Big Data final project

Student 22045146

University of Arts London

Table of Contents

BIG DATA DOCUMENTATION

Introduction

The idea behind my Big Data project is to explore my native language and the limit of Machine Learning when working with an indigenous language. My family comes from an ethnic republic in Russia that is called Tatarstan. To this day, Tatars are the second largest ethnic minority in Russia with 4.3 million native speakers. My idea was to explore the language, particularly poetry and train a model that would generate poems based on the scrapped dataset. I am not a native speaker of Tatar, I was studying it when I was a child and my grandparents were speaking with me in Tatar, but it was not my first language. When I grew up, I became very interested in Tatar, its origin and the language itself. That is why I decided to do this project. Besides, training a language model based on indigenous language dataset is also quite experimental in terms of research of capabilities of the Machine Learning models at the moment. The creative outcome of my project is a paper zine that contains the real poems, its translation and the poems that were generated with the model that I trained for comparison.

· *Data*

I scraped the data using different resources throughout my project. When I was training the dataset in English, I used several websites that I scraped from the texts of the poems. When I was preparing my dataset in Tatar language, I found way more resources and scrapped both from the websites and pdf files.

· *Motivation*

With my project I am intending to explore my native language heritage as well as to explore capabilities of the Machine Learning models. Besides, I think this project is a great way to educate other people about the vibrance of cultures and heritage of small ethnicities.

· *Research*

After thorough research about similar projects with Tatar language in particular, I only managed to find a Large Language Model (LLM) that can perform basic language modelling operations in Tatar, for example word analogies or summarization of the text (https://huggingface.co/Tweeties/tweety-tatar-base-7b-v24a). I also found a lot of resources about models that are able to translate Tatar to another language, or models that recognize speech and convert it to text (https://github.com/neurotatarlar/awesome-tatar?tab=readme-ov-file#language-analyzers).

In terms of data that I gathered for scraping, I used a general internet search to find the translation of the poems for the English dataset. I encountered the first obstacle when acquiring the English dataset. Since the Tatar language is indigenous there is not enough data to be found on the Internet. Not all of the poems that are written in Tatar are translated even to Russian, so it was very hard to gather a decent amount of data to train the model in English. For the Tatar dataset, the process was a bit smoother, although when I tried to access the official online catalogue of the library of the Republic of Tatarstan, some of the documents were yet to be scanned and uploaded in the online format. I managed to find the pdf files of the books that contain poems in Tatar and scrapped them using Python and the *pdfplumber* library in particular.

Nevertheless, my final dataset consisted of around 1200 poems of different authors, lengths and themes.

· Ideas

My first idea was to design a language model and train it on an English dataset. The reason why I chose English is that I was not sure if LLMs could work with cyrillic alphabet. Besides, Tatar alphabet not only consists of cyrillic characters, but also has additional characters, such as Әә, Өө, Үү, Җж, Ңн, Һh. Even some of the fonts do not support these characters, so I decided to first train the model with an English dataset. Later when I did my research on Transfer Learning, it turned out I could use the poems in Tatar. Nevertheless, starting with gathering an English dataset showed me the obstacles people can face when trying to work with an indigenous language if you are not a native speaker.

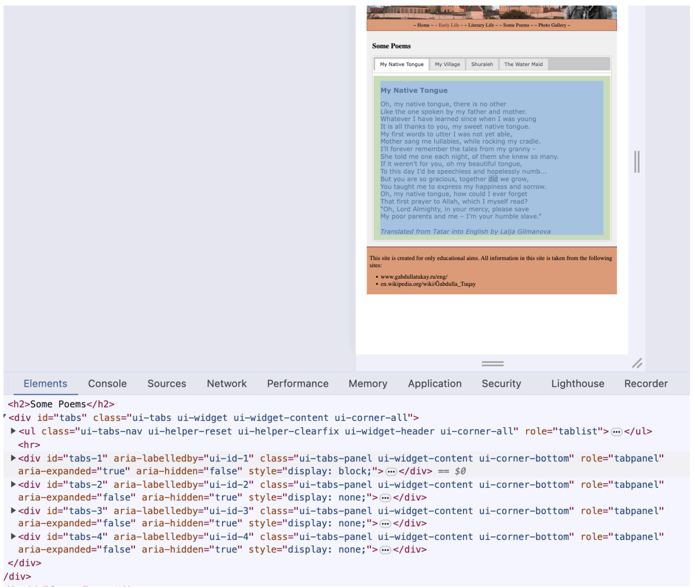
After discarding the first idea, I managed to gather a relatively large dataset in Tatar language. I used pdf files to scrap the data. I was mainly focusing on the poetry of the most renowned authors. The poems were about love, native language, motherland, religion, family and other topics. The sizes of the poems are also different, some are just 4 lines, some are more than one A4 page long.

Design

· Design process

First stage: working with an English dataset.

To scrape the data using Python I utilised such libraries as *BeautifulSoup* and *requests*. For the data I only found 2 websites that had Tatar poems translated into English. The first website [http://tukay.informatikaexpert.ru/SomePoems.html] was pretty easy to scrap.



Some Poems

My Native Tongue | My Village | Shuraleh | The Water Maid

My Native Tongue

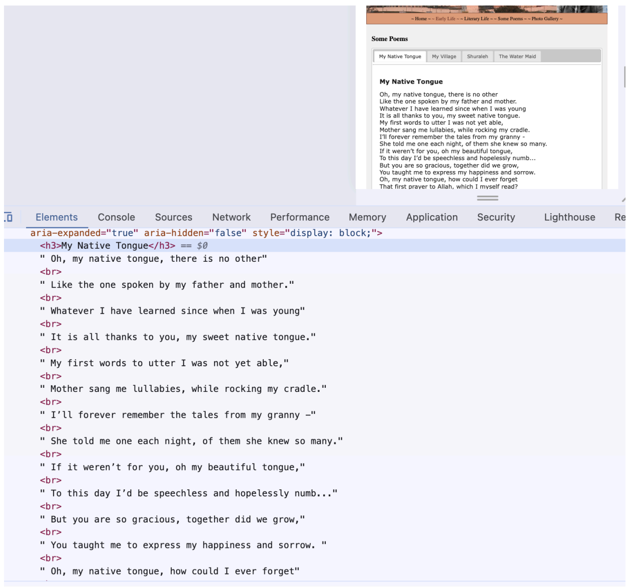
Oh, my native tongue, there is no other
Like the one spoken by my father and mother.
Whatever I have learned since when I was young
It is all thanks to you, my sweet native tongue.
My first words to utter I was not yet able,
Mother sang me lullabies, while rocking my cradle.
I'll forever remember the tales from my granny -
She told me one each night, of them she knew so many.
If it weren't for you, oh my beautiful tongue,
To this day I'd be speechless and hopelessly numb...
But you are so gracious, together did we grow,
You taught me to express my happiness and sorrow.
Oh, my native tongue, how could I ever forget
That first prayer to Allah, which I myself read?
"Oh, Lord Almighty, in your mercy, please save
My poor parents and me – I'm your humble slave."

