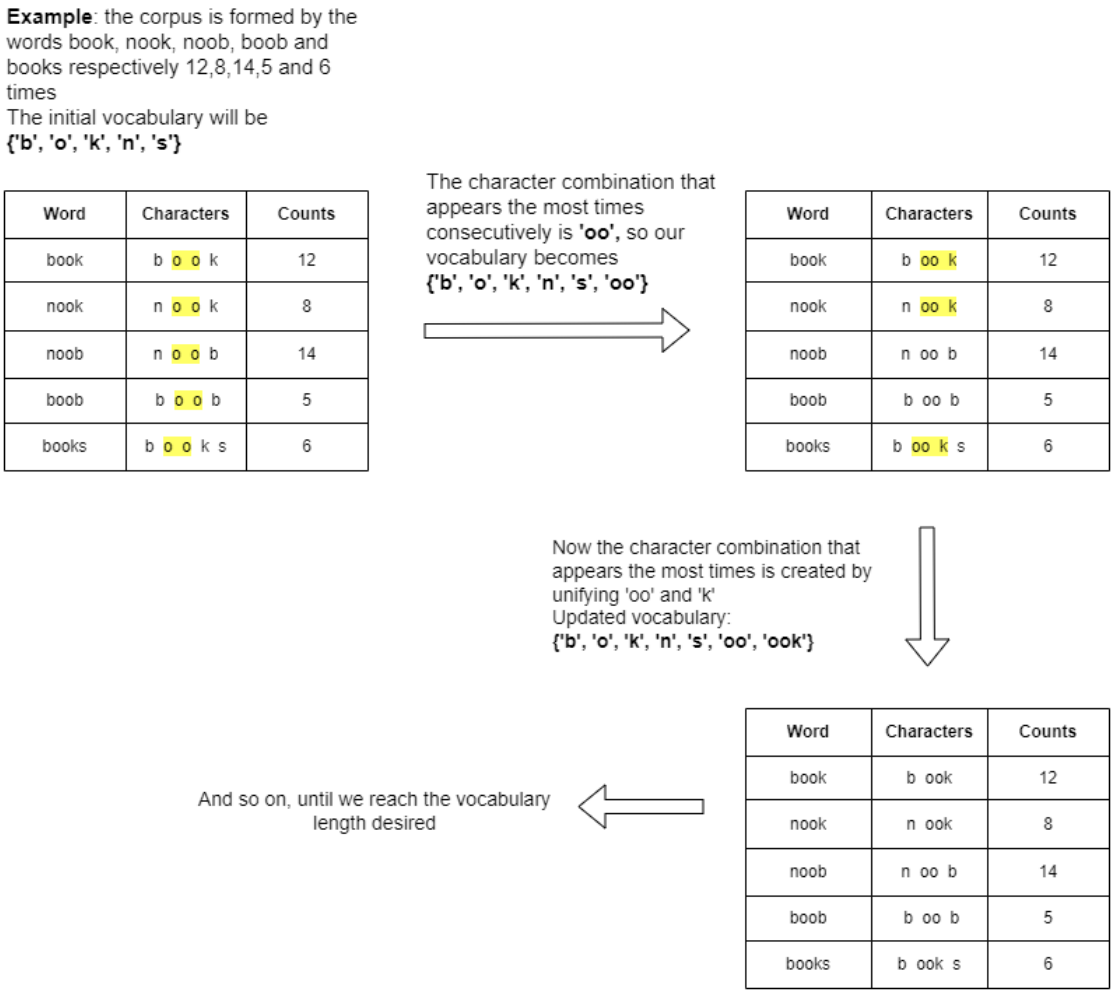


Figure 3.10: Byte-Pair Encoding Tokenization Example

- Rotary Embedding
- The attention layer and the feedforward layer in parallel for decreased communication

Rotary Position Embedding

Position embeddings are used to infer the notion of position to the model, which does not have any sense of position for each token. In other words, using the attention mechanism each token "match" with the other tokens

in the same manner, not considering where if the other token is located right after the current one or if it is at the end of the sentence. Position embeddings are used to add to the model this sense of position.
Rotary position embedding has been introduced by Su et al., it is a novel method that unifies absolute and relative approaches to position embeddings. The typical approach, that is also used by GPT-3, is to use a sinusoidal embedding, which is defined as

$$\begin{cases} p_{1,2t} = sin(k/10000^{2t/d}) \\ p_{1,2t+1} = cos(k/10000^{2t/d}) \end{cases}$$

where $p_{1,2t}$ is the 2^{th} element of the d-dimensional vector p_i RoPE instead proposes to incorporate the relative position information by multiplying the context representation with the sinusoidal functions instead of directly adding them.
If we define

$$\begin{aligned} q_m &= f_q(x_m, m) \\ k_n &= f_k(x_n, n) \end{aligned}$$

where f_q and f_k are functions that incorporates the m^{th} and n^{th} positions respectively to the vector embeddings x_m and x_n to produce the query and key vectors, we can require the inner product of the query q_m and k_n to be formulated by a function g that depends only on the word embeddings x_m, x_n and their relative position $m - n$

$$\langle f_q(x_m, m)\ ,\ f_k(x_n, n)\rangle = g(x_m, x_n, m - n)$$

In the simplest case $d = 2$, $f_{\{q,k\}}$ are defined as

$$f_{(\{q,k\},\{m,n\})} = (W_{\{q,k\}} x_{\{m,n\}}) e^{i\{m,n\}\theta}$$

and therefore we obtain

$$g(x_m, x_n, m - n) = Re[(W_q x_m)(W_k x_n)^T e^{i(m-n)\theta}]$$

which preserves the relative positional information of the word embeddings. This equation is used when calculating the self-attention, which will become

$$Attention(Q, K, V) = softmax\left(\frac{g(x_m, x_n, m - n)}{\sqrt{d}}\right) V$$

This equation can be generalized for $d > 2$, as shown in the original paper. In the end, incorporating the RoTE is pretty straightforward, you just have to rotate the word embedding by a multiple of its position index.
According to the researchers that published GPT-J[10], using RoTE leads to a faster convergence of training and validation losses and a lower overall validation loss.