Translated from Tatar into English by Lalja Gilmanova

This site is created for only educational aims. All information in this site is taken from the following sites:

- www.gabdullatukay.ru/eng/
- en.wikipedia.org/wiki/Ğabdulla_Tuqay

```
<h2>Some Poems</h2>
<div id="tabs" class="ui-tabs ui-widget ui-widget-content ui-corner-all">
  <ul class="ui-tabs-nav ui-helper-reset ui-helper-clearfix ui-widget-header ui-corner-all" role="tablist">…</ul>
  <hr>
  <div id="tabs-1" aria-labelledby="ui-id-1" class="ui-tabs-panel ui-widget-content ui-corner-bottom" role="tabpanel"
  aria-expanded="true" aria-hidden="false" style="display: block;">…</div> == $0
  <div id="tabs-2" aria-labelledby="ui-id-2" class="ui-tabs-panel ui-widget-content ui-corner-bottom" role="tabpanel"
  aria-expanded="false" aria-hidden="true" style="display: none;">…</div>
  <div id="tabs-3" aria-labelledby="ui-id-3" class="ui-tabs-panel ui-widget-content ui-corner-bottom" role="tabpanel"
  aria-expanded="false" aria-hidden="true" style="display: none;">…</div>
  <div id="tabs-4" aria-labelledby="ui-id-4" class="ui-tabs-panel ui-widget-content ui-corner-bottom" role="tabpanel"
  aria-expanded="false" aria-hidden="true" style="display: none;">…</div>
</div>
/div>
```



Some Poems

My Native Tongue | My Village | Shuraleh | The Water Maid

My Native Tongue

Oh, my native tongue, there is no other
Like the one spoken by my father and mother.
Whatever I have learned since when I was young
It is all thanks to you, my sweet native tongue.
My first words to utter I was not yet able,
Mother sang me lullabies, while rocking my cradle.
I'll forever remember the tales from my granny -
She told me one each night, of them she knew so many.
If it weren't for you, oh my beautiful tongue,
To this day I'd be speechless and hopelessly numb...
But you are so gracious, together did we grow,
You taught me to express my happiness and sorrow.
Oh, my native tongue, how could I ever forget
That first prayer to Allah, which I myself read?

```
aria-expanded="true" aria-hidden="false" style="display: block;">
  <h3>My Native Tongue</h3> == $0
  " Oh, my native tongue, there is no other"
  <br>
  " Like the one spoken by my father and mother."
  <br>
  " Whatever I have learned since when I was young"
  <br>
  " It is all thanks to you, my sweet native tongue."
  <br>
  " My first words to utter I was not yet able,"
  <br>
  " Mother sang me lullabies, while rocking my cradle."
  <br>
  " I'll forever remember the tales from my granny –"
  <br>
  " She told me one each night, of them she knew so many."
  <br>
  " If it weren't for you, oh my beautiful tongue,"
  <br>
  " To this day I'd be speechless and hopelessly numb..."
  <br>
  " But you are so gracious, together did we grow,"
  <br>
  " You taught me to express my happiness and sorrow. "
  <br>
  " Oh, my native tongue, how could I ever forget"
```

It contained 4 poems which were under the *div tag* with an *id* that specified the number of the tab.

```
poems = []
response = requests.get("http://tukay.informatikaexpert.ru/SomePoems.html")
soup = BeautifulSoup(response.content, "html.parser")
for i in range(5):
  tag = soup.find("div", id=f"tabs-{i}") # Find the right tabs for the poem
extraction
  if tag:
      title_tag = tag.find("h1")   # Extract the title of the poem
      title = title_tag.get_text() if title_tag else f"Poem {i}"
      content = tag.get_text()
      poems.append({
          "Title": title,
          "Poem": content.strip()  # Strip whitespace
      })
print(poems)
```

Then using the *csv* library, I append the content to the csv file.

```
fieldnames = ['Title', 'Poem']
with open('poems.csv', mode='a', newline='', encoding='utf-8') as file:
   writer = csv.DictWriter(file, fieldnames=fieldnames, quoting=csv.QUOTE_ALL)
   for poem in poems:
       writer.writerow(poem)
```

The second website required a bit more work to do [http://gabdullatukay.ru/eng/work/]. The poems were divided by the years when they were written. Firsty, I skimmed through all of the pages to collect all of the <*a href*> tags. After that, I extracted the title and the content of each poem. Lastly, I added them to the csv file named *"eng_poems.csv"*.

```
for i in range(8):
   response2 = requests.get(f"http://gabdullatukay.ru/eng/work/page/{i+1}/")
   soup = BeautifulSoup(response2.content, "html.parser")
   poems = soup.find_all('a', rel='bookmark')
   # Collect all poem links
   for poem in poems:
       all_ahref.append(poem['href'])
for url in all_ahref:
   response = requests.get(url)
   soup = BeautifulSoup(response.content, 'html.parser')
   title_tag = soup.find('h1', class_='entry-title')
   content_tags = soup.find_all('p')  # Find all <p> tags
   if title_tag and content_tags:
       title = title_tag.get_text().strip()
       content_lines = []
       for tag in content_tags:
           text = tag.get_text().strip()
           # Stop parsing if the word "Translated" is found in the text
           if "Translated" in text or "Оригинал" in text or re.search(r'\b\d{4}\b',
text):
               break
```

```
            if text:  # Only add non-empty lines
                content_lines.append(text)
        content = '\n'.join(content_lines)
        all_poems.append({
            "Title": title,
            "Poem": content
        })
```

After that, I combined the 2 csv files to get the final dataset in english.

```
"Title","Poem"
"Gabdulla Tukay   My Legacy","My tired soul, let peace be your reward,
It’s time to go to Allah, meet the Lord.
Lift up your head; reveal your honest eyes,
Then take one final look into the skies.
My friends, let mullahs heed my last request
For them to sing my verse, when I am put to rest."
"Gabdulla Tukay   The Reply","Pushkin and Lermontov, two poets, two stars…
The third one is Tukai, he comes from the Tatars.
The venom you just drank exacerbates the pain,
Dreams of fortune and fame drive you insane.
The sea is infinite, no matter how you try,
You spill the water, and your tongue is dry.
You, sordid, miserable creature, stay away
Find other dogs like you, and join the fray,
Bark from a distance at the Milky Way."
"Gabdulla Tukay   From Shakespeare","What is it sitting on my loved one’s cheek?
An arrogant fly somehow managed to sneak;
What a happy creature this fly must be
Almighty God, why couldn’t it be me?"
"Gabdulla Tukay   Winter, Farewell","Winter go away, we’re tired of the cold.
Welcome spring is here, her carriage is of gold.
Vanish, Santa Clause, keep safe your snowy beard,
Watch out for the flowers you hated and feared.
The spring does not need a rifle or a sword
Her suite of butterflies obeys her every word."
"Gabdulla Tukay   Miracle","“All magic ended with the Prophet’s death; and never
Will a camel rise from stone, and no endeavor
The moon’s great sphere shall split in two,” the sages say.
Should I believe them? Nay… A miracle occurred today.
It happened, that a friend of mine was passing by,
And paid me back his debt – God only knows why!"
```

Then I proceeded to create the model and train it on this dataset.

- https://youtu.be/QM5XDc4NQJo?si=EiNvI-l_-6KQJVC2
- https://youtu.be/ZMudJXhsUpY?si=7ScDvT-83emeLKH1

The links above are the tutorial videos I used to understand how to create a model and train it to generate the text that would be rhyming in a poetry style. I used *TensorFlow* library for the model.

```
# Initialize the tokenizer
tokenizer = Tokenizer()
# Read the file containing the poems
filepath = 'combined_poems_eng.csv'
data = open(filepath, 'rb').read().decode(encoding='utf-8').lower()
# Split the data into individual lines/poems
```

```
corpus = data.lower().split("\n")
# Fit the tokenizer on the corpus to build the word index
tokenizer.fit_on_texts(corpus)
total_words = len(tokenizer.word_index) + 1  # Total unique words + 1 for padding
# Prepare input sequences
input_sequences = []
for line in corpus:
    token_list = tokenizer.texts_to_sequences([line])[0]  # Convert text to sequence
of integers
    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i+1]  # Create n-gram sequences
        input_sequences.append(n_gram_sequence)
# Pad sequences to ensure they are of the same length
max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len,
padding='pre'))
# Create predictors (xs) and labels (ys)
xs, labels = input_sequences[:, :-1], input_sequences[:, -1]
ys = tf.keras.utils.to_categorical(labels, num_classes=total_words)  # One-hot
encode the labels
# Build the model
model = Sequential()
model.add(Embedding(total_words, 100, input_length=max_sequence_len-1))  #
Embedding layer
model.add(Bidirectional(LSTM(150)))  # Bidirectional LSTM layer
model.add(Dense(total_words, activation='softmax'))  # Output layer with softmax
activation
adam = Adam(learning_rate=0.01)  # Optimizer
model.compile(loss='categorical_crossentropy', optimizer=adam,
metrics=['accuracy'])  # Compile the model
# Train the model
history = model.fit(xs, ys, epochs=100, verbose=1)
# Seed text for generating new text
seed_text = 'My tired soul, let peace be your reward'
next_words = 100
for _ in range(next_words):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]  # Convert seed text
to sequence
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1,
padding='pre')  # Pad the sequence
    predicted = np.argmax(model.predict(token_list), axis=-1)  # Predict the next
word
    output_word = ""
    for word, index in tokenizer.word_index.items():  # Find the word corresponding
to the predicted index
```

```
       if index == predicted:
           output_word = word
           break
   seed_text += " " + output_word  # Append the predicted word to the seed text
# Print the generated text
print(seed_text)
```

First, I load the data from the csv file and break into separate words for tokenization. After creating the sequences and padding them for the uniform size of the input, I am starting to build the model. The *Sequential()* structure helps to organise the neural network that predicts the sequence of words in the poem. The first layer is an embedding layer that converts word indices into dense vectors of fixed size. The next layer is a Bidirectional LSTM (Long Short-Term Memory) layer which processes the sequence in forward and backward directions. The final layer is a dense layer with softmax activation which outputs the probability distribution over the vocabulary. After that, I train the model with a pre-specified number of epochs. After training the model, it is able to generate the most likely sequence of the words.

This was the result of the model:

“ My tired soul, let peace be your reward
both bitter and commands of agree his clear goal droplets of oil toes
toes toes oil will last toes last last chatters after last chatters toes southward last toes
austere oil his clear of early shield
and zuhra’s chatters rain clear clear relief droplets of dew last last line unfair graceful
last clear loved one’s chatters will last clear of oil unfair of early woman last laughter after last hand and chatters of spoil start last breach spoil clear good of his toes and chatters last unfair his torch of disgrace of oil birds of death instead last clear all ”

The result does not make much sense. One of the probable reasons why it turned out this way is the size of the training dataset. The dataset contained fewer than 100 poems, which is insufficient for training a language model to achieve decent results.

Another reason why this dataset and the model did not make sense is that I was focusing on exploring the actual Tatar language and poetry. When the poems are translated, they lose their “uniqueness” with the translation. Since I could only gather small training dataset which cannot provide good results, I decided to try and gather the dataset in Tatar and use a pre-trained model for text generation.

To scrap the data from the pdf files, I found a library called *pdfplumber*. This was the following process for all of the pdf files:

1. Read the pdf file using the *pdfplumber* library
```
with pdfplumber.open('tukai3.pdf') as pdf:
   text_data = ''
   for page in pdf.pages:
       text_data += page.extract_text()
```

2. Save the extracted data in the txt file
```
with open('extracted_text.txt', 'w') as text_file:
   text_file.write(text)
```

3. Clean the data, separate the title and the content, clean the content of the data

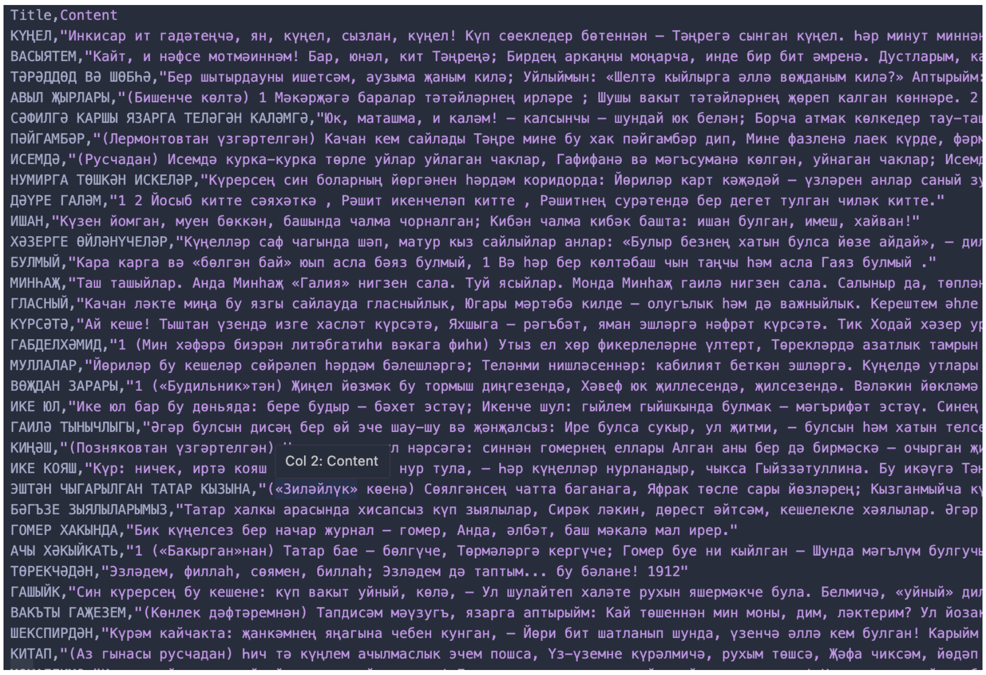
```python
def clean_text(text):
    # Remove text within square brackets
    text = re.sub(r'\[.*?\]', '', text)
    # Remove all digits
    text = re.sub(r'\d+', '', text)
    # Remove any extra whitespace left after removing annotations and
numbers
    text = re.sub(r'\s+', ' ', text).strip()
    return text
```

4. Separate the poems, so that each poem would fit into one line in csv file

```python
for line in poems:
    line = line.strip()
    if line.isupper():
        if current_title:
            poems_dict[current_title] = '
'.join(current_content).strip()
        current_title = line
        current_content = []
    else:
        current_content.append(line)
```

5. Append the lines to the dictionary
6. Convert the dictionary to the DataFrame and add it to the csv file.

This is the layout of the final csv file:



```
Title,Content
КҮҢЕЛ,"Инкисар ит гадәтеңчә, ян, күңел, сызлан, күңел! Күп сөекледер бөтеннән — Тәңрегә сынган күңел. һәр минут миннән
ВАСЫЯТЕМ,"Кайт, и нәфсе мотмәиннәм! Бар, юнәл, кит Тәңреңә; Бирдең аркаңны моңарча, инде бир бит әмренә. Дустларым, ка
ТӘРӘДДӨД ВӘ ШӨБҺӘ,"Бер шытырдауны ишетсәм, аузыма җаным килә; Уйлыймын: «Шелтә кыйлырга әллә вөҗданым килә?» Аптырыйм:
АВЫЛ ҖЫРЛАРЫ,"(Бишенче көлтә) 1 Мәкәрҗәгә баралар тәтәйләрнең ирләре ; Шушы вакыт тәтәйләрнең җөреп калган көннәре. 2
СӘФИЛГӘ КАРШЫ ЯЗАРГА ТЕЛӘГӘН КАЛӘМГӘ,"Юк, маташма, и каләм! — калсынчы — шундай юк белән; Борча атмак көлкедер тау-таш
ПӘЙГАМБӘР,"(Лермонтовтан үзгәртелгән) Качан кем сайлады Тәңре мине бу хак пәйгамбәр дип, Мине фазленә лаек күрде, фәрм
ИСЕМДӘ,"(Русчадан) Исемдә курка-курка төрле уйлар уйлаган чаклар, Гафифанә вә мәгъсуманә көлгән, уйнаган чаклар; Исемд
НУМИРГА ТӨШКӘН ИСКЕЛӘР,"Күрерсең син боларның йөргәнен һәрдәм коридорда: Йөриләр карт кәҗәдәй — үзләрен анлар саный зу
ДӘҮРЕ ГАЛӘМ,"1 2 Йосыб китте сәяхәткә , Рәшит икенчеләп китте , Рәшитнең сурәтендә бер дегет тулган чиләк китте."
ИШАН,"Күзен йомган, муен бөккән, башында чалма чорналган; Кибән чалма кибәк башта: ишан булган, имеш, хайван!"
ХӘЗЕРГЕ ӨЙЛӘНҮЧЕЛӘР,"Күңелләр саф чагында шәп, матур кыз сайлыйлар анлар: «Булыр безнең хатын булса йөзе айдай», — дил
БУЛМЫЙ,"Кара карга вә «бөлгән бай» юып асла бәяз булмый, 1 Вә һәр бер көлтәбаш чын таңчы һәм асла Гаяз булмый ."
МИНҺАҖ,"Таш ташыйлар. Анда Минһаҗ «Галия» нигзен сала. Туй ясыйлар. Монда Минһаҗ гаилә нигзен сала. Салыныр да, төплән
ГЛАСНЫЙ,"Качан ләкте миңа бу язгы сайлауда гласныйлык, Югары мәртәбә килде — олугълык һәм дә важныйлык. Керештем әһле
КҮРСӘТӘ,"Ай кеше! Тыштан үзендә изге хасләт күрсәтә, Яхшыга — рәгъбәт, яман эшләргә нәфрәт күрсәтә. Тик Ходай хәзер ур
ГАБДЕЛХӘМИД,"1 (Мин хәфәрә биэрән литәбгатиһи вәкага фиһи) Утыз ел хөр фикерлеләрне үлтерт, Төрекләрдә азатлык тамрын
МУЛЛАЛАР,"Йөриләр бу кешеләр сөйрәлеп һәрдәм бәлешләргә; Теләнми нишләсеннәр: кабилият беткән эшләргә. Күңелдә утлары
ВӨҖДАН ЗАРАРЫ,"1 («Будильник»тән) Җиңел йөзмәк бу тормыш диңгезендә, Хәвеф юк җиллесендә, җилсезендә. Вәләкин йөкләмә
ИКЕ ЮЛ,"Ике юл бар бу дөньяда: бере будыр — бәхет эстәү; Икенче шул: гыйлем гыйшкында булмак — мәгърифәт эстәү. Синең
ГАИЛӘ ТЫНЫЧЛЫГЫ,"Әгәр булсын дисәң бер өй эче шау-шу вә җәнҗалсыз: Ире булса сукыр, ул җитми, — булсын һәм хатын телсе
КИҢӘШ,"(Позняковтан үзгәртелгән) "             л нәрсәгә: синнән гомернең еллары Алган аны бер дә бирмәскә — очырган җи
ИКЕ КОЯШ,"Күр: ничек, иртә кояш               нур тула, — һәр күңелләр нурланадыр, чыкса Гыйззәтуллина. Бу икәүгә Тән
ЭШТӘН ЧЫГАРЫЛГАН ТАТАР КЫЗЫНА,"(«Зиләйлүк» көенә) Сөялгәнсең чатта баганага, Яфрак төсле сары йөзләрең; Кызганмыйча ку
БӘГЪЗЕ ЗЫЯЛЫЛАРЫМЫЗ,"Татар халкы арасында хисапсыз күп зыялылар, Сирәк ләкин, дөрест әйтсәм, кешелекле хәялылар. Әгәр
ГОМЕР ХАКЫНДА,"Бик күңелсез бер начар журнал — гомер, Анда, әлбәт, баш мәкалә мал ирер."
АЧЫ ХӘКЫЙКАТЬ,"1 («Бакырган»нан) Татар бае — бөлгүче, Төрмәләргә кергүче; Гомер буе ни кыйлган — Шунда мәгълүм булгучь
ТӨРЕКЧӘДӘН,"Эзләдем, филлаһ, сөямен, биллаһ; Эзләдем дә таптым... бу бәлане! 1912"
ГАШЫЙК,"Син күрерсең бу кешене: күп вакыт уйный, көлә, — Ул шулайтеп халәте рухын яшермәкче була. Белмичә, «уйный» дил
ВАКЪТЫ ГАҖЕЗЕМ,"(Көнлек дәфтәремнән) Тапдисәм мәүзугъ, язарга аптырыйм: Кай төшеннән мин моны, дим, ләктерим? Ул йозак
ШЕКСПИРДӘН,"Күрәм кайчакта: җанкәмнең яңагына чебен кунган, — Йөри бит шатланып шунда, үзенчә әллә кем булган! Карыйм
КИТАП,"(Аз гынасы русчадан) һич тә күңлем ачылмаслык эчем пошса, Үз-үземне күрәлмичә, рухым төшсә, Җәфа чиксәм, йөдәп
```

After gathering the dataset, I started my research on pre-trained models I could use. LLMs have recently gained immense popularity due to their remarkable efficiency in recognizing and generating text. For example, the ChatGPT chatbot, which was released for public access in November 2022, saw its user base grow to 100 million within just two months of its launch (Porter, 2023). Since then, ChatGPT has undergone numerous updates, with the latest versions being based on the GPT-4 Large Multimodal Model (OpenAI, 2024). These recent versions can accept not only text but also images as input (OpenAI, 2024).

The widespread popularity of LLMs, coupled with the ease of access to OpenAI's pre-trained GPT-2 language model for fine-tuning, has motivated this project (Radford et al., 2019).

Addressing the text generation challenge during the fine-tuning phase involves several key steps.

1. Tokenizing the text corpus.
2. Selecting appropriate embeddings.
3. Choosing the relevant model.
4. Searching for optimal hyperparameters.

Firstly, in the context of Natural Language Processing, tokenization involves splitting a text corpus into smaller units, such as words. For example, consider the string "Before the Russian invasion of Tatarstan, Tatar poets used Arabic script." Tokenizing this string by words would result in ["Before", "the", "Russian", "invasion", "to", "Tatarstan", ",", "tatar", "poets", "used", "Arabic", "script", "."]. Notice that the punctuation marks, the comma "," and the dot ".", are treated as separate tokens. There are many built-in packages in Python that handle text tokenization, such as *nltk*. However, LLMs often come with their own associated tokenizers. In my case, the tokenizer associated with the GPT-2 model is downloaded from the pre-trained model (Radford et al., 2019).

```
from transformers import AutoTokenizer, TFAutoModelForCausalLM
# Load tokenizer and model
model_name = "gpt2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = TFAutoModelForCausalLM.from_pretrained(model_name)
```

The GPT-2 tokenizer, which is based on Byte Pair Encoding, splits text strings into subwords (Radford et al., 2019). This approach gives the language model a superior ability to handle rare structures that are not present in its token vocabulary (Radford et al., 2019). Let us examine the GPT-2 token vocabulary and view ten random tokens.

```
import random
# Get the vocabulary
vocab = tokenizer.get_vocab()
tokens = list(vocab.keys())
# Print 10 random tokens
random_tokens = random.sample(tokens, 10)  # Get 10 random tokens
print("Random Tokens:", random_tokens)
```

```
Random Tokens: ['Ġstatistic', 'Ġsatirical', '157', 'ju', 'Return', 'ĠPir', 'rug', 'ceive', 'etic', 'ĠHes']
```

Note that the Ġ symbol in the tokenized output signifies a space before the token, indicating that the token represents either an entire word or the beginning of a word.
Since the GPT-2 model has been primarily trained on English-language content sourced from Reddit, subword tokenization approach is particularly beneficial for my objective of working with a Tatar text corpus (Radford et al., 2019). Most importantly, Byte Pair Encoding enables GPT-2 to handle Unicode standard characters, including cyrillic characters and unique Tatar language characters such as *ә, ө, ү, җ, ң, һ*.

For sequential batch model training, all inputs must be of the same length. To achieve this, 1188 poems of varying lengths need to be tokenized and standardised to a uniform length. This standardisation is done by padding, which involves adding special padding tokens (recognized by the model) to each tokenized input. The following code appends end-of-sequence tokens to the existing tokens. The length of the tokenized sequence is limited to 256 tokens for memory efficiency.

```
# Add a padding token if not present
if tokenizer.pad_token is None:
   tokenizer.add_special_tokens({'pad_token': tokenizer.eos_token})
   model.resize_token_embeddings(len(tokenizer))
# Tokenize all poems at once
max_length = 256  # Define max length for sequences
inputs = tokenizer(poems, return_tensors='tf', padding='max_length',
truncation=True, max_length=max_length)
```

The dictionary inputs contain two keys: *input_ids* and *attention_mask*. Each key corresponds to tensors of shape (1188, 256), representing the number of poems and uniform length of the tokenized sequence.

The attention mask is a component unique to transformer models, including GPT-2. It consists of elements with values of 1 or 0, where 1 indicates relevant tokens that the model should consider, and 0 indicates padding tokens that should be ignored during training.

The script below prepares the inputs for the GPT-2 model.

```
# Convert tokenized inputs to TensorFlow dataset
input_ids = inputs['input_ids']
attention_masks = inputs['attention_mask']
dataset = tf.data.Dataset.from_tensor_slices((input_ids, attention_masks))
```

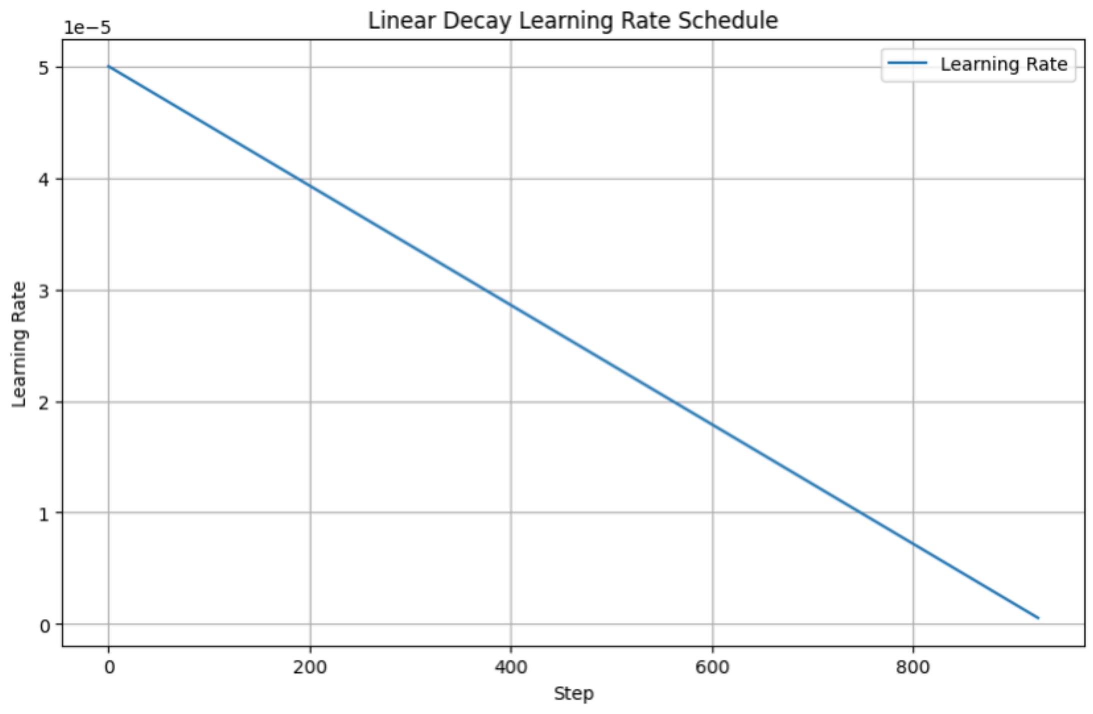
One of the initial layers in language models is the embedding layer, which maps unique token IDs to their corresponding embeddings. An embedding is a vector representation of a token that serves as the model's input.

Model training is conducted in batches of 32, with the number of training epochs set to 25. These values have been selected after some experimentation and provide optimal results. Overall, there are 925 iterations through the dataset, calculated as 1188 poems divided by 32 poems per batch, multiplied by 25 training epochs.

Learning rate schedule is set as follows.

```
learning_rate_schedule = PolynomialDecay(
   initial_learning_rate=5e-5,
   end_learning_rate=5e-7,
   decay_steps=total_steps,
   power=1.0  # Linear decay
)
```

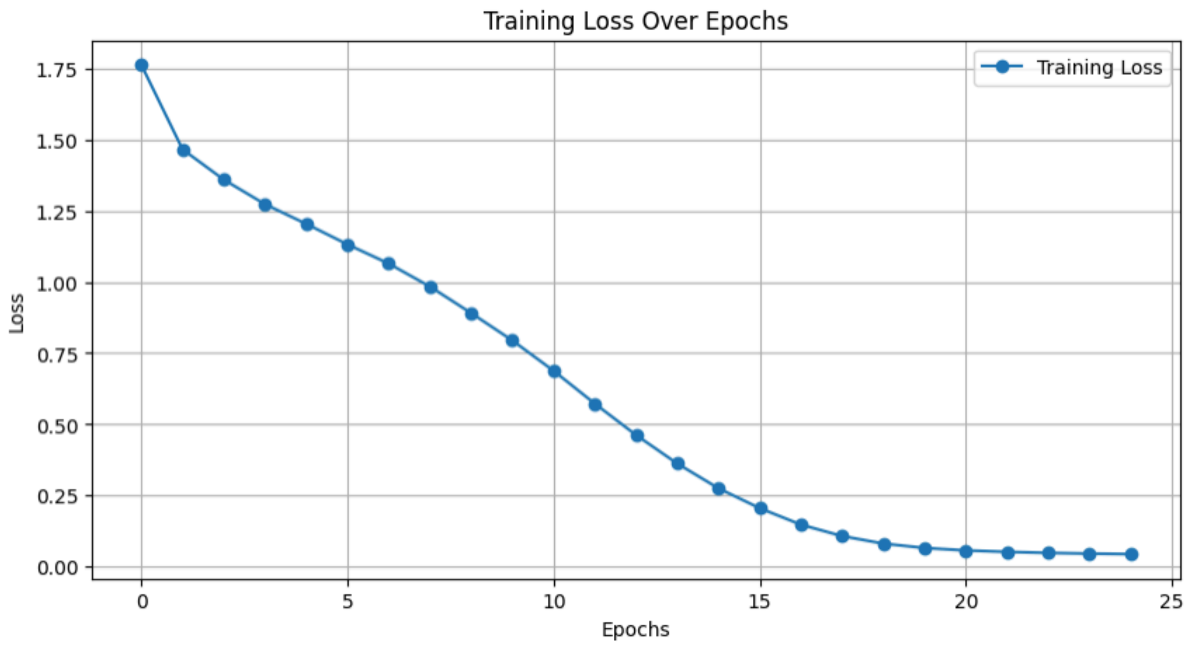
The learning rate decreases gradually in a linear manner as the number of iterations through the dataset increases, as illustrated in the figure below. The initial learning rate is set to a small value of 0.00005. This decaying learning rate and the small initial value are designed to leverage the pre-trained knowledge of the GPT-2 language model.



In essence, during the training phase, the loss function is minimised with respect to the embeddings and GPT-2 model parameters. These embeddings and model parameters are initially set to the corresponding values from the pre-trained GPT-2 model and are fine-tuned throughout the training process. Calibration of embeddings and GPT-2 model parameters is known as optimization. The following script defines the *AdamWeightDecay* optimizer, which combines the *Adam* optimization algorithm with weight decay regularisation. This algorithm provides benefits for fine-tuning deep learning models, such as GPT-2.

```
# Define the optimizer
optimizer = AdamWeightDecay(learning_rate=learning_rate_schedule)
```

Furthermore, the training phase follows a standard procedure involving automatic gradient calculation and loss function minimization using the built-in functionality of the *TensorFlow* Python library. The following figure shows the dynamics of average loss per epoch, clearly demonstrating that the loss consistently decreases, nearly reaching zero.



Once the fine-tuning of the GPT-2 language model is completed, we can begin generating Tatar poems based on Tatar input. The following function is used for poem generation.

```
def generate_poem(prompt: str, temperature, top_k, top_p, min_length, max_length)
-> str:
   input_ids = tokenizer.encode(prompt, return_tensors='tf')
   attention_mask = tf.ones_like(input_ids)  # Create attention mask
   generated_text_ids = model.generate(
       input_ids,
       attention_mask=attention_mask,
       min_length=min_length,
       max_length=max_length,
       num_return_sequences=1,
       temperature=temperature,
       top_k=top_k,
       top_p=top_p,
       pad_token_id=tokenizer.eos_token_id
   )
   generated_text = tokenizer.decode(generated_text_ids[0],
skip_special_tokens=True)
   return generated_text
```

Parameters of the function

- *temperature* controls the randomness of predictions; values below the default of 1.0 reduce randomness, while values above 1.0 increase it,
- the larger the *top_k*, the greater the range of next words considered,

- *top_p* helps achieve a balance between the variety of outputs and output coherence.

These parameter values are calibrated through experimentation.

With the dataset of a bit more than 400 poems, the model took a couple of hours to train. The results were still not as good as I expected them to be. Here are some of the outputs.

Output	Translation
“ Мин сине яратам шагыйрьнеӊ шул шунда Мин сине яратамын? Норадын калдырды. Мин сине яратам шәберенә бер изге хәберенә Мин сине яратам. КҮрәк икән алтын яратам. Мин сине баратам. КҮрәк икәннә алтын яратам.”	“I love you where the poet Do I love you? Left Norad. I love you for one holy message I love you. If a shovel, I love gold. I'm coming to you. I love gold on crops.”
“Мин сине яратам Җаный, Мин сине минеӊ начамый, Белмимимимимимимими; КҮзеӊез хиселек исә татумый, КҮзеӊезләнеп тугъръм булган киӊгырем кашыма. КҮзеӊезләргә калган кагый. Ай-Ай Ай”	“I love you, soul, I do not love you, Belmimimimimi; Your eyes don't feel, Your eyes look at me. What's left of the eye. Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon-Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon Moon”

The first few words after the prompt make some kind of sense, but then it turns to the repeating of one word.

Subsequently, I increased the dataset to 1188 poems and retrained the model. The training took 15 hours on my computer, and the results were comparatively more satisfactory.

Output	Translation
“Мин сине яратам. Сине берҮксез искән? Кайдан кҮктә булып карап китә. Син берҮксез искән кайтып карап китә. Берсеннән тҮгел иӊгән бер хәзер Синдем кҮген кҮгем бән бәмнән Җитсән Җитсән Җит”	“I love you. Do you love me? From where he looks at the sky. You come back from nowhere. Not one descends now”
“Туган тел, Бу кунган телсә инде, КҮӊел эресендер кҮресендер.	“Native language, If I wanted, I would fight for the soul.

Күресен күресен! Күп яшьрен телəм йолдыз күзлəре. ‶	Let's see, see, see! With age, rainbow star eyes.″
‶Шагыйрь калмый! Атылмасмый! Бусынасын килгəн китеп йөрəк төсле мəхəбəт бит калмый! Батылмасын калмый бу йөрəклəрде мəхəбəт бит калмый!″	‶There is no poet left! Don't shoot! The book he came from never ceases to be heartbreaking love! There is no love in these hearts!″

· Future Implementations

Even though the results are not of the best quality, throughout the iteration of my project, there is a dynamic of improvement. In terms of future implementation of my project, I would look into other pre-trained LLMs such as LLama model which has 32 000 embeddings, while GPT-2 has only 768 embeddings (https://huggingface.co/docs/transformers/main/en/model_doc/llama2). Besides, increasing the training dataset would also be helpful in increasing the quality of the output.

· Challenges

Throughout my project, I faced several challenges. Firstly, gathering the dataset in English was hard and not enough for training a model, which showed the reality of working with an indigenous language translations. Then, the model, which was training on tatar dataset, took some experimental modifications to get the best outcome possible. The first outcomes of the model were mostly repetition of one word all over the length of the output. Then the outputs got better. Another problem I encountered with, I saved my model that I trained, but was not able to load it again. The saved model is uploaded in Github (the folder with the name: “saved_model3”)

This was the error I got when trying to load the model:

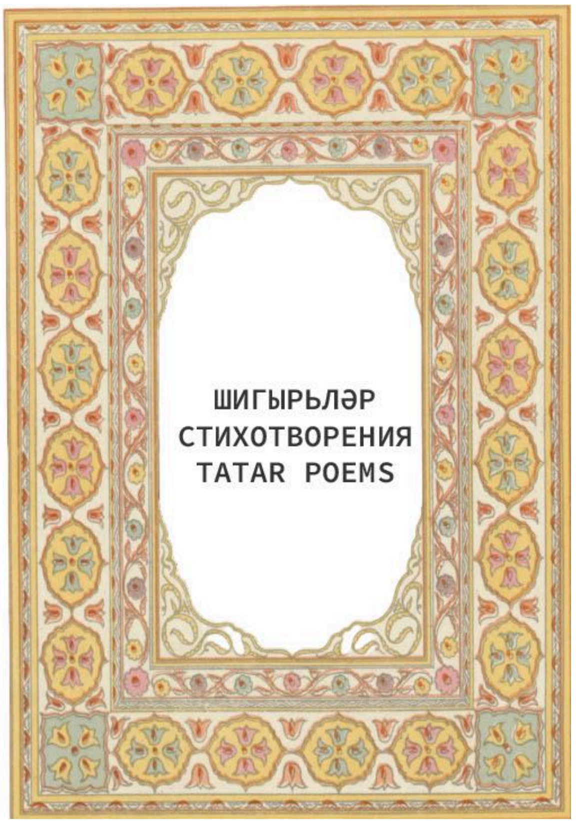
```
        OSError: saved_model3 does not appear to have a file named config.json.
Checkout 'https://huggingface.co/saved_model3/tree/None' for available files.
```

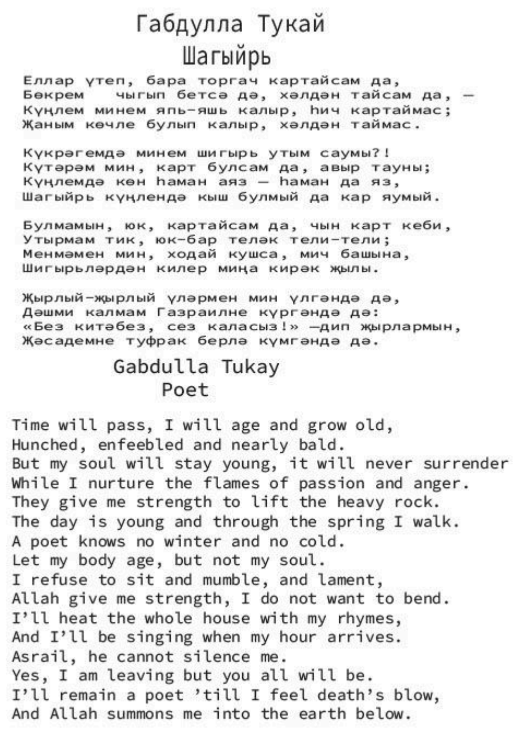
```
# Load the saved model
saved_model_path = 'saved_model3'
loaded_model = tf.saved_model.load(saved_model_path)

tokenizer = AutoTokenizer.from_pretrained(saved_model_path)
model = TFAutoModelForCausalLM.from_pretrained(saved_model_path)
```

· Physical outcome

As a physical outcome of my project, I created a zine that contains the generated poems. The zine was created using *Adobe Illustrator*. I added some illustrations of the traditional Tatar ornaments from the book (Speransky, P.T.). The content of the zine is as follows: each album layout page has a poem that was written by a Tatar author, its translation, a poem that was generated by a model on a similar topic and its translation. I thought it would be interesting to show both human-written and machine-generated poems to feel the differences. The zine is bilingual containing poems both in Tatar and the English translations.





Габдулла Тукай
Шагыйрь

Еллар үтеп, бара торгач картайсам да,
Бөкрем   чыгып бетсә дә, хәлдән тайсам да, —
Күңлем минем япь-яшь калыр, һич картаймас;
Җаным көчле булып калыр, хәлдән таймас.

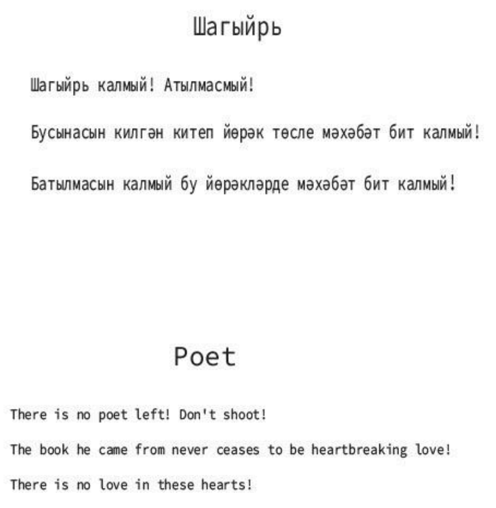
Күкрәгемдә минем шигырь утым саумы?!
Күтәрәм мин, карт булсам да, авыр тауны;
Күңлемдә көн һаман аяз — һаман да яз,
Шагыйрь күңлендә кыш булмый да кар яумый.

Булмамын, юк, картайсам да, чын карт кеби,
Утырмам тик, юк-бар теләк тели-тели;
Менмәмен мин, ходай кушса, мич башына,
Шигырьләрдән килер миңа кирәк җылы.

Җырлый-җырлый үләрмен мин үлгәндә дә,
Дәшми калмам Газраилне күргәндә дә:
«Без китәбез, сез каласыз!» —дип җырлармын,
Җәсадемне туфрак берлә күмгәндә дә.

Gabdulla Tukay
Poet

Time will pass, I will age and grow old,
Hunched, enfeebled and nearly bald.
But my soul will stay young, it will never surrender
While I nurture the flames of passion and anger.
They give me strength to lift the heavy rock.
The day is young and through the spring I walk.
A poet knows no winter and no cold.
Let my body age, but not my soul.
I refuse to sit and mumble, and lament,
Allah give me strength, I do not want to bend.
I'll heat the whole house with my rhymes,
And I'll be singing when my hour arrives.
Asrail, he cannot silence me.
Yes, I am leaving but you all will be.
I'll remain a poet 'till I feel death's blow,
And Allah summons me into the earth below.



Шагыйрь

Шагыйрь калмый! Атылмасмый!

Бусынасын килгән китеп йөрәк төсле мәхәбәт бит калмый!

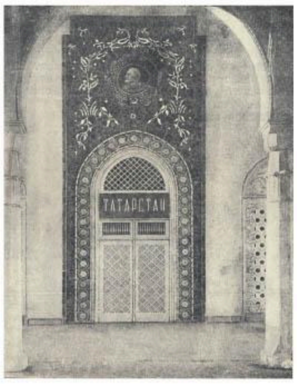
Батылмасын калмый бу йөрәкләрде мәхәбәт бит калмый!

Poet

There is no poet left! Don't shoot!

The book he came from never ceases to be heartbreaking love!

There is no love in these hearts!



Татарстан

Татарстан турында булып карап көньяга.
Тавыштык турында булып карап.
Кагылып та өстендә булып карап… Язык та өстендә булып карап.

Кагылып карап карап күрергә…
Барык та өстендә булып карап…
Язык та өстендә булып карап…

Tatarstan

Looking south at Tatarstan.
Looking at the world.
Touched and looked from above... Looking at the language.

Touch and see...
And everyone was looking at him...
Looking at the language…

Габдулла Тукай
Казан вә Казан арты

И Казан шәһре, торасың тауда зур шәмдәл кеби,
Мәсҗедең, чиркәүләрең, һәр часларың шәмнәр кеби.

Син, үзеңне чорнаган һәрбер өязгә нур чәчеп,
Бик мәһабәтле торасың, барчага юл күрсәтеп.

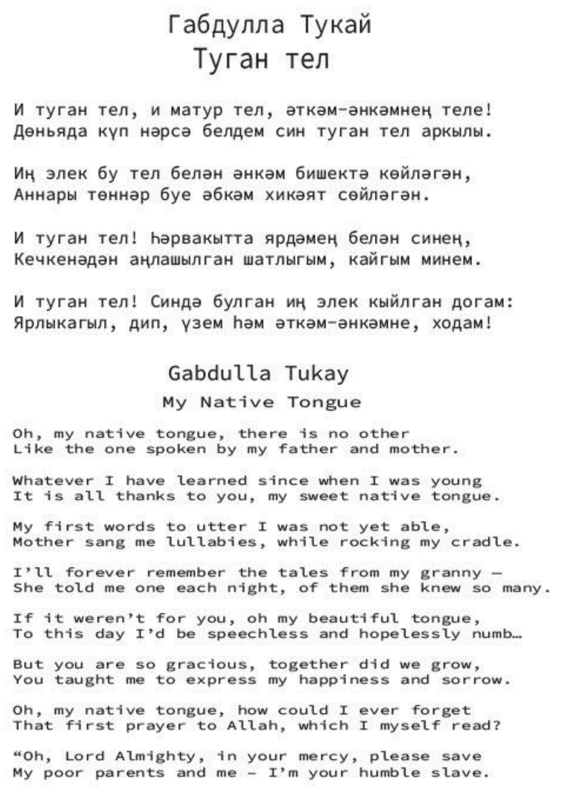
Нур ала синнән бөтен як: Чистапул, Спас, Тәтеш
һәм Чабаксарга, Мамадышларга Чар, Малмыж катыш.

Бер борыл да, и Казан, син бу Казан артын кара:
Нур чәчәсең бар өязгә, үз өязең кап-кара.

«Үз төбенә төшми, ди, шәм-лампаның һичбер нуры», –
Шул мәкаль монда дөрест шул, ах, аны җен оргыры!

Gabdulla Tukay
Kazan and Beyond

The city of Kazan up on a hill is radiant like a torch,
Tall as a candle is each mosque, minaret, and church.
Kazan, your light shines forth for other towns so bright,
Showing all and sundry, which way for them is right.
Tetyushi and Spas, Chary and Chistopol,
They are so dear to me I'd like to name them all.
Kazan, look round at these towns and villages,
While sharing light with them, you linger in dark ages.
The spot beneath a lamp or candle has no light,
God, I hate this wisdom 'cause it's damn so right.⊠



Габдулла Тукай
Туган тел

И туган тел, и матур тел, әткәм-әнкәмнең теле!
Дөньяда күп нәрсә белдем син туган тел аркылы.

Иң элек бу тел белән әнкәм бишектә көйләгән,
Аннары төннәр буе әбкәм хикәят сөйләгән.

И туган тел! Һәрвакытта ярдәмең белән синең,
Кечкенәдән аңлашылган шатлыгым, кайгым минем.

И туган тел! Синдә булган иң элек кыйлган догам:
Ярлыкагыл, дип, үзем һәм әткәм-әнкәмне, ходам!

Gabdulla Tukay
My Native Tongue

Oh, my native tongue, there is no other
Like the one spoken by my father and mother.

Whatever I have learned since when I was young
It is all thanks to you, my sweet native tongue.

My first words to utter I was not yet able,
Mother sang me lullabies, while rocking my cradle.

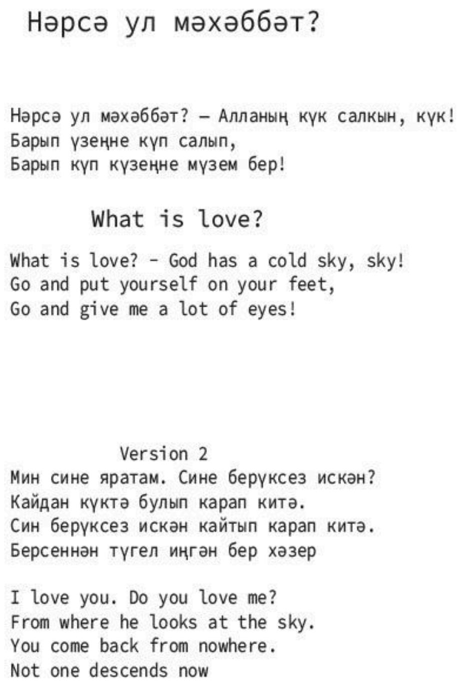
I'll forever remember the tales from my granny –
She told me one each night, of them she knew so many.

If it weren't for you, oh my beautiful tongue,
To this day I'd be speechless and hopelessly numb…

But you are so gracious, together did we grow,
You taught me to express my happiness and sorrow.

Oh, my native tongue, how could I ever forget
That first prayer to Allah, which I myself read?

"Oh, Lord Almighty, in your mercy, please save
My poor parents and me – I'm your humble slave.



Нәрсә ул мәхәббәт?

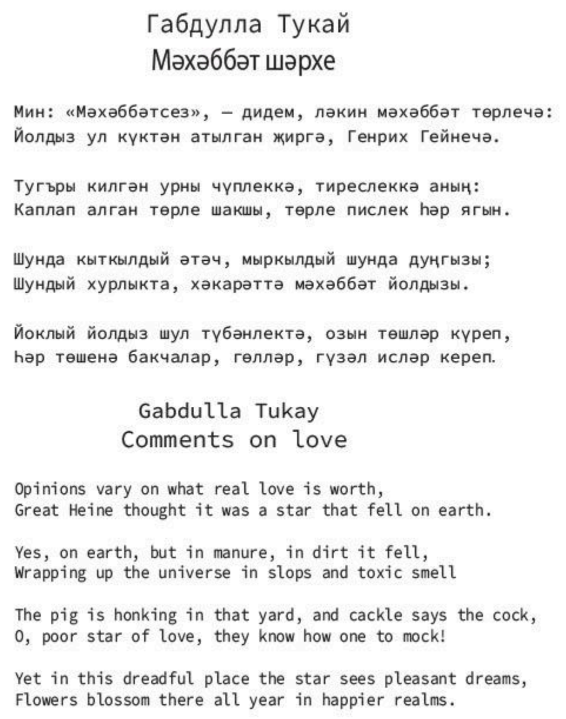
Нәрсә ул мәхәббәт? — Алланың күк салкын, күк!
Барып үзеңне күп салып,
Барып күп күзеңне мүзем бер!

What is love?

What is love? - God has a cold sky, sky!
Go and put yourself on your feet,
Go and give me a lot of eyes!

Version 2
Мин сине яратам. Сине берүксез искән?
Кайдан күктә булып карап китә.
Син берүксез искән кайтып карап китә.
Берсеннән түгел иңгән бер хәзер

I love you. Do you love me?
From where he looks at the sky.
You come back from nowhere.
Not one descends now



Габдулла Тукай
Мәхәббәт шәрхе

Мин: «Мәхәббәтсез», — дидем, ләкин мәхәббәт төрлечә:
Йолдыз ул күктән атылган җиргә, Генрих Гейнечә.

Тугъры килгән урны чүплеккә, тиреслеккә аның:
Каплап алган төрле шакшы, төрле пислек һәр ягын.

Шунда кыткылдый әтәч, мыркылдый шунда дуңгызы;
Шундый хурлыкта, хәкарәттә мәхәббәт йолдызы.

Йоклый йолдыз шул түбәнлектә, озын төшләр күреп,
Һәр төшенә бакчалар, гөлләр, гүзәл исләр кереп.

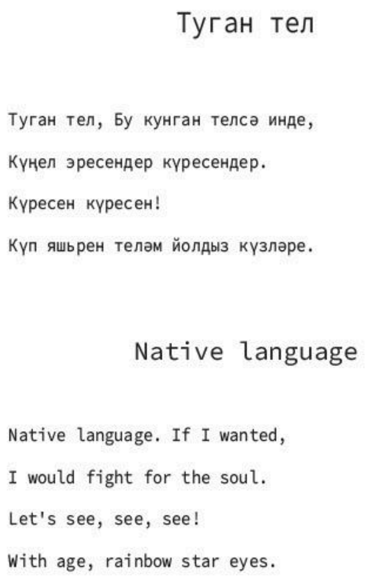
Gabdulla Tukay
Comments on love

Opinions vary on what real love is worth,
Great Heine thought it was a star that fell on earth.

Yes, on earth, but in manure, in dirt it fell,
Wrapping up the universe in slops and toxic smell

The pig is honking in that yard, and cackle says the cock,
O, poor star of love, they know how one to mock!

Yet in this dreadful place the star sees pleasant dreams,
Flowers blossom there all year in happier realms.



Туган тел

Туган тел, Бу кунган телсә инде,

Күңел эресендер күресендер.

Күресен күресен!

Күп яшьрен теләм йолдыз күзләре.

Native language

Native language. If I wanted,

I would fight for the soul.

Let's see, see, see!

With age, rainbow star eyes.

Here is a link for the video presentation of my zine:
https://drive.google.com/file/d/11X6LrqZ0spbmrmRei32wROtsqHttNtJE/view?usp=sharing

Link for video documentation:
https://drive.google.com/file/d/1CXeq69yUNqHAAo9zoISpscvXWd4_UYwD/view?usp=sharing
Link to the github repository: https://git.arts.ac.uk/22046145/bigdata

Bibliograpy

1. Porter, J. (2023) *ChatGPT active user count OpenAI developer conference*, The Verge. Available at: *https://www.theverge.com/2023/11/6/23948386/chatgpt-active-user-count-openai-developer-conference* (Accessed: 5 June 2024).
2. OpenAI (2024) *Models*, OpenAI Platform. Available at: *https://platform.openai.com/docs/models* (Accessed: 5 June 2024).
3. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019) *Language Models are Unsupervised Multitask Learners.*
4. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019) *GPT-2: Code and Models*. Available at: https://github.com/openai/gpt-2 (Accessed: 5 June 2024).
5. Speransky, P.T., Year. Tatar Folk Ornament.

AI Assistance Log Instance

AI Tool used	Purpose	Section of the code
OpenAI ChatGPT 3.5	Understanding complex concepts of machine learning. (e.g Sequential structure with layers, optimizers, learning rate parameters)	Creating a model and training it both in English and Tatar dataset
OpenAI ChatGPT 3.5	Understand what are the best parameters to experiment with when trying to improve quality of the machine learning model	Model training with tatar dataset
OpenAI ChatGPT 3.5	Clarity on how to scrap the data from a PDF files	Scraping the data for the tatar dataset