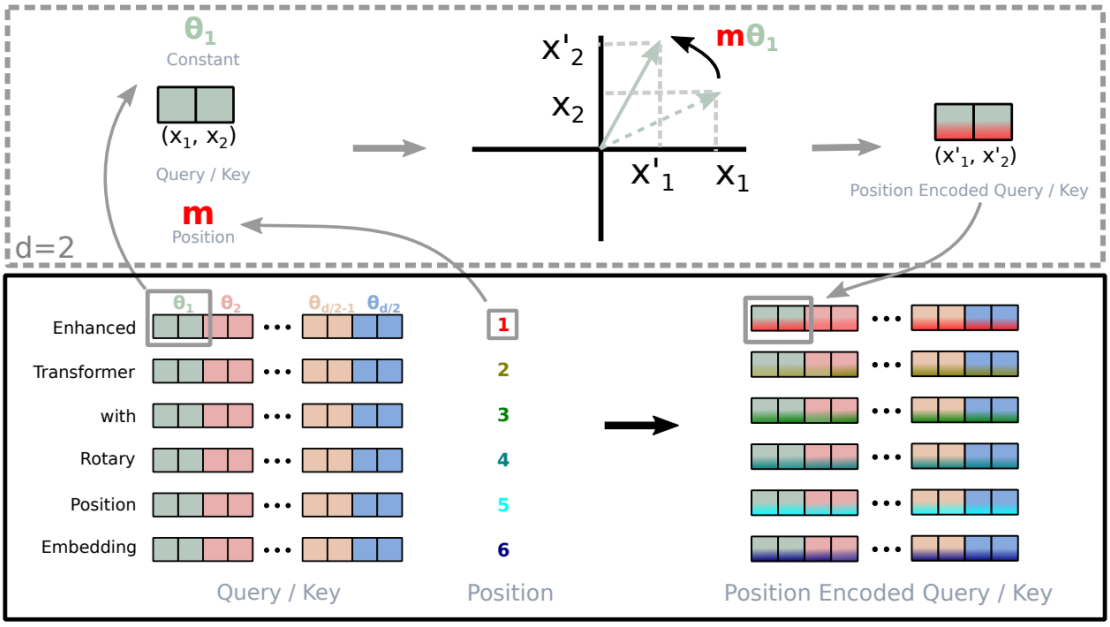


Figure 3.11: Implementation of RoPE, Image taken from original paper

3.5 Survey

In order to understand whether the tickets were similar to real tickets, we conducted a survey internal to the company. We were not able to get real data from the company's HR due to the sensitive nature of the data.
We asked all our collegues to fill a form in an Excel file, where the users were given twenty randomly prompts, similar to the ones that are given to the GPT model.
After giving a brief explanation of the project and what we were trying to achieve, we recommended the users not to use their personal information while writing the fake tickets, but to use the same style of text and the same vocabulary they would use in a real situation.

For each ticket, there were a number of information attached that appeared over all the tickets, which were name of the person, category of the ticket and sub category of the ticket, and some information that were specific to the ticket's category, for example for the category-sub_category 'Salary - Salary Raise' the additional information given was the new requested salary. We asked the users not to use necessarly all the information that were present in the prompts, but only the ones that felt natural to use, and to slightly changing them if it made sense to them (Ex: Sick leave from Oct 20 2022 can become 'from this Thursday, the 20')
Each user was given twenty different prompt sampled randomly from a set of 800 total prompts, belonging to the same categories-sub_categories of our taxonomy. The prompts contained only the contextual information of the ticket and no ticket's text, so, unlike the prompts used for the GPT models, there were no initial text such as 'Dear Sir/Madame, I would like to . . . '
The respondant were also asked to tick the information that they used, and these information was supposed to used as testing in the NER use case. However, many people either did not tick any information or they ticked only partially what thay used, therefore in the end we did not make use of these data.

Ticket details		Detail used (X if yes, empty if not)
First name	Zachary	
Last name	Traore	
Ticket category	Life event	
Ticket sub-category	Health issues	
Date start absence	18/07/2017	
Reason absence	COVID	X
Subject	Request for sick leave for 2 days	
	Ticket text	
	Dear Sir/Madame, I am positive to COVID and I cannot come to work for the next days. I don't exactly know when I will be able to return to work. Zachary	

Figure 3.12: Example of survey

Bibliography

[1] Steven M Bellovin, Preetam K Dutta, and Nathan Reitinger. «Privacy and synthetic datasets». In: *Stan. Tech. L. Rev.* 22 (2019), p. 1.

[2] Ildikó Pilán, Pierre Lison, Lilja Øvrelid, Anthi Papadopoulou, David Sánchez, and Montserrat Batet. «The text anonymization benchmark (tab): A dedicated corpus and evaluation framework for text anonymization». In: *arXiv preprint arXiv:2202.00443* (2022).

[3] Yi-Chun Chen, Tim A Wheeler, and Mykel J Kochenderfer. «Learning discrete Bayesian networks from continuous data». In: *Journal of Artificial Intelligence Research* 59 (2017), pp. 103–132.

[4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. «Language models are unsupervised multitask learners». In: *OpenAI blog* 1.8 (2019), p. 9.

[5] Leo Gao et al. «The pile: An 800gb dataset of diverse text for language modeling». In: *arXiv preprint arXiv:2101.00027* (2020).

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. «Layer normalization». In: *arXiv preprint arXiv:1607.06450* (2016).

[7] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. «Understanding and improving layer normalization». In: *Advances in Neural Information Processing Systems* 32 (2019).

[8] LucyTalks. *The Modern Tokenization Stack for NLP: Byte Pair Encoding*. `https://lucytalksdata.com/the-modern-tokenization-stack-for-nlp-byte-pair-encoding`. [Online; accessed]. 2022.

[9] Ben Wang and Aran Komatsuzaki. *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*. `https://github.com/kingoflolz/mesh-transformer-jax`. May 2021.

[10] Stella Biderman, Sid Black, Charles Foster, Leo Gao, Eric Hallahan, Horace He, Ben Wang, and Phil Wang. *Rotary Embeddings: A Relative Revolution*. `blog.eleuther.ai/`. [Online; accessed]. 2021